Triquadratic Reconstruction for Interactive Modelling of Potential Fields

Loïc Barthe

Rainbow Group
Computer Laboratory
University of Cambridge (UK)
lb282@cl.cam.ac.uk

Benjamin Mora

Department of Computer Science
University of Toulouse (France)
mora@irit.fr

Neil Dodgson

Rainbow Group
Computer Laboratory
University of Cambridge (UK)
nad@cl.cam.ac.uk

Malcolm Sabin

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

Abstract

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 filter 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 of application of our data structure, how advanced Boolean operators realised with soft or "functionally controlled" transitions are performed under the influence of an interactive modelling tool.

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 the 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. Three-dimensional data are usually stored in regular grids. 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. [7], 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 [8,9,10], reconstructed distance field from isosurface [11] or modelled implicit objects [12] can be integrated and manipulated with a common representation. These volume objects can usually be defined by continuous real functions as $f(x,y,z) \leq 0$ [7]. Conversion methods to convert one representation to another and details on voxel based object representation/manipulation are widely discussed in [13,14]. Different operations can then be applied on these objects from classical linear transformations and Boolean composition to more advanced algebraic operations [14], space mapping [15], sweeping by a moving solid [16] and Boolean compositions with soft [17,18] or "functionally defined" transitions [19].

Different families of volume modeller exist. Some, like the *BlobTree*, are essentially based on function representation and their composition in a CSG tree [18]. To obtain interactivity, only bounded skeleton primitives are used [20,21,22] and the surface is polygonalised for rendering. Others are based on discrete structures like regular grids [23]. There are also hybrid systems which integrate as primitives most of the volume representations [13]. The primitives are manipulated using their own representation. Different techniques are then available to interactively render the surface: polygonalisation, points, ray-casting using tri-linear interpolation, etc. The interactively visualised surface is always an approximation of the modelled object and depending on this approximation precision and on the sharpness of the shape, the error can grow, consequently reducing fidelity. ADFs [24] are adaptive octrees, which generally become deeper in proximity to the surface. They allow a faithful interactive rendering of the surface [25] but the value of the potential at a random point of the working space has to be computed as a distance from the surface, making combination with soft transition unsuitable for precise interactive modelling.

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 and offers a high shape representation quality.
- Visualisation is an accurate representation of the object.
- 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 "functionally defined" and point-by-point controlled transitions.

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 and justify our choice of data structure. Regular grids discretely storing the field values combined with a triquadratic approximation filter are used to defined volume objects. This association of a grid and an approximation/interpolation filter 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 the third section, the triquadratic filter 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 fourth section. We discuss the use of Boolean composition with "functionally defined" transition and we describe, as an example of 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. 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 continuous potential functions f: $R^3 \rightarrow R$. Among 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: $R^3 \rightarrow R$ 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[1] 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 ϕ: $R^n \rightarrow R$, where n is the number of primitives [17]. The result of an operation F=$\phi(f_1,..,f_n)$ 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 implied that the equation complexity grows with the size of the tree. With even a small number of levels, the function evaluation become expensive in computing time and rendering the shape in interactive time become 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 [26], Boolean operators with soft transition require, in addition, fast evaluation in the transition area [21,27,18] and Boolean composition with "functionally defined" 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 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 [28], 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 discontinuities while interpolating the values in the grid to evaluate the field at a point p of R^3.

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:

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

where f is the sampled function and g the filter. It is obvious that the quality of the reconstructed signal is dependent on the complexity of the filter. Thus the choice

---

[1] 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.

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 [29,30,31]. The most studied are: the linear filter, cubic filter based on BC-splines, gaussian filters and windowed sinc filters. Even though the linear filter is widely used in the volume rendering applications, studies have shown that it gives poor quality results. Better results can be obtained with cubic filters or windowed sinc filters. Indeed, the original function is better approximated and they allow smooth transitions due to a high degree of continuity. However, a tricubic filter requires 64 volume samples to reconstruct the function within a voxel, and windowed sinc filters need even more samples. Thus 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 that allows interactive renderings [32] with a high quality recontruction. It has been shown that quadratics are significantly faster to compute than cubics whilst providing comparable quality in the resulting interpolation [33]. 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 filter is used to accurately visualise the grid and if the rendering viewport is adapted to the grid's 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.

