

## INTERACTIVE IMPLICIT MODELLING BASED ON $C^1$ CONTINUOUS RECONSTRUCTION OF REGULAR GRIDS

LOIC BARTHE

*RWTH-Aachen, Informatik VIII  
Ahornstrasse 55, 52074 Aachen, Germany  
barthe@informatik.rwth-aachen.de*

BENJAMIN MORA

*Equipe synthèse d'images et réalité virtuelle, IRIT-UPS,  
118 route de Narbonne, 31072 Toulouse Cedex 4, France  
mora@irit.fr*

NEIL DODGSON

*Rainbow Group-University of Cambridge  
15 JJ Thomson Avenue, Cambridge CB3 0FD, UK  
nad@cl.cam.ac.uk*

MALCOLM SABIN

*Numerical Geometry Ltd.  
26 Abbey Lane, Lode, Cambridge, UK  
malcolm@geometry.demon.co.uk*

Our goal is to provide a kernel to allow interactive and accurate modelling of volume objects. As a first step, we present a data structure for three-dimensional fields  $C^1$  continuous in the modelling space. Regular grids storing the field values discretely are combined with a triquadratic approximation filter to define volume objects. This association of a grid and an approximation/interpolation reconstruction allows the field to be defined by a  $C^1$  continuous real function and the surface to be directly visualised from its own equation. We show how accurate and high quality interactive visualisation is obtained during the modelling process, and we explain why the visualisation is faithful to the object definition. We also describe, as an example application of our data structure, how advanced Boolean operators realised with soft or free-form transitions are performed under the influence of an interactive modelling tool.

*Keywords:* Implicit modelling, interactive modelling, interactive volume rendering, volume reconstruction.

### 1. Introduction

Many algorithms for the visualisation of discrete three-dimensional volume data now exist and most of them allow interactive visualisation of isopotential surfaces [1,2] or of potential variations in the volume (using transparency or colour variation on the surface) [3,4]. Some new hardware also integrates three-dimensional texture rendering and pixel or vertex shaders, which are efficient tools to increase interactivity in volume visualisations [5]. A significant advantage of this representation is that it offers an alternative to classical polygon

rendering while avoiding some topological problems resulting from shape polygonalisation [6].

While fast visualisation is available, techniques to model such data interactively need to be improved and new models proposed.

Our aim is to propose a unified structure to precisely and interactively model and visualise volume objects. The modeller must be able to import objects from different sources and to convert them into its internal representation. Then all operations on objects will be applied on this new representation. Objects must then be able to be saved and exported in different formats. The wish for precision and interactivity (during both modelling and visualisation processes) leads us to the following requirements:

- Objects are represented as volumes.
- Visualisation is interactive.
- Visualisation is accurate.
- The modelling process is accurate and interactive.
- Transformations can be applied to objects.
- Model and interface are adapted to the use of Boolean composition operators with free-form and point-by-point controlled transitions [7,8].

The goal of this paper is not to propose a complete solution, but as a first step, to present and to justify a kernel on which such a modeller can be based. In the following section, we present an overview of different volume modelling systems, and the structure they are based on. We then justify our choice of data structure in Section 3. Regular grids discretely storing the field values combined with a triquadratic approximation reconstruction are used to define volume objects. This association of a grid and an approximation/interpolation reconstruction allows the field to be defined by a  $C^1$  continuous real function and the surface to be directly visualised from its own equation. In Section 4, the triquadratic B-spline is described. We present the visualisation method and explain how interactive and high precision visualisation is obtained. This allows us to show that the visualisation is faithful to the object definition. Modelling tools and processes are presented in the fifth Section. We discuss the use of Boolean composition with free-form transition and we describe, as an example use of our data structure, an interactive interface to allow the user to precisely control the modelled shape. To illustrate the work presented in this paper, we conclude with some examples.

## 2. Related works

Different families of volume modellers exist. Some, like the *BlobTree*, are essentially based on function representation and their composition in a CSG tree [9]. To obtain interactivity, only bounded skeleton primitives are used [10,11,12] and the surface is polygonalised for rendering. Others are based on discrete structures like regular or adaptive grids, and again, the surface is polygonalised for rendering [13]. The potential field can be defined by either skeleton primitives [14] or level set methods [15,16]. All these models are targeted for a specific category of implicit surfaces. There are also hybrid systems which integrate as primitives most of the volume representations [17]. The primitives are manipulated using their own representation. Different techniques are then available to interactively render the surface: polygonalisation, points, ray-casting using trilinear interpolation, etc. The interactively visualised surface is always an approximation of the modelled object and depending on this approximate precision and on the sharpness of the shape, the error can grow, consequently reducing fidelity. ADFs [18] are adaptive octrees, which generally become deeper in proximity to the surface. Compared with regular grids, they still define a discrete representation of the potential field, but they avoid the limitation of the level of detail. They allow a faithful interactive rendering of the surface [19] but the value of the

potential at a random point of the working space has to be computed as a distance from the surface, or as a reconstructed value from the samples of the adaptive structure. These evaluations are time consuming and it makes ADF structures complicated to use for our interactive modelling purpose.

Grids are not only useful to store three-dimensional data. On each voxel of the grid, the data can be a scalar, the three components of the colour (four if the alpha channel is used), or any other multidimensional vector data. As shown by V.V. Savchenko et al. [20], this data structure is a template to unify objects defined by potential fields (discrete or not). Thus a wide variety of different input models, like output scanner 3D volumes, reconstructed field from scattered data [21,22,23,24], reconstructed distance field from isosurface [25] or modelled implicit objects [26] can be integrated and manipulated with a common representation. These volume objects can usually be defined by continuous real functions as  $f(x,y,z) \leq 0$  [20]. Conversion methods to convert one representation to another and details on voxel based object representation/manipulation are widely discussed in [17,27]. Different operations can then be applied on these objects from classical linear transformations and Boolean composition to more advanced algebraic operations [27], space mapping [28], sweeping by a moving solid [29] and Boolean compositions with soft [30,9] or free-form transitions [7,8].

### 3. Volume data structure

There is currently a great deal of research interest in finding methods to define, model or fit volume/surface data using real functions  $f: \mathbb{R}^3 \rightarrow \mathbb{R}$ . Amongst the reasons that motivate this interest, we draw attention to the wide variety of geometric operations that are available, the straightforward evaluation of the position of a point in regards to the surface and the possibility to choose and adapt the visualisation accuracy from the surface equation.

A volume described by a potential function  $f: \mathbb{R}^3 \rightarrow \mathbb{R}$  is defined as follows: The set of points  $p(x,y,z)$  for which  $f(x,y,z)=0$  defines the surface and the set of points  $p$  for which  $f(x,y,z) < 0^a$  defines the inside of the volume. This representation is implicit and the generated zero isosurface is commonly called an implicit surface. Primitives  $f_i$  thus defined are transformed or combined by composing them with operators  $\phi: \mathbb{R}^n \rightarrow \mathbb{R}$  where  $n$  is the number of primitives [28]. The result of an operation  $F = \phi(f_1, \dots, f_n)$  is a new primitive also defined as a volume. Operations on primitives are organised in hierarchical structures like CSG trees. Leaves are the primitives and nodes are operators. At each node, the new equation is a composition of the combined or transformed primitives. This implies that the equation complexity grows with the size of the tree. With even a small number of levels, the function evaluation becomes expensive in computing time and rendering the shape in interactive time becomes more difficult. Moreover, while a fast evaluation of combined functions  $f_i$  at the surface level is sufficient to interactively compose primitives with classical Boolean composition operators [31], Boolean operators with soft transition require, in addition, fast evaluation in the transition area [11,32,9] and Boolean composition with free-form transition requires fast evaluation in the whole modelling space because the stretch of the transition is not predictable before it has been created.

