Constructive Sculpting using 4D Spline Volumes

B. Schmitt*
LaBRI,
University of Bordeaux,
France.

A. Pasko†
Faculty of Computer and Information Sciences,
Hosei University,
Japan.

C. Schlick‡
LaBRI,
University of Bordeaux,
France.

Abstract

This paper presents an approach to constructive modelling of FRep solids defined by real-valued functions using Bspline volumes as primitives. A 4D uniform rational cubic Bspline volume is employed to define a 3D solid. While the first three coordinates are used to represent the spatial component of the volume to be sculpted, the fourth coordinate is used as a scalar, which corresponds to a function value or a volume density. Thus, the shape can be manipulated by changing the scalar control coefficients of the spline volume. This modelling process is interactive as the isosurface can be polygonized and visualized in real time. The distance property we obtain, combined with the properties of the spline volumes, allow us to use the resulting 3D solid as a leaf of a constructive modelling tree and to apply to it set-theoretic, blending and other operations defined using R-functions. Additional deformations can be achieved by moving arbitrary points in the coordinate space and applying space mapping at any level of the constructive tree. The final constructive solid is defined by a single real-valued function evaluated by the tree traversing procedure.

Keywords: FRep, Implicit Surfaces, Volume Sculpting, Extended Space Mapping.

1 Introduction

There are many different approaches to model complex shapes. One of these approaches can be the decomposition of this complex shape into a set of simple ones, called primitives, that are then combined with different operations. This approach can be called constructive modelling with the corresponding geometric data structure called a constructive tree. Each leaf of the constructive tree contains a primitive while the inner nodes contain the operations.

The most known technique following this approach is Constructive Solid Geometry (CSG, for short) developed during the 70s. In CSG, primitives are restricted to solids with simple shapes such as spheres, blocks, prisms and other regular shapes, and operations are limited to set-theo-retic ones (union, intersection, sub-traction) and linear transformations. The main limitation of CSG is that it can hardly be extended to a larger set of primitives or a larger set of operations. A more recent technique following the constructive modelling approach is the Function Representation (FRep, for short) [7]. Here, the primitives are mathematically defined with the use of a real-valued function f, which has to be C^0 continuous at any point of the space. The points belonging to the primitive are given by the inequality $f(X) \geq 0$, where X is a vector of point coordinates in the Euclidean space of any dimension. Any new primitive can be easily added to this set by providing a real-valued function f. At the inner nodes of the constructive tree, any operation (blending, set-theoretic, etc) yielding a real-valued function may be used. Nevertheless, building the constructive tree to generate a given shape is usually time consuming and sometimes difficult.

In the field of geometric modelling, there is an alternative approach called interactive volume sculpting. The corresponding techniques can be divided in two main families. In the first one, the sculpting process is based on the deformation of an existing shape [1]. In the second one, sculpting is done by adding or removing (carving) material. The main advantage of the second approach is its intuitivity as it mimics a true sculpting process. An interesting implementation of this approach has been proposed by Elber et al. [9]. In this technique, the object is represented as a zero set of Bspline volumes and the sculpting scheme offers multi-resolution control and real-time visualization, which makes virtual sculpting really intuitive and realistic.

A valuable approach would be to combine constructive modeling and volume sculpting, because both approaches contain useful features. Usually, a complex shape contains both regular parts that can be easily decomposed into a set of primitives, and some other parts more difficult to decompose but easier to model with a sculpting tool. To combine these approaches, one has to define a primitive than can be both sculpted and included in a constructive tree. To add a new primitive to a FRep tree, one has to verify the previously mentioned property (at least C^0 continuity of the function everywhere in the space.). In [9], the sculpting area depends on the space where the parametric function is defined, i.e., it is restricted within the boundary of the parameter space of the Bspline volume. As a matter of fact, such definition makes it difficult to use the sculpted object in another context, and especially to use it as a primitive. A similar approach has been proposed for Bézier volumes in [10] and extended to Bspline volumes in [11]. The main difference is in the mathematical framework. The same parametric function is used to define a solid, with the use of the inequality given by the function representation definition, but a so-called functional clipping is employed so that the function gets negative outside the parameter space. This ensures the function to take positive values only where the sculptor chooses to create some material.

In this paper, we first propose to use the primitives defined in [11], combine them in an FRep constructive tree and add some new features to improve the interactivity of the virtual sculpting tool, like a multiresolution scheme and a real-time visualization. In the second step, complex shapes modelled with this process can be combined with traditional primitives using different high-level operations like blending or twisting. Furthermore, once a constructive tree is built, one should be able to deform some parts of the whole shape. So, we also propose to define a general deformation, close to a carving scheme, where it becomes possible to move any point of the object in the space and thus to define a space mapping.

2 Volume Sculpting

2.1 Definition

In this section, we recall some definitions given in [11]. As we mentioned above, a 3D solid can be generated by defining a trivariate

*schmitt@labri
†pasko@k.hosei.ac.jp
‡schlick@labri

scalar function F(u,v,w) and finding the points where $F(u, v, w) \geq 0$. In our case, function F is a trivariate cubic uniform Bspline defined by a set of $l \times m \times n$ scalar coefficients λ_{ijk}, called control coefficients, that are placed on a 3D uniform grid. The corresponding expression is:

$$F(u, v, w) = \sum_{i=0}^{l} \sum_{j=0}^{m} \sum_{k=0}^{n} N_i(u) N_j(v) N_k(w) \lambda_{ijk} \quad (1)$$

where $N(t)$ are the cubic Bspline basis functions [2]. According to the definition, one has to understand that a control point is defined in 4D space, where the first three coordinates are used to locate it in the space, i.e. the usual xyz coordinates, and the fourth coordinate contains the scalar coefficient. By editing the values of the control coefficients, i.e. the last coordinate, the area where the function F gets positive can be changed and thus different shapes can be modelled. In the remainder of this paper, we will call a **Bspline volume** the volume defined by the positive part of this scalar function (even if this is not a Bspline volume in the sense of the spline literature).