3. 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 approximation filter 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 interactivity.
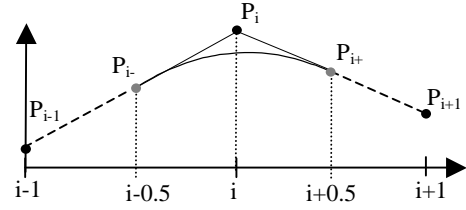
3.1. One-dimensional quadratic filter

Figure 3.1: 1D filter

Before introducing the triquadratic filter, we describe the filter in a one-dimensional case. Three neighbouring samples (P_{i-1}, P_i, P_{i+1}) are needed (fig. 1) 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:

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

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 :

$$f_{P_{i-1},P_i,P_{i+1}}(t) = (1-t).((1-t).P_{i-} + t.P_i) + t.((1-t).P_i + t.P_{i+})$$

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

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 the filter function is internally C^2 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 :

$$f'_{P_{i-1},P_i,P_{i+1}} = (1-t).(P_i - P_{i-1}) + t.(P_{i+1} - P_i)$$
$$= (1-t).G_{i-} + t.G_{i+}$$

with :

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

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 interpolation 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 filter. Indeed the central difference at the sampled points is the mean value of two middle difference gradients G_{i-} and G_{i+}. Thus the interpolation of the mean values ($M_i = 1/2.G_{i-} + 1/2.G_{i+}$) used in the central difference estimation leads to a loss of information in the gradient reconstruction, by regard to the use of the direct interpolation of middle differences G_i. However the quadratic filter is more sensitive to the data noise that is fortunately usually not present in the case of three-dimensional signals generated from equations.

3.2. Extension to volumes

The extension to three-dimensional volumes is straightforward and the previous results are also true while extended to this case. 27 volume samples ($3\times3\times3$) are now required to reconstruct the C^1 continuous signal within a parallelepiped aligned on the central sample (fig. 3.2). Like the usual trilinear interpolation, applying the previous filter 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.
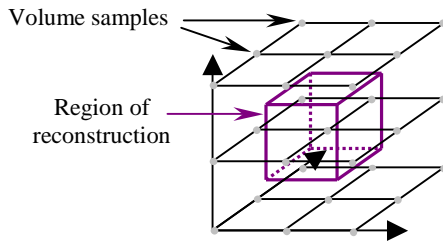

Figure 3.2: three-dimensional reconstruction

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

$$h_{P000..P222}(x, y, z) = \sum_{i,j,k=0}^{2} S_{ijk}.x^i y^j z^k . \quad \textbf{(1)}$$

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

3.3. Fast Visualisation

By now the visualisation of the implicit surface crossing this parallelepiped of 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 in an interactive time. 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:

$$x = x_0 + t.(x_1 - x_0), \quad y = y_0 + t.(y_1 - y_0), \quad z = z_0 + t.(z_1 - z_0) \ \textbf{(2)}^{[2]}$$

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

$$h(t) = \sum_{i=0}^{6} C_i.t^i \qquad 0 \le t \le 1. \qquad \textbf{(3)}$$

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 consisting in sampling many values (fixed to 16 here) along the ray has been 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 that can be computed 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 filter. This solves one of our requirements: The visualised shape is a faithful representation of the modelled one.

---

[2] Where (x_0, y_0, z_0) are the coordinates of the first point of intersection between a ray and a voxel, (x_1, y_1, z_1) are the coordinates of the second and $t \in [0,1]$.

3.4. Optimisations

Because many computations are required by the quadratic reconstruction, the visualisation process must be deeply optimised. Our optimised algorithm allows interactive rendering times (less than 1 second) for the visualisation of 256^3 voxels in a 512^2 viewport using an AMD Athlon processor at 1.0 GHz. Furthermore this algorithm allows interactive settings of the isosurface threshold. A complete description is given in [32], so we will just describe here the main optimisations.

The fast computation of coefficients C_i of the polynomial equation (Equation 3) represents the main requirement to perform the interactive rendering. They depend on both the S_{ijk} coefficients (Equation 1) and the ray parameters (Equation 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 specially increase raytracers performances while allowing to find the intersection between the ray and the sampled surface in $O(Log(n))$ steps [34]. Our implementation of an octree provides the same improvement. Rendering time is not anymore principally dependent on the volume size, but on the image size. The use of the Min-Max octree as an additional structure may seems to be a contradiction within our approach, but its construction time costs less than one second for a grid of 128^3 voxels and it allows our rendering technique to be interactive.