A formal visualisation of the potential function is not possible in most cases, the potential function must be sampled and, therefore, the reconstructed isosurface will always be an approximation to the initial function. To allow the computation time to depend only on the equation complexity of the applied operation or of the imported primitive, we store the

---

<sup>a</sup> The sign of the function defining the inside is chosen by convention. In this paper, we use  $f(x,y,z) < 0$  but the inverse convention where the inside of the volume is defined by  $f(x,y,z) > 0$  could also be chosen.

potential in a regular voxel grid at each level of the tree (leaves and nodes). The grid furnishes the sampled representation of the potential field. Thus, this method takes advantage of the fact that direct rendering of a voxel grid is possible [33], and the loss of time generated by the construction of an intermediate structure (usually a triangular mesh) to visualise the surface is avoided. ADFs or adaptive irregular grids are not used because we need to construct the grid as fast as possible without the problem of expensive computations while interpolating the values in the discrete structure to evaluate the field at a point  $p$  of  $\mathbb{R}^3$ . Furthermore, the use of regular grids allow us to integrate most of the volume primitives and operators in our model (as explained in Section 2).

However visualisation requires the inverse process that reconstructs a signal everywhere in the volume. This operation can be performed using the well-known convolution operator:

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(\tau) \times g(t - \tau) d\tau ,$$

where  $f$  is the sampled function and  $g$  the filter. Thus the choice of an efficient filter is essential for this application. Several kinds of 3D filters for volume visualisation have been studied in many ways in the past [34,35,36]. The most studied are: the linear B-spline filter, cubic B-spline filter based on BC-splines, gaussian filters and windowed sinc filters. Even though the linear B-spline is widely used in the volume rendering applications, studies have shown that it gives poor quality results. Better results can be obtained with cubic B-spline or windowed sinc filters. Indeed, the original function is better approximated and they allow smooth transitions due to a high degree of continuity. But their complexity prohibits any use within interactive software. Therefore the main problem of our approach is to find a filter allowing both interactive renderings and smooth reconstructions of the isosurfaces.

A useful solution can come from a triquadratic filter (trivariate quadratic B-spline) that allows interactive renderings [37] with a high quality reconstruction (Figure 1). It has been shown that quadratics are significantly faster to compute than cubics (due to their smaller support and their lower degree) while providing comparable quality in the resulting interpolation [38]. In adding this filter to the grid to define objects, the original implicit representation is reduced to a smooth  $C^1$  continuous representation. We thus combine the advantages of a sampled representation to store the potential values and a real continuous  $C^1$  representation to define and evaluate in a constant time the value of the potential and its first derivative everywhere in the modelling space, whatever the shape complexity. Thus, both the grid and the triquadratic approximation define our objects. Objects are no longer defined by their original data structure. This ensures that if the triquadratic B-spline reconstruction is used to accurately visualise the grid and if the rendering viewport is adapted to the grid's

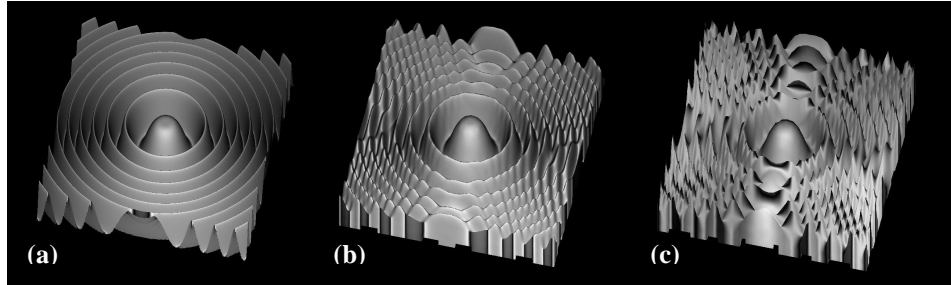


Fig. 1. Rendering of the Marschner & Lobb dataset ( $41^3$  samples) with an *ideal* filter (a), the triquadratic B-spline filter (b) and the usual trilinear B-spline filter (c). We can notice that trilinear B-spline interpolation gives poor results, especially with surface shading. This is essentially due to the erroneous shadings provided by the separation between the signal reconstruction and the gradient reconstruction (dark surfaces in c).

size, no unwanted noise or detail will appear if a more accurate visualisation technique is used. It is obvious that the thinness of the details in the modelled shape directly depends on the grid size, and the maximum size of the grid depends on the machine computation performance and the range of time considered as interactive.

#### 4. Triquadratic reconstruction

This section describes our visualisation method, which is based on ray casting. The quadratic filter and its reconstruction properties are first described. Then we show how high quality visualisation and interactive rendering are provided with the use of the triquadratic B-spline to smoothly reconstruct the potential values of a regular grid. We conclude this section with some optimisations that allow us to decrease computation time and reach the desired interactivity.

##### 4.1. Univariate quadratic B-spline

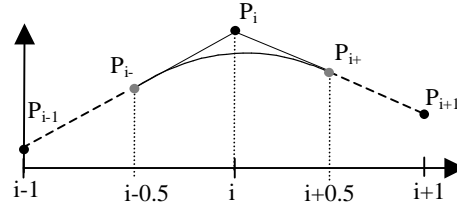


Fig. 2. Univariate quadratic B-spline.

Before introducing the triquadratic B-spline, we describe the filter in a one-dimensional case. Three neighbouring samples ( $P_{i-1}$ ,  $P_i$ ,  $P_{i+1}$ ) are needed (Figure 2) in order to reconstruct the one-dimensional signal within the interval  $[i-0.5, i+0.5]$ . Now the reconstruction is made in two steps. First, two points  $P_{i-}$  and  $P_{i+}$ , which are respectively the middles of the two segments  $[P_{i-1}, P_i]$  and  $[P_i, P_{i+1}]$ , are defined:

$$P_{i-} = \frac{P_{i-1} + P_i}{2} \quad P_{i+} = \frac{P_i + P_{i+1}}{2}.$$

Then a quadratic Bézier curve is defined from the three control points ( $P_{i-1}$ ,  $P_i$ ,  $P_{i+1}$ ). Thus the reconstruction equation, with  $t=0$  at  $i-0.5$  and  $t=1$  at  $i+0.5$ , can be written as :

$$f_{P_{i-1}, P_i, P_{i+1}}(t) = \left( \frac{P_{i-1} + P_{i+1}}{2} - P_i \right) t^2 + (P_i - P_{i-1})t + \frac{P_{i-1} + P_i}{2}$$

or in a matrix representation :

$$f_{P_{i-1}, P_i, P_{i+1}}(t) = \begin{bmatrix} t^2 & t & 1 \end{bmatrix} \times \begin{bmatrix} 0.5 & -1 & 0.5 \\ -1 & 1 & 0 \\ 0 & 0.5 & 0.5 \end{bmatrix} \times \begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \end{bmatrix}$$

The  $C^0$  and  $C^1$  continuities are obvious. While B-spline functions are internally  $C^\infty$  continuous, the continuity across the knots, between segments, is only guaranteed to be  $C^1$  by construction.

Another interesting property is that the reconstructed signal does not pass through the sampled points. Local maxima are never reached and the high frequencies of the signal are smoothed.

The final property of note is the first derivative. It is analytically given by :

$$\begin{aligned} f'_{P_{i-1}, P_i, P_{i+1}} &= (1-t)(P_i - P_{i-1}) + t(P_{i+1} - P_i) \\ &= (1-t)G_{i-} + tG_{i+} \end{aligned}$$

with:

$$G_{i-} = P_i - P_{i-1} \quad G_{i+} = P_{i+1} - P_i.$$

The derivative function is in fact a linear interpolation of two middle difference gradients. In a usual volume visualisation, the gradient is estimated with a (tri)linear approximation of central differences gradients computed at the sampled points using the formula:

$$G^{CDIF}(i) = \frac{P_{i+1} - P_{i-1}}{2} = \frac{G_{i-} + G_{i+}}{2} = G^{MDIF}(i).$$

The equation above shows that the gradient estimated with the usual central difference gradient is less accurate than the gradient estimation provided by the quadratic B-spline. Indeed, the central difference at the sampled point is the mean value of two middle difference gradients  $G_{i-}$  and  $G_{i+}$ . Thus the interpolation of the mean values ( $M_i = 1/2 \cdot G_{i-} + 1/2 \cdot G_{i+}$ ) used in the central difference estimation leads to a loss of information in the gradient reconstruction, with regard to the use of the direct interpolation of middle differences  $G_i$ . However the quadratic reconstruction is more sensitive to the data noise that is fortunately usually not present in the case of three-dimensional signals generated from equations.

#### 4.2. Trivariate quadratic B-spline

The extension to three-dimensional volumes is done using tensor products and the previous results are also true while extended to this case. 27 volume samples ( $3 \times 3 \times 3$ ) are now required to reconstruct the  $C^1$  continuous signal within a parallelepiped aligned on the central sample (Figure 3). Like the usual trilinear interpolation, applying the univariate B-spline on the three axes performs the reconstruction. First 9 passes on the x axis are done providing 9 intermediate samples that undergo 3 passes on the y axis. It leads to 3 intermediate results that finally are used in one pass on the z axis.

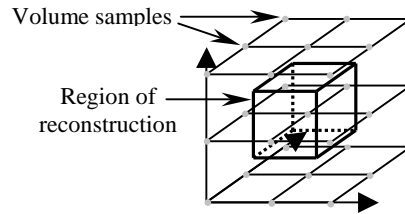


Fig. 3. Three-dimensional reconstruction.

The final reconstructed potential function  $h$  of the signal within the parallelepiped at a location  $(x, y, z)$  can be written as:

$$h_{P_{000} \dots P_{222}}(x, y, z) = \sum_{i,j,k=0}^2 S_{ijk} \cdot x^i y^j z^k. \quad (1)$$

Here the 27  $S_{ijk}$  coefficients are computed from the 27 grid potentials ( $P_{ijk}$ ) by using a  $27 \times 27$ -transformation matrix representative of the triquadratic B-spline. Fortunately, a lot of matrix coefficients are null, reducing the computations of the  $S_{ijk}$  coefficients.

### 4.3. Fast Visualisation

By now the visualisation of the implicit surface crossing this parallelepiped reconstruction can be reduced to the visualisation of a  $C^1$  continuous surface defined by the equation  $h(x,y,z)=0$ , with  $x$ ,  $y$  and  $z$  in  $[0..1]$ . However it is not obvious how to find the projection of this surface on the image plane at interactive rates. The proposed solution allows fast rendering while preserving a high quality of visualisation. It is based on the fact that every ray going through this parallelepiped can be written in a parametric form:

$$x = x_0 + t.(x_1 - x_0), \quad y = y_0 + t.(y_1 - y_0), \quad z = z_0 + t.(z_1 - z_0) \quad (2)$$

By substituting the coordinates in the previous equation, the reconstruction along the ray can be written as a sixth degree polynomial:

$$h(t) = \sum_{i=0}^6 C_i t^i \quad 0 \leq t \leq 1. \quad (3)$$

This polynomial should be solved in order to find the intersection with the isosurface. However there is no algorithm that allows the accurate computation of such a solution in a very short time. The Sturm theorem could be used in order to efficiently approximate the roots, but its implementation is currently too expensive to be used in an interactive visualisation tool. Thus a less elegant but more efficient solution which samples many values (fixed to 16 here) along the ray is used. Once an intersection has been detected (i.e.  $h(t_{i-1}) < 0 < h(t_i)$ ), the final value of  $t$  is linearly interpolated from the last two samples. This allows us to obtain an accurate approximation of the intersection between the ray and the isosurface. Finally, the shading process requires the surface normal which is obtained by computing from the intersection point using the following equation:

$$\nabla h_{p000..P222}(x,y,z) = \begin{pmatrix} d \sum_{i,j,k=0}^2 S_{ijk} \cdot x^i y^j z^k / dx \\ d \sum_{i,j,k=0}^2 S_{ijk} \cdot x^i y^j z^k / dy \\ d \sum_{i,j,k=0}^2 S_{ijk} \cdot x^i y^j z^k / dz \end{pmatrix}.$$

The accurately raytraced surface is the zero isosurface reconstructed from the grid by the triquadratic B-spline. This solves one of our requirements: The visualised shape is a faithful representation of the modelled one.

### 4.4. Optimisations

Because many computations are required by the quadratic reconstruction, the visualisation process must be highly optimised. Our optimised algorithm allows interactive rendering times for the visualisation of  $256^3$  voxels in a  $512^2$  viewport using a 1.0 GHz AMD Athlon processor. Furthermore this algorithm allows interactive settings of the isosurface threshold. A complete description is given in [37], hence we will just describe the main optimisations.

The fast computation of coefficients  $C_i$  of the polynomial equation (Eq. (3)) represents the main requirement to perform the interactive rendering. They depend on both the  $S_{ijk}$  coefficients (Eq. (1)) and the ray parameters (Eq. (2)). A naïve raycasting algorithm would compute the  $S_{ijk}$  values for every voxel crossed by the ray. A more suitable approach is to use a sorted representation of the object where voxels are projected in a front-to-back order. Then the  $S_{ijk}$  values are computed only once for all the rays going through the voxel. Their

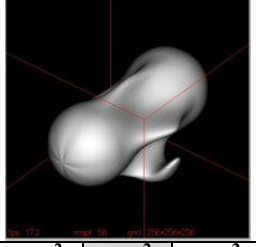
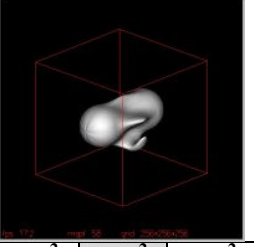
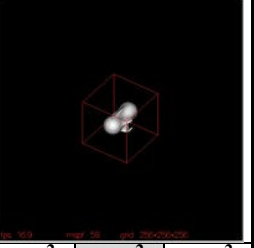
computation requires 316 additions and 80 multiplications per voxel. The evaluation of the  $C_i$  coefficients from the ray parameters and the  $S_{ijk}$  values requires an addition of 724 multiplications and 108 additions per ray. But most of these multiplications depend only on the ray position, and by using a large set of precomputed rays (instead of the real rays), the number of multiplications is reduced to 108 per ray.

A Min-Max octree is also used in order to compute only the voxels of interest. The octree is run in a front-to-back order and the nodes that do not enclose the min-max values are skipped. The leaf nodes contain the min and max values of the 27 samples needed for the reconstruction. These samples also represent an absolute boundary of the reconstructed signal within the voxel.