4. Potential function manipulation

The data structure and its visualisation method are now well defined. We point out 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 an AMD K7 at 1.0 Ghz processor and 256 Mbytes of DDR memory. 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 "functionally defined" precise 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 use in an interactive interface to precisely model volume objects. The application is an example given to illustrate our model capabilities.

4.1. Operators

Operators ϕ^{comp} defined[3] 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.

But grids are large data structures: Their storage requires a lot of memory. To define fine objects in a large enough modelling space, we use grids that are 128^3 float values (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 ϕ^{map} transforms the space in which the potential field is defined. While deformations like tapering, twisting or bending can be applied on

---

[3] Offsetting, Boolean composition with or without soft transitions and metamorphoses are elements of this operator set.

volume objects when their equation is known [35,18], 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 4.1. Specific research will have to be done to correctly surround this question and to propose suitable solutions. An introduction to this problem can be found in [36] which catalogues a variety of possible solutions. Thus, only bounded space mapping operators [37] having their bounded box included in the modelling space are able to be directly used on our volume objects. A solution to avoid this problem is to use only bounded primitives [16,38,18,23] sampled in an infinite grid [23]. 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.
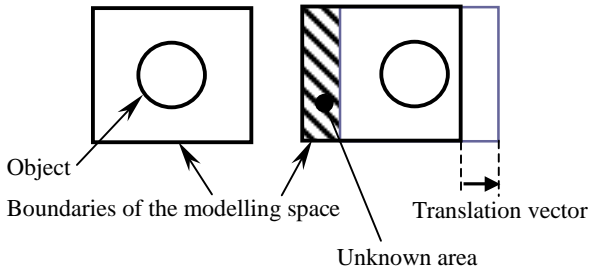

Figure 4.1: It is an open question how to compute the potential field values in the unknown area while conserving regular variation of the potential field.

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 can costs too much memory. To minimise the number of stored grids, we only use unary and binary operators. As shown in [17], 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, even the classical n-ary blending operator used for "blobs" can be decomposed and applied as a succession of binary blending operators [23].

Boolean composition operators with soft transition are considered as powerful and useful [20,39,7,18] for implicit modelling, therefore, one of our main preoccupations is to propose an accurate and interactive tool to realise them. As shown in [19], point-by-point control of the blend is

possible and the classical smooth and regular curved or inflated transition can be extended to a "functionally controlled" transition.

The transition is said to be "functionally controlled" when its form is defined by functions h: R→R. Moreover, the functions used to perform the transition preserve the metric of the combined primitives outside the part of the fields affected by the blend [19] and keep regular variations inside, which make this approach very relevant for composition of volume objects [40]. The used expressions for Boolean compositions are the following:

$$O_1 \cup O_2 : \phi^{bool}(f_1, f_2) = \min(f_1, f_2) - h_i(|f_1 - f_2|) \quad i=1,2 \ ,$$

$$O_1 \cap O_2 : \phi^{bool}(f_1, f_2) = \max(f_1, f_2) + h_i(|f_1 - f_2|) \quad i=1,2 \ ,$$

$$O_1 / O_2 : \phi^{bool}(f_1, f_2) = \max(f_1, -f_2) + h_i(|f_1 + f_2|) \quad i=1,2 \ .$$

Where ϕ^{bool} are binary Boolean composition operators, potential functions f_1 and f_2 represent the combined primitives and functions h_i: R→R (i=1,2) are 1D cubic polynomial splines controlled point-by-point [41] that define the form of the transition. Where $|f_1-f_2|=0$ ($f_1=f_2$), the min/max function generates a differential discontinuity (the tangent at this point is controlled to ensure C^1 continuity): f_1 is selected on one side of the frontier (the set of points where $|f_1-f_2|=0$) and f_2 is selected in the other. For each side a function h has to be defined. Function h_1 defines the transition in the side where f_1 is selected and function h_2 where f_2 is selected, as illustrated figure 4.2. More details are given in [19].
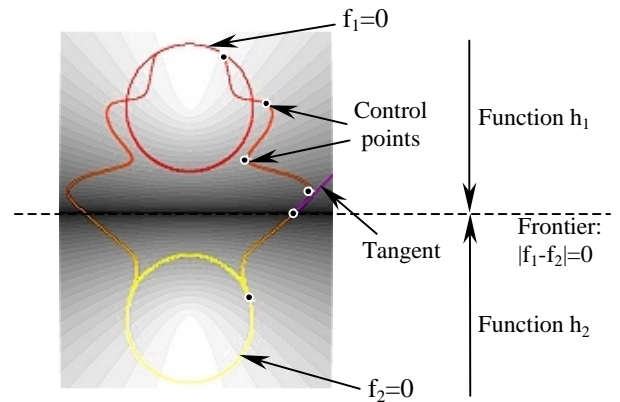

Figure 4.2: Union of two spheres.