Trees are often used in computer graphics. They increase raytracers performances while finding the intersection between the ray and the sampled surface in  $O(\text{Log}(n))$  steps [39]. Our implementation of an octree provides the same improvement. The use of the Min-Max octree as an additional structure may seem to be a contradiction within our approach, but it allows our rendering technique to be interactive.

As we can see in Table 1, rendering time is not anymore principally dependent on the volume size, but on the zoom on the surface and on the image size. Notably, the visualization speed mainly depends on the number of rays intersecting the surface. A compromise allowing both fast construction of the Min-Max octree (in 1.6 seconds) and interactive rendering can be proposed, as shown with the grey cells in Table 1 (grids of  $128^3$  voxels and a rendering window of  $512^2$  pixels). For our modelling purpose, we use this configuration. Less interactive but more accurate rendering is possible in a larger image to temporally check small details on the surface if necessary.

Table 1. Min-Max octree computation times (in sec) for different sizes of the grid (in voxels) and corresponding rendering times (in sec/frame) for different zooms on the surface and different sizes of the rendering window (in pixels). The grey cells show a compromise allowing both fast computation of the Min-Max octree and interactive rendering. These times are obtained using a 1.0 GHz AMD Athlon processor with 512 Mbytes of DDR memory.

Grid's size in number of voxels	Min-Max octree in sec	Zoom, window's size in pixels and rendering in seconds per frame								
										
		$256^2$	$512^2$	$768^2$	$256^2$	$512^2$	$768^2$	$256^2$	$512^2$	$768^2$
$64^3$	0.17	0.29	1.25	3.33	0.09	0.33	0.71	0.04	0.14	0.32
$128^3$	1.6	0.25	0.83	1.67	0.10	0.28	0.59	0.04	0.13	0.27
$256^3$	14	0.30	0.83	1.67	0.12	0.33	0.59	0.06	0.14	0.28

## 5. Potential function manipulation

The data structure and its visualisation method are now well defined. Note that interactivity during the modelling process depends on the size of the grid and the computation complexity of the operators whereas interactive visualisation essentially depends on the size of the rendering window, whatever the grid.

The computer used to test the modeller has a 1.8 GHz AMD Athlon XP processor and 1.0 Gbytes of SDR memory at 133 MHz. As suggested in Section 4.4, the standard grid is



composed of  $128^3$  float values of storage eight Mbytes each and the visualisation is done in a  $512^2$  viewport (larger viewports are available for less interactive but finer rendering). Times given in this section refer to this configuration.

In order to model complex objects, primitives have to be combined or transformed. In this section we show how operators are applied and we discuss the limitations caused by the bounded representation of the potential field in a grid. We also recall how free-form accurate blending is performed and controlled in order to introduce requirements for a suitable interface. This allows us to show how our data structure can be used in an interactive interface to precisely model volume objects. The application is an example given to illustrate our model's capabilities.

### 5.1. Operators

Operators  $\phi^{\text{comp}}$  defined<sup>b</sup> as a function of primitives  $f_i$  are directly applied on the grids representing the primitives. The potential field resulting from the operation is stored in a new grid and the modelled object is defined by its approximation by the triquadratic filter. This approach allows us to modify or compose objects in a time independent of the shape complexity.

However, grids are large data structures: Their storage requires a lot of memory. To define fine objects in a large enough modelling space, we use  $128^3$  float values grids (smaller or larger grids can be used depending on the processor speed). It is inconceivable to store all the grids representing each node or leaf of the tree: Even if a lot of memory is available, objects defined by many operations are liable to overflow the maximum available storage space. For this reason, it is unsuitable to modify the object by acting on primitives from which it is defined. These primitives being nodes or leaves situated at lower level of the tree. Actually, if a node or a leaf of the current object is modified, all the computation will have to be done from the leaves to the current node. This complete evaluation of the tree can be a very long process.

For this reason, it is preferable to directly apply operators on the current object to be able to modify it in an interactive time (if operators allowing the desired modification are available). This is important to take into account when modelling. Indeed, to limit backtracks in the tree and long computations, the user has to carefully and accurately model his object operation by operation.

Space mapping operator  $\phi^{\text{map}}$  transforms the space in which the potential field is defined. While deformations like tapering, twisting or bending can be applied on volume objects when their equation is known [40,9], it becomes delicate when the volume is defined in a bounded space without the knowledge of the potential values outside the boundaries and when the operator applies itself on the whole modelling space. Even for basic operators like translation or rotation, how can we compute all the values of the grid representing a volume after its translation where at least metric, and maybe curvature, of the potential field have to be conserved?

We do not give a solution to this problem, which is illustrated in two dimensions in figure 4. Specific research will have to be done to explore this question and to propose suitable solutions. An introduction to this problem can be found in [41] which catalogues a variety of possible solutions. Approaches like level set methods [42] may also be worth investigating. Thus, only bounded space mapping operators [43] having their bounded box included in the modelling space are able to be directly used on our volume objects. A solution to avoid this

---

<sup>b</sup> Offsetting, Boolean composition with or without soft transitions and metamorphoses are elements of this operator set.

problem is to use only bounded primitives [29,44,9,14] sampled in an infinite grid [14]. The values outside the boundaries are a known constant scalar and unknown areas are avoided. Moreover, even if the grid is infinite, the visualisation could remain interactive (rendering speed depends mostly on the viewport size). Volumes defined by bounded potential fields can be used as input objects, but mechanisms to restrict all primitives to this representation are out of the scope of this paper.

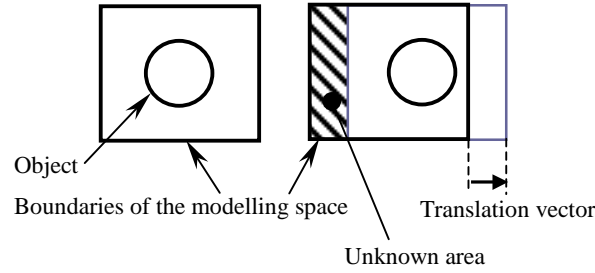


Fig. 4. It is an open question how to compute the potential field values in the unknown area while conserving regular variation of the potential field.

Because each primitive is stored in its own grid, the use of n-ary operators leads us to store at least the n grids, which is a memory greedy operation. To minimise the number of stored grids, we only use unary and binary operators. As shown in [29], this does not consequently restrict the variety of available operations. Moreover, because it is defined as the sum of potential functions having specific field variations, operators like the classical n-ary blending operator used for “blobs” can be decomposed and applied as a succession of binary blending operators [14].

Boolean composition operators with soft transition are considered as powerful and useful [10,45,20,9] for implicit modelling, therefore, one of our main preoccupations is to propose an accurate and interactive tool to realise them. As shown in [7,8], point-by-point control of the blend is possible and the classical smooth and regular curved or inflated transition can be extended to a free-form transition.

Our free-form composition operator [8] is defined by a two-dimensional potential function  $g: \mathbb{R}^2 \rightarrow \mathbb{R}$  as follows:

$$\phi^{comp}(f_1, f_2) = g(f_1, f_2).$$

After composition, the resulting shape is defined by  $g(f_1, f_2) = 0$  and the applied operator depends on the form of the curve  $g(x, y) = 0$ . This curve is controlled by control points and the classical Boolean composition operators with soft transitions (union, intersection and difference) are generalized and extended to a unique free-form operator that combines, sculpts or creates implicit primitives (Figure 5).

Functions  $g$  used to create the curve preserve the metric of the combined primitives outside the part of the field affected by the blend and keep regular variations inside, which makes this approach very relevant for composition of volume objects [46]. Furthermore, control points can be directly selected from a Euclidean space. In an adapted interface, this allows the user to accurately and easily define and control the form of the transition from its modelling space. Such an interface is presented in the following section and a complete description of this model is given in [7,8].

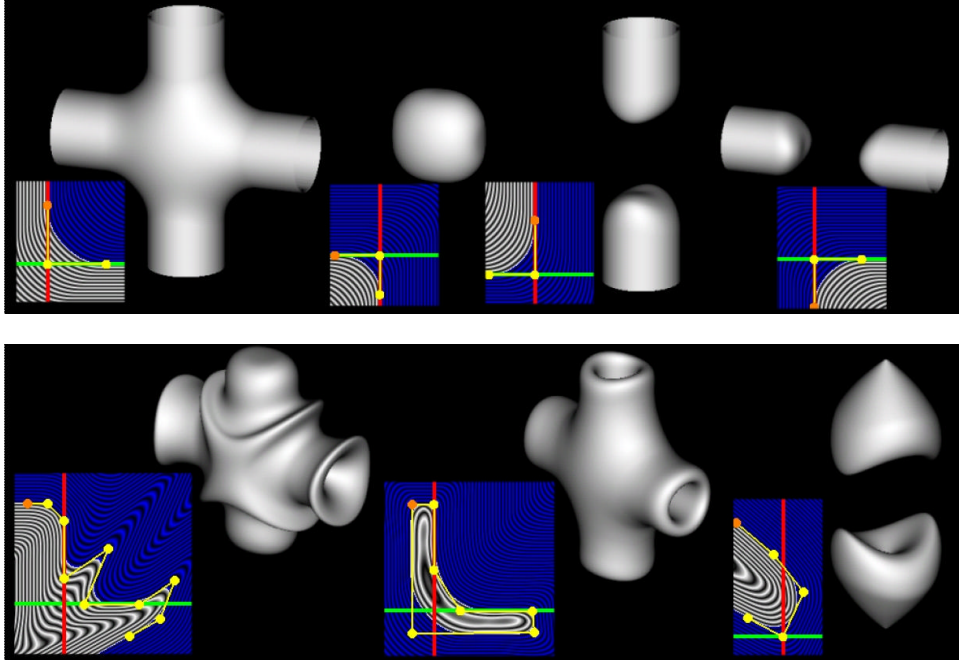


Fig. 5. Different free-form operators applied on two orthogonal cylinders. The top row shows classical union, intersection and the two difference operators with soft transitions. The following row illustrates more advanced possibilities offered by our free-form operator. In the operator picture, the vertical axis represents  $x=0$  (which corresponds to the surface  $f_1=0$  when  $g$  is composed with  $f_1$  and  $f_2$ ) and the horizontal axis represents  $y=0$  (which corresponds to the surface  $f_2=0$  when  $g$  is composed with  $f_1$  and  $f_2$ ).

## 5.2. Application in a modelling interface

During the object rendering process, both frame-buffer and Z-buffer are computed from the grid and the triquadratic reconstruction. Once these buffers are initialised, the standard three-dimensional object visualisation library OpenGL is used to build interface components in the scene, minimally affecting the interactivity of the visualisation. Indeed, if specific graphics hardware is used, the processor is mostly relieved of interface component manipulation and rendering tasks (other three-dimensional object visualisation libraries accelerated by graphics hardware could be used with the same efficiency).

To apply binary operators on primitives, at least three grids are necessary: One for each primitive and one for the resulting object. In our interface, we use a supplementary grid to temporarily store a volume. Thus, a modelled object can be stored while another is created. The temporary object can be used later as a primitive to be combined with the current one.

At this time, our rendering method only allows the visualisation of a single grid. A new primitive is positioned with regards to the current object, in visualising them in the same grid using the classical union operator (realised with the min function [31]). It takes 0.10 seconds to compute the grid. The classical intersection and difference operators are also available and obviously, the rendered object resulting from one of these compositions can be selected as a new primitive. The sharp transition normally created between the combined objects is, in grids reconstructed by a triquadratic B-spline, a small oscillating transition the size of which depends on the grid resolution (Figure 6). We point out that the same type of oscillations can be found around ridges or at the level of very thin surfaces (Figure 7).

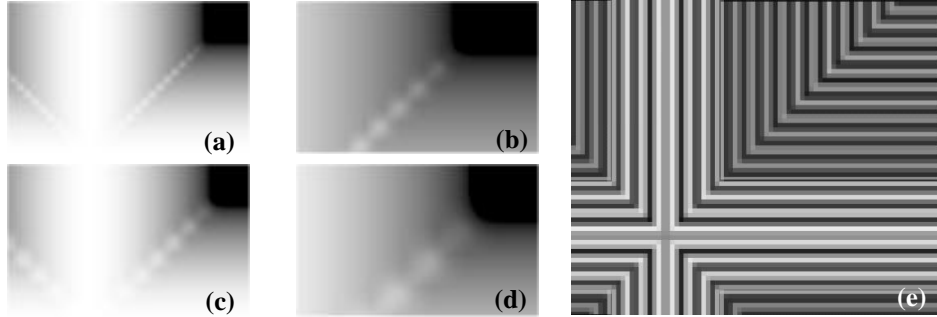


Fig. 6. Sharp transition between two cylinders obtained using Ricci's Min-Max union Boolean operator [31].

Grid	Zoom out	Zoom in
$128^3$	(a)	(b)
$64^3$	(c)	(d)

Figures (a),(b),(c) and (d) show the oscillations along the sharp transition reconstructed from the grid and Figure (e) shows the variations of the potential field in a 2D section.

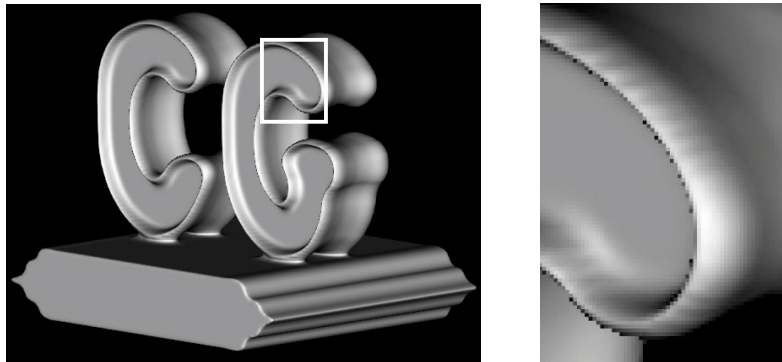


Fig. 7. On the left, an object modelled with our interface. The zoom on the area delimited by the white rectangle is shown on the right. We can clearly see the oscillations on the boundary of thin surfaces.