As shown in [19], control points and vectors 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.

4.2. Application in a modelling interface

During the object rendering process, both frame-buffer and Z-buffer are computed from the grid and the triquadratic filter. 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 are used, the processor is mostly discharged of the interface component manipulation and rendering (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 just allows the visualisation of a single grid. A new primitive is positioned with regard to the current object, in visualising them in the same grid using the classical union operator (realised with the min function [26]). It takes 0.23 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 grid reconstructed by a triquadratic filter, a small oscillating transition of which the size depends on the grid resolution (Figure 4.3).
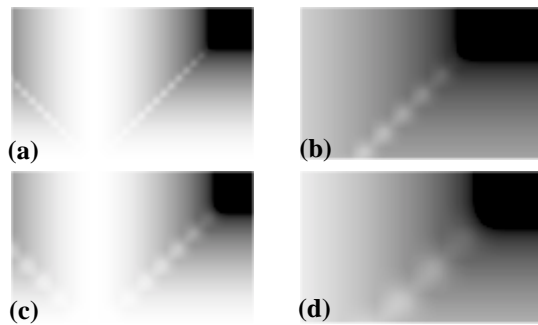
Figure 4.3: Sharp transition between two cylinders.

Grid	Without zoom	With zoom
128^3	(a)	(b)
64^3	(c)	(d)

It would be useful if these undesired oscillations could be avoided. A solution could consist in applying a filter to grid values. More research and experiments are needed in this area.

As described in section 4.1, when "functionally defined" transitions are used in Boolean operators, the form of the blend can be controlled by points and vectors selected in the modelling space. Depending on the operator complexity, the computation time of the grid varies following table 4.1.

Table 4.1: Time of computation of a grid while using a Boolean operator with a "functionally defined" soft transition depending on the number of control points used to defined its form.

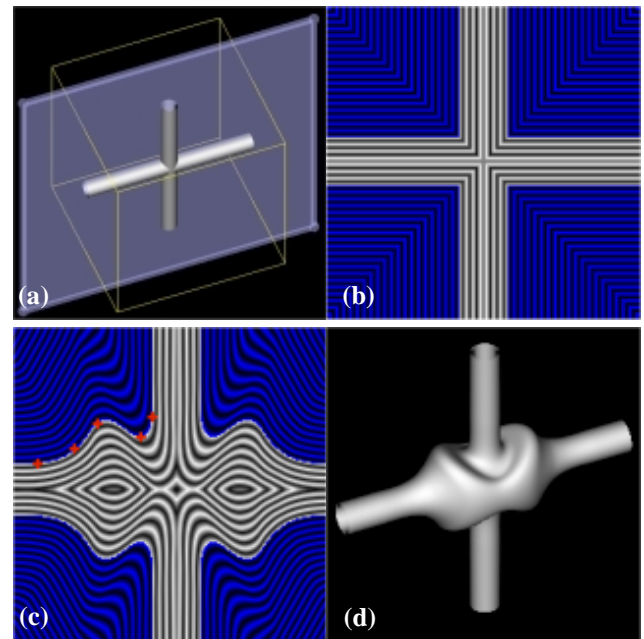| Number of control points	Computation of the grid	Computation of a plane section of 256^2 pixels, crossing the grid
3	0.5 sec	0.20 sec
4	0.7 sec	0.21 sec
6	0.9 sec	0.22 sec
8	1.2 sec	0.24 sec
12	≈ 2 sec	0.27 sec
20	≈ 3.5 sec	0.34 sec

Figure 4.4: 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 (c). The corresponding object is accurately visualised in the three-dimensional modelling space (d).

To allow the maximum of 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 4.4a). Then the section is rendered in a viewport of 512^2 pixels (Figure 4.4b) in approximately 0.08 seconds. The triquadratic filter is also used here to faithfully represent the field values of the volume. The transition is selected with the mouse in this plane section (Figure 4.4c) and when desired, the resulting surface can be visualised in the three-dimensional modelling space (Figure 4.4d).