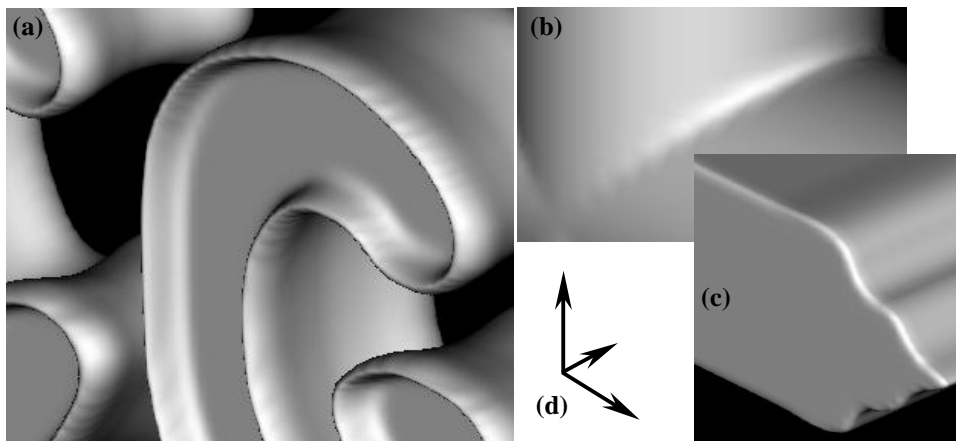


Fig. 8. Illustration of oscillations generated by the three-directions quadratic Box-spline reconstruction. Figures (a), (b) and (c) show respectively the reconstruction of thin details, sharp transitions and ridges. Figure (d) shows the Box-spline directions. As we can see, no oscillation appears along these directions.

It would be useful if these undesired oscillations could be avoided. The uniform triquadratic B-spline is also a trivariate (by tensor product) quadratic Box-spline [49] and this phenomenon is a known problem of Box-spline reconstruction. However, around ridges or thin details, it has smooth reconstruction properties if the feature is along one of the Box-spline directions (Figure 8). In our case, grid axes are Box-spline directions. This oscillation phenomenon is similar to the lateral artifact in subdivision surfaces [47] and a solution could be to use a local reconstruction based on Box-splines having more different directions. More research and experiments are needed in this area.

It is also possible to produce a sharp transition with a  $C^1$  discontinuity on the surface while maintaining a  $C^1$  continuous potential field everywhere else in the potential field [30,8]. At the grid level, it generates smoother samples but as shown in Figure 9, there is no improvement of the oscillation effects (with respect to Figure 6).

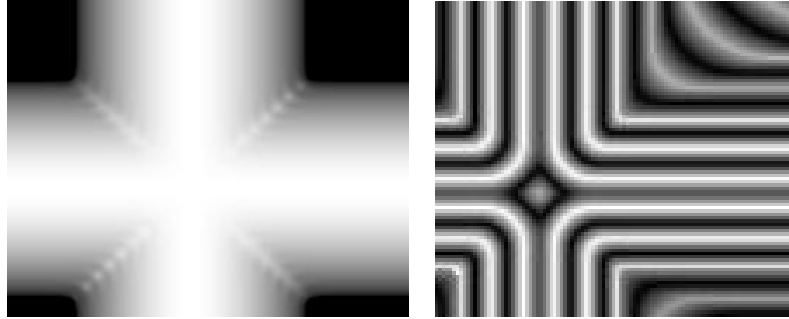


Fig. 9. Sharp transition between two cylinders obtained using the union Boolean operator proposed by Barthe et al. in [8]. This operator provides a potential field that is  $C^1$  continuous when both functions  $f_1$  and  $f_2$  are not equal to *zero*. On the right, the 3D visualisation from a grid composed of  $128^3$  voxels and on the left, a 2D section of the potential field.

As described in section 5.1, when free-form transitions are used in Boolean operators, the form of the blend can be controlled by points selected in the modelling space. Depending on the operator complexity, the computation time of the grid varies following Table 2.

Table 2. Time of computation of a grid while using a Boolean operator with a free-form soft transition depending on the number of control points used to defined its form.

Number of control points	Computation of the grid	Computation of a plane section of $256^2$ pixels, crossing the grid
3	0.27 sec	0.14 sec
4	0.40 sec	0.15 sec
6	0.60 sec	0.16 sec
8	0.88 sec	0.17 sec
12	1.37 sec	0.20 sec
20	2.40 sec	0.23 sec

To allow the maximum interactivity while selecting the control points, we visualise the variation of the potential field in a plane section crossing the grid. The plane is first positioned in the modelling space, with respect to the combined primitives (Figure 10a). Then the section is rendered in a viewport of  $512^2$  pixels (Figure 10b) in approximately 0.08 seconds. The triquadratic reconstruction is also used here to faithfully represent the field values of the volume. Different colours are used to identify the different relevant areas of the

volumes, i.e. inside/inside respectively object1/object2, inside/outside, outside/inside and outside/outside. The transition is selected with the mouse in this plane section (Figures 10c and 10e) and the result is interactively visualised with a transparency effect. When desired, the resulting surface can be visualised in the three-dimensional modelling space (Figures 10d and 10f). The delay to visualise the first 3D picture of the resulting object is equal to the sum of computation times of both grid and Min-Max octree. In general, this time varies between 1.5 and 2.5 seconds (on the computer used for our modeller, the computation of the Min-Max structure is reduced to 1.0 second). Once the grid and the Min-Max octree are built, the object is interactively visualised with several frames per second.

Two-dimensional plane section representing the potential field values can be used to increase the interactivity (Table 2) and control the potential field variations while applying any composition operators with soft transitions. For bounded operators, only a small region of the section need to be visualised and real time control of the form of the transition can be expected.

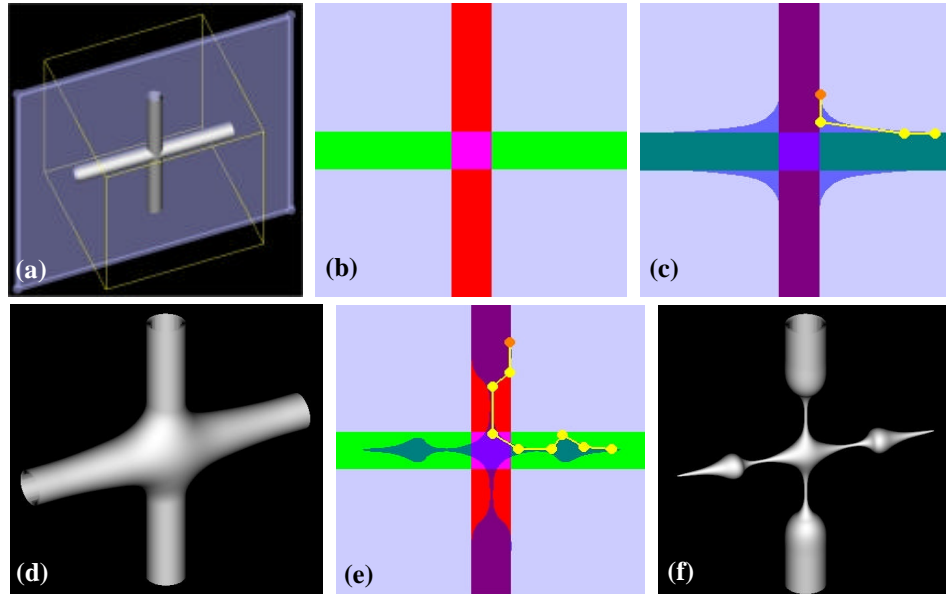


Fig. 10. The plane section is positioned in the modelling space with respect to the two combined cylinders (a), and the section of the potential field is visualised in a two-dimensional viewport (b), in which the transition is interactively selected and the result is visualised using a transparency effect (c). The corresponding object is accurately visualised in the three-dimensional modelling space (d). More complicated operations can be achieved when different forms of the curve  $g$  are selected in the two-dimensional viewport (e) and our rendering method provides an accurate and faithful three-dimensional visualisation (f).

## 6. Conclusion and future work

The combination of a discrete representation of potential fields with an adequate reconstruction generates a continuous approximation of the field in a bounded area.

A regular grid allows a fast storage of the potential values and the triquadratic approximation B-spline generates a  $C^1$  continuous function. Our rendering algorithm provides a fast, accurate and smooth visualisation of the field isosurfaces reconstructed with the triquadratic B-spline. Thus, the definition of volume objects using the association of these two components allows us to obtain a faithful visualisation and provide interactive modelling tools.

These properties of our volume objects have been successfully illustrated with the application of advanced Boolean composition operators with free-form transitions in an interactive modelling software.

For these reasons, the proposed modelling kernel fulfils most of our requirements. Further studies can now be done to improve and complete this approach. In particular, research needs to be undertaken in the following areas:

- How to apply space mapping operators.
- Potential fields are stored with float values (coded with 32 bits). As suggested by Adzhiev and al. [17], the values can be mapped in short integer and coded with 16 bits. Our rendering algorithm already supports such a grid and the 16 bits left could then be used to store colour components.
- Different reconstructions could be applied on the grid values to try to avoid aliasing in the approximation of sharp edges.
- A local update of the data structure and of the rendered image would increase the interactivity while applying local operators.

## 7. Results

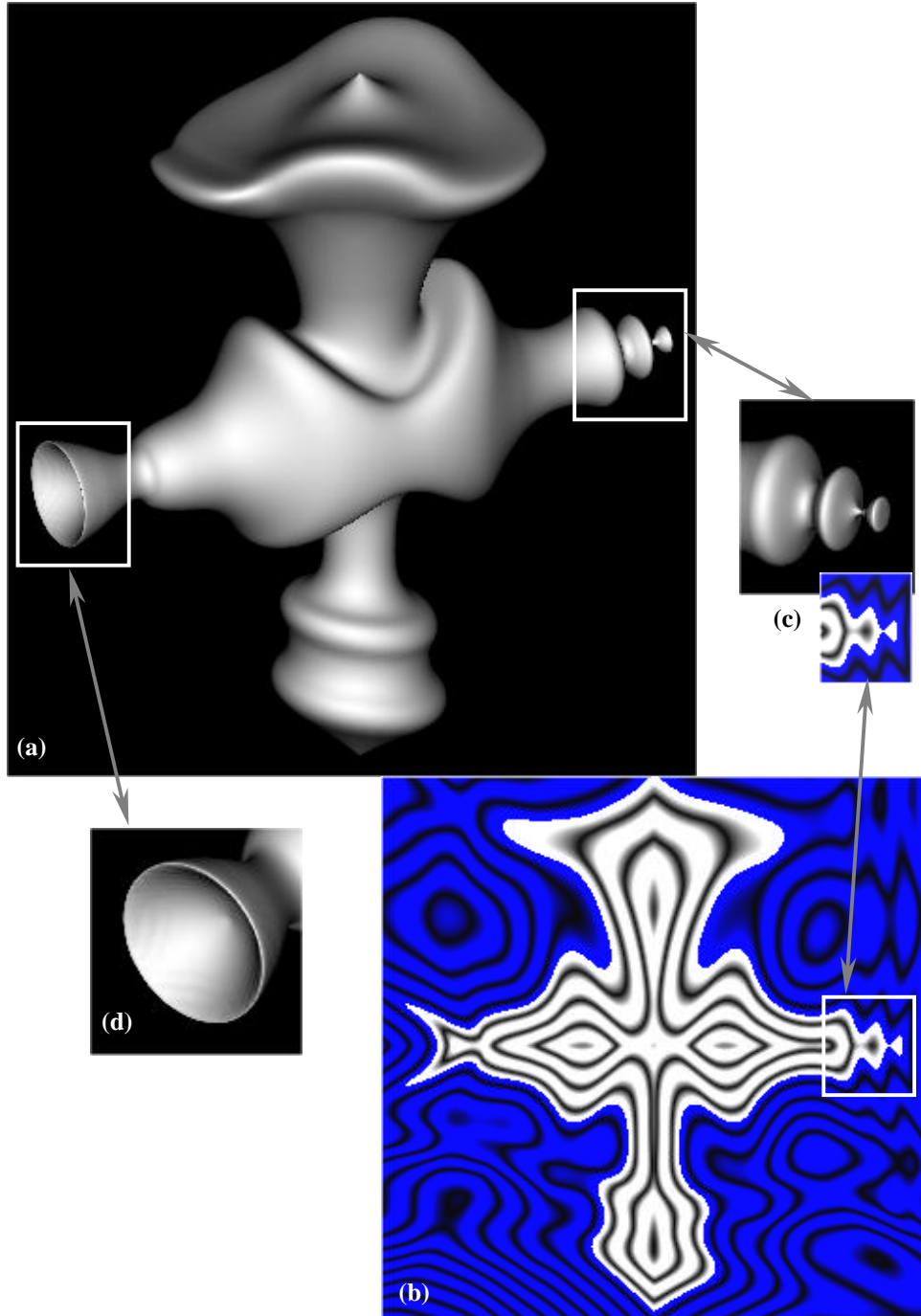


Fig. 10. Illustration of the visualisation of a smooth free-form isosurface in the usual  $512^2$  modelling rendering viewport (a) and of its faithful representation of the potential field in a plane section (b). Accurate representation of thin details is shown in (c) and (d).



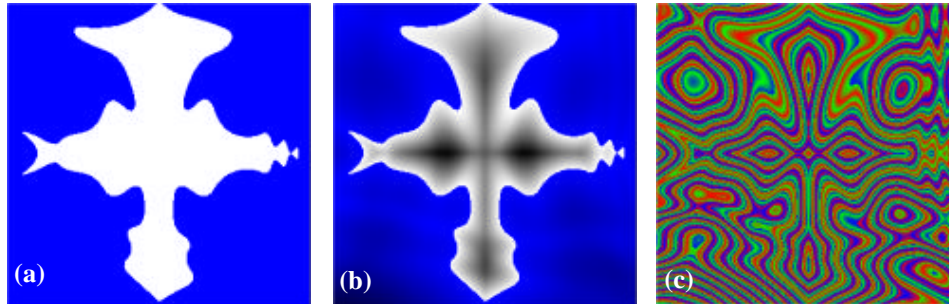


Fig. 11. Different visualisations of the plane section of the potential field: (a) uniform colour, (b) variation of the colour with the potential field value and (c) tricolour isosurfaces.

### Acknowledgements

This work was partially funded by E.U., contract HPRN-CT-1999-00117.

### References

1. W. Lorensen and H. Cline, Marching cubes: A high-resolution 3D surface construction algorithm, *SIGGRAPH'87* (1987) 163-169.
2. Balazs Csébfalvi, Andreas König, Eduard Gröller, Fast Surface Rendering of Volumetric Data, *WSCG'2000*, short paper (2000).
3. P. Lacroute and M. Levoy, Fast volume rendering using a shear-warp factorization of the viewing transformation, *SIGGRAPH'94* (1994) 451-458.
4. G. Knittel, "The Ultravis System", *IEEE/ACM SIGGRAPH Volume visualization and graphics symposium 2000* (October 2000) 71-78.
5. K. Engel, M. Krauss and T. Ertl, High-quality pre-integrated volume rendering using hardware-accelerated pixel shading" *Proc. of Eurographics/Siggraph Graphics Hardware Workshop 2001* (2001).
6. L.P. Kobbelt, M. Botsch, U. Schwanerke and H-P Seider, Feature sensitive surface from volume data, *Computer Graphics Proceedings, Annual Conference Series* (August 2001) 57-66.
7. L. Barthe, V. Gaildrat and R. Caubet, Extrusion of 1D profiles: Theory and first application, *International Journal of Shape Modeling* **7**(2) (December 2001) 179-198.
8. L. Barthe, N.A. Dodgson, M.A. Sabin, B. Wyvill and V. Gaildrat, Different applications of two-dimensional potential fields for volume modeling, *Technical Report, University of Cambridge*, UCAM-CL-TR-541, ISSN 1476-2986 (August 2002).
9. B. Wyvill, A. Guy and E. Galin, Extending the CSG tree: Warping, blending and Boolean operations in an implicit surface modeling system, *Computer Graphics Forum* **18**(2) (June 1999) 149-158.
10. J.F. Blinn, A generalization of algebraic surface drawing, *ACM Transaction on Graphics* **1**(3) (July 1982) 235-256.
11. J. Bloomenthal and B. Wyvill, Interactive techniques for implicit modelling, *Computer Graphics (proc. of SIGGRAPH'90)* **24**(2) (1990) 109-116.
12. B. Crespin and C. Schlick, Implicit sweep objects, *Eurographics'96* **15**(3) (1996) 165-174.