Two-dimensional plane section representing the potential field values can be used to increase the interactivity (Table 4.1) 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.
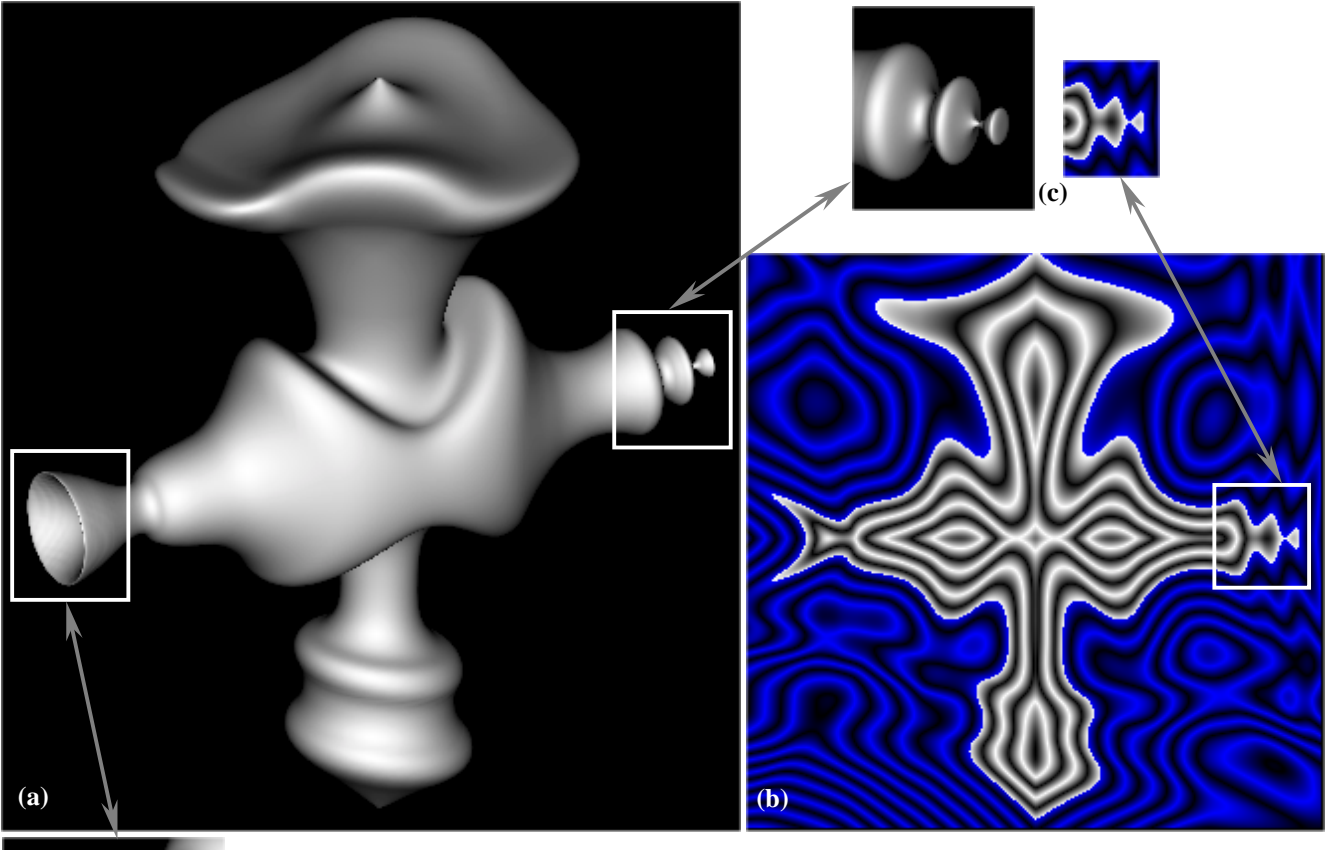
5. Results

Figure 5.1: 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).
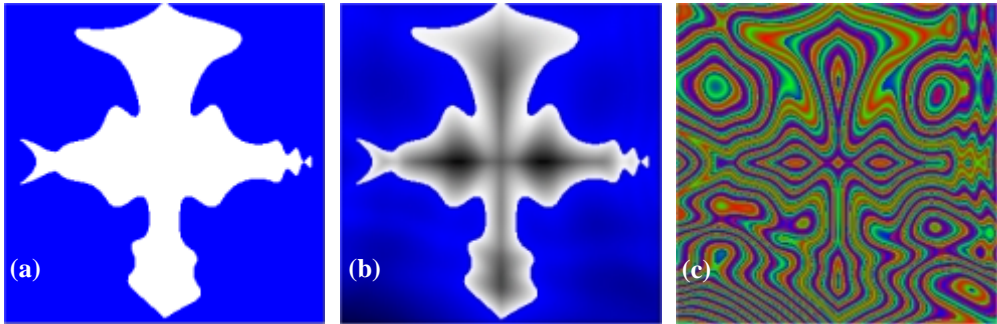

Figure 5.2: Different visualisations of the plane section of the potential field: (a) uniform colour, (b) variation of the colour with the potential field value and (c) tricolour isosurfaces.
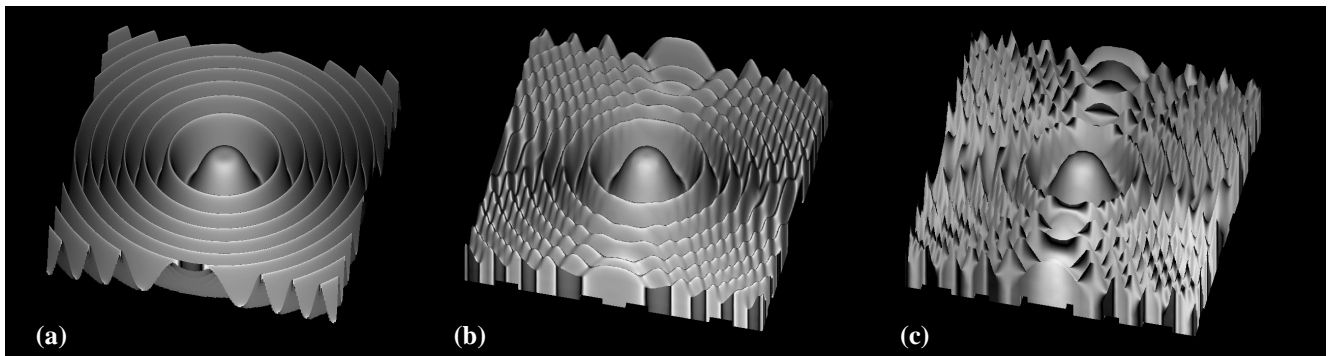
Figure 5.3: Rendering of the Marschner & Lobb dataset (41^3 samples) with an *ideal* filter (a), the quadratic filter (b) and the usual linear filter (c). We can notice that trilinear interpolation gives poor results, especially with surface shading. This is essentially due to the erroneous shadings provided by the separation between the signal reconstruction and the gradient reconstruction (dark surfaces in 5.2c).

6. Conclusion and future work

The combination of a discrete representation of potential fields with an adequate interpolation filter 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 filter generates a C^1 continuous function. Our rendering algorithm provides a fast, accurate and smooth visualisation of the field isosurfaces reconstructed with the triquadratic filter. 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 "functionally defined" transitions in an interactive modelling software.

For these reasons, the proposed modelling kernel corresponds to most of our wishes. Further studies can now to 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 V. Adzhiev and al [13], 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 filters 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.

References

[1] W. Lorensen and H. Cline, "Marching cubes: A high-resolution 3D surface construction algorithm", *SIGGRAPH'87*, 1987, pp. 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, pp. 451-458.

[4] G. Knittel, "The Ultravis System", *IEEE/ACM SIGGRAPH Volume visualization and graphics symposium 2000*, October 2000, pp. 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. Schwanecke and H-P Seider, "Feature sensitive surface from volume data", *Computer Graphics Proceeding*, Annual Conference Series August 2001, pp. 57-66,.

[7] 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*, June 1998, pp. 4-13.

[8] G. Turk and J.F O'Brian, "Shape transformation using variational implicit surfaces", *Computer Graphics Proceedings*, Annual Conference Series, 1999.

[9] 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, pp. 89-98.

[10] 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, pp. 67-76.

[11] M. Jones and R. Satherley, "Shape representation using space filled sub-voxel distance fields", *Proc. of Shape Modeling International'01*, May 2001, pp. 316-325.

[12] 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.