13. A. Raviv and G. Elber, Three dimensional freeform sculpting via zero sets of scalar trivariate functions, *Computer-Aided Design* 32(8-9) (August 2000) 513-526.
14. E. Ferley, M.P. Cani and J.D. Gascuel, Resolution adaptive volumetric sculpting, *Graphical Models* (March 2002).
15. J.A. Bærentzen and N.J. Christensen, Volume sculpting using the level-set method, *Proc. of Shape Modeling International'02* (May 2002) 175-182.
16. K. Museth, D.E. Breen, R.T. Whitaker and A.H. Barr, Level set surface editing operators, *ACM Transaction on Graphics (Proc. of SIGGRAPH)* 21(3) (July 2002) 330-338.
17. V. Adzhiev, M. Kazakov, A. Pasko and V. Savchenko, Hybrid system architecture for volume modelling, *Computer & Graphics* 24(1) (2000) 67-78.
18. S.F. Frisken, R.N. Perry, A. Rockwood and T.R. Jones, Adaptively sampled distance fields: A general representation of shape for computer graphics, *Computer Graphics Proceedings, Annual Conference Series* (July 2000).
19. R.N. Perry and S.F. Frisken, Kizamu: A system for sculpting digital characters, *Computer Graphics Proceedings, Annual Conference Series* (August 2001) 47-56.
20. V.V. Savchenko, A.A. Pasko, A.I. Sourin and T.L. Kunii, Volume Modelling: Representations and advanced operations, *Proc. of Computer Graphics International'98* (1998) 4-13.
21. G. Turk and J.F O'Brian, Shape transformation using variational implicit surfaces, *Computer Graphics Proceedings, Annual Conference Series* (1999).
22. B.S. Morse, T.S. Yoo, P. Rheingans, D.T. Chen and K.R. Subramanian, Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions, *Proc. of Shape Modeling International'01* (May 2001) 89-98.
23. J.C. Carr, R.K. Beatson, J.B. Cherrie, T.J. Mitchell, W.R. Fright, B.C. McCallum and T.R. Evans, Reconstruction and representation of 3D objects with radial basis functions, *Computer Graphics Proceedings, Annual Conference Series* (August 2001) 67-76.
24. H.K. Zhao, S. Osher and R. Fedkiw, Fast surface reconstruction using the level set method, *IEEE Workshop on variational and level set methods*, Vancouver, Canada (2001) 194-202.
25. M. Jones and R. Satherley, Shape representation using space filled sub-voxel distance fields, *Proc. of Shape Modeling International'01* (May 2001) 316-325.
26. J. Bloomenthal, C. Bajaj, J. Blinn, M.P. Cani-Gascuel, A. Rockwood, B. Wyvill and G. Wyvill, *Introduction to implicit surfaces*, Morgan Kaufmann Publishers, 1997.
27. M. Chen and J.V. Tucker, Constructive volume geometry, *Computer Graphics forum* 19(4) (2000) 281-293.
28. V. Savchenko and A. Pasko, Transformation of functionally defined shapes by extended space mapping, *The Visual Computer* 14(5/6) (1998) 257-270.
29. A. Sourin and A. Pasko, Function representation for sweeping by a moving solid, *IEEE Transaction on Visualization and Computer Graphics* 2(1) (1996) 11-18.
30. A. Pasko, V. Adzhiev, A. Sourin and V. Savchenko, Function representation in geometric modeling: Concepts, implementation and applications, *The visual Computer* 8(2) (1995) 429-446.
31. A. Ricci, A constructive geometry for computer graphics, *The Computer Journal* 16(2) (1973) 157-160.
32. M.P. Cani-Gascuel and M. Desbrun, Animation of deformable models using implicit surfaces, *IEEE Transaction on Visualisation and Computer Graphics* 3(1) (March 1997).
33. A. Kaufman, D. Cohen and R. Yagel, Volume graphics, *IEEE Computer* 26(7) (July 1993) 51-64.
34. S.R. Marschner and R.J. Lobb, An evaluation of reconstruction filters for volume rendering, *Proc. of visualization'94* (October 1994) 100-107.

35. T. Möller, R. Machiraju, K. Mueller and R. Yagel, Evaluation and design of filters using a Taylor series expansion, *IEEE Transaction on visualization and computer graphics* **3**(2) (April 1997) 184-190.
36. T. Theubl, H. Hauser and E. Gröller, Mastering windows: Improving reconstruction, *Proc. of IEEE/ACM SIGGRAPH Volume visualization and graphics symposium 2000* (October 2000) 101-108.
37. B. Mora, J.P. Jessel and R. Caubet, Visualization of isosurfaces with parametric cubes, *Eurographics'01 Proc.* **20** (3) (September 2001) 377-384.
38. N.A. Dodgson, Quadratic interpolation in image resampling, *IEEE Transaction on Image Processing* **6**(9) (September 1997) 1322-1326.
39. I. Wald, P. Slusallek, C. Benthin and M. Wagner, Interactive rendering with coherent raytracing, *Eurographics'01 Proc.* **20**(3) (2001) 377-384.
40. A.H. Barr, Global and local deformations of solid primitives, *Computer Graphics* **18**(3) (1984) 21-30.
41. N.A. Dodgson, Image Resampling, *Technical Report no. 261*, University of Cambridge Computer Laboratory (1992).
42. J.A. Sethian, Level set methods and fast marching methods, *Cambridge University Press*, second edition, 1999.
43. B. Crespin, Implicit free-form deformations, *Proc. of Implicit Surfaces'99* (1999).
44. C. Grimm, Implicit generalized cylinders using profile curves, *Proc. of Implicit Surfaces'99* (1999) 33-41.
45. C. Hoffman and J. Hopcroft, Automatic surface generation in computer aided design, *The Visual Computer* **1** (1985) 92-100.
46. A.P. Rockwood, The displacement method for implicit blending surfaces in solid models, *ACM Transaction on Graphics* **8**(4) (1989) 279-297.
47. M.A. Sabin and L. Barthe, Artifacts in recursive subdivision surfaces, submitted for publications.
48. M.A. Sabin, Subdivision of Box-splines, *Tutorials on Multiresolution in Geometric Modeling*, A. Iske, E. Quak and M. Floater (ed.), Springer (2002).
49. H. Prautzsch, W. Boehm and M. Paluszny, Box splines, *Bezier and B-spline techniques*, Mathematics+Visualization, G. Farin, H.C. Hege, D. Hoffman, C.R. Johnson and K. Polthier (ed.), Springer (2002) 239-258.