[13] V. Adzhiev, M. Kazakov, A. Pasko and V. Savchenko, "Hybrid system architecture for volume modelling", *Computer & Graphics*, 24(1), 2000, pp. 67-78.

[14] M. Chen and J.V. Tucker, "Constructive volume geometry", *Computer Graphics forum*, 19(4), pp 281-293, 2000.

[15] V. Savchenko and A. Pasko, "Transformation of functionally defined shapes by extended space mapping", *The Visual Computer*, 14(5/6), 1998, pp. 257-270.

[16] A. Sourin and A. Pasko, "Function representation for sweeping by a moving solid", *IEEE Transaction on Visualization and Computer Graphics*, 2(1), pp 11-18, 1996.

[17] A. Pasko, V. Adzhiev, A. Sourin and V. Savchenko, "Function representation in geometric modeling: Concepts, implementation and applications", *The visual Computer*, 8(2), 1995, pp. 429-446.

[18] 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, pp. 149-158.

[19] L. Barthe, V. Gaildrat and R. Caubet, "Extrusion of 1D profiles: Theory and first application", *International Journal of Shape Modeling*, 2002, to appear.

[20] J.F. Blinn, "A generalization of algebraic surface drawing", *ACM Transaction on Graphics*, 1(3) , July 1982, pp. 235-256.

[21] J. Bloomenthal and B. Wyvill, "Interactive techniques for implicit modelling", *Computer Graphics (proc. of SIGGRAPH'90)*, 24(2) , 1990, pp. 109-116.

[22] B. Crespin and C. Schlick, "Implicit sweep objects", *Eurographics'96*, 15(3), 1996, pp. 165-174.

[23] E. Ferley, M.P. Cani and J.D, "Gascuel. Practical volumetric sculpting", *The Visual Computer*, 16(8) , December 2000, pp. 469-480.

[24] 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.

[25] R.N. Perry and S.F. Frisken, "Kizamu: A system for sculpting digital characters", *Computer Graphics Proceedings*, Annual Conference Series, August 2001, pp. 47-56.

[26] A. Ricci, "A constructive geometry for computer graphics", *The Computer Journal*, 16(2) , 1973, pp. 157-160.

[27] 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.

[28] A. Kaufman, D. Cohen and R. Yagel, "Volume graphics", *IEEE Computer*, 26(7), July 1993, pp. 51-64.

[29] S.R. Marschner and R.J. Lobb, "An evaluation of reconstruction filters for volume rendering", *Proc. of visualization'94*, October 1994, pp. 100-107.

[30] 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*, vol.3, no. 2, April 1997, pp. 184-190.

[31] T. Theubl, H. Hauser and E. Gröller, "Mastering windows: Improving reconstruction", *Proc. of IEEE/ACM SIGGRAPH Volume visualization and graphics symposium 2000*, October 2000, pp. 101-108.

[32] B. Mora, J.P. Jessel and R. Caubet, "Visualization of isosurfaces with parametric cubes", *Eurographics'01 Proc.*, vol. 20 no. 3, September 2001, pp. 377-384.

[33] N.A. Dodgson, "Quadratic interpolation in image resampling", *IEEE Transaction on Image Processing*, 6(9) , September 1997, pp. 1322-1326.

[34] I. Wald, P. Slusallek, C.Benthin and M. Wagner, "Interactive rendering with coherent raytracing", *Eurographics'01 Proc.*, vol. 20 no. 3, September 2001, pp. 377-384.

[35] A.H. Barr, "Global and local deformations of solid primitives", *Computer Graphics*, 18(3) , 1984, pp. 21-30.

[36] N.A. Dodgson, "Image Resampling", *Technical Report no. 261*, University of Cambridge Computer Laboratory, August 1992.

[37] B. Crespin, "Implicit free-form deformations", *Proc. of Implicit Surfaces'99*, 1999.

[38] C. Grimm, "Implicit generalized cylinders using profile curves", *Proc. of Implicit Surfaces'99*, 1999, pp. 33-41.

[39] C. Hoffman and J. Hopcroft, "Automatic surface generation in computer aided design", *The Visual Computer*, 1, 1985, pp. 92-100.

[40] A.P. Rockwood, "The displacement method for implicit blending surfaces in solid models", *ACM Transaction on Graphics,* 8(4) , 1989, pp. 279-297.

[41] C. de Boor, *A practical guide to spline*, Applied Mathematical Sciences, 27, pp. 156-162, 1978.