This definition allows one to model complex shapes up to a certain limit. If it is used alone (i.e. is not mixed with other primitives) the behaviour of the function outside the parameter domain does not matter. But as we want to use it as a new primitive for a constructive tree, it has to respect the FRep definition, that is to be positive only in the domain of interest, and negative everywhere else. As the Bspline volume is parametrically defined, its behaviour outside the domain of interest is difficult to predict, and under some circumstances, may becomes positive and produces some ghost solids, as Fig. 1 shows . To avoid this problem, we use a functional clipping as proposed in [11].

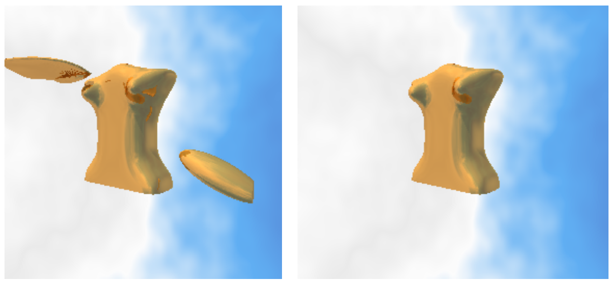


Figure 1: On the left, "ghosts" solids appear outside the domain. After application of the functional clipping, they disappear.

We can force the Bspline volume, or more generally any function, to become negative outside a certain domain. In our case, our interest turns toward the parameter space. It can be always considered as a unit cube with the use of some simple scaling operations. Then, the Bspline volume has to be negative outside this cube. Thus, a functional clipping can be expressed as:

$$F_{clip}(u, v, w) = F(u, v, w) \& \Omega(u, v, w), \quad (2)$$

With $\Omega(u, v, w)$ the unit cube, representing the parameter space, defined as :

$$\Omega(u, v, w) = \Omega(u) \& \Omega(v) \& \Omega(w) \quad (3)$$

where $\Omega(t) = t(1 - t)$ and & represents the intersection operation, defined as :

$$g(u) \& g(v) = g(u) + g(v) - \sqrt{g(u)^2 + g(v)^2} \quad (4)$$

The use of this intersection operation solves the discontinuity problem that is encountered if one uses the more classical minimum or maximum operation for this definition [7]. By applying this intersection operation to the Bspline volume and the unit cube, the resulting function will be negative outside the unit cube. Furthermore, this operation provides a distance property of the Bspline volume, the property that we will use later. This functional clipping also allows a largest degree of freedom when modeling with a Bspline volume. As a matter of fact, it becomes possible to change the scalar coefficients lying on the boundary of the control grid. Usually, those points are not modified in a wish of preserving a certain continuity, and edges are very difficult to obtain. By applying the presented functional clipping, it become easy to obtain edges on the border of a solid as it is shown in Fig. 2.

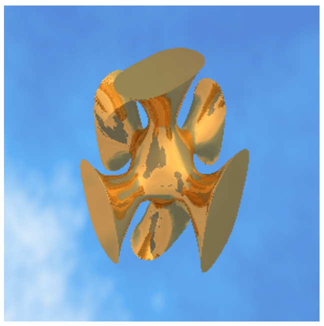


Figure 2: With the use of the functional clipping, it becomes easy to get edges on a Bspline volume primitive.

2.2 Modelling with Bspline Volumes

Modelling an object with a Bspline volume involves changing of the control coefficients. The modelling process we propose is achieved by first selecting one plane of control coefficients (remember that they are organised as a 3D uniform grid). In order to provide a visual representation of the control coefficients, we draw each vertex of the grid using a color chosen on the usual temperature color scale ("hot colors" correspond to positive values, while "cold colors" correspond to negative ones, see Fig. 3). Using this color scale, we are thus able to represent the 4D control coefficients. Some 2D painting tools are then proposed to the user so that he/she can easily change the values of the control coefficients. Increasing (decreasing) the values means adding (removing) material in the solid. Fig. 11 presents a screenshot of the interface we implemented, where a selected and already colored patch is presented. At the same time, this patch is drawn in the rendering area, in order to localise the part where the material is going to be added. As soon as a modification occurs, the update and the visualization of the colored patch and of the polygonized model is achieved in real-time.

To visualize the object, we polygonize the surface defined as $F(x, y, z) = 0$. Many different algorithms have been proposed for this task. We chose the polygonalization algorithm based on hyperbolic arcs proposed in [8]. As in the classical Marching Cube (MC) algorithm [4] exhaustive enumeration of the 3D grid cells is applied, but instead of using a look-up table to generate the polygons belonging to a given cell, this algorithm uses a trilinear interpolation inside the cell combined with a bilinear interpolation on the cell faces, and resolves ambiguities using hyperbolic arcs. The strength of the algorithm is that the polygonal model it generates is always correct, while MC algorithms requires expensive correction steps to solve all the ambiguity cases. In our implementation, with the use of this algorithm, the shape is updated in real-time, which leads to an interactive modelling tool.

One major weak point with Bspline volumes, is that modelling a complex shape with a lot of details requires a huge number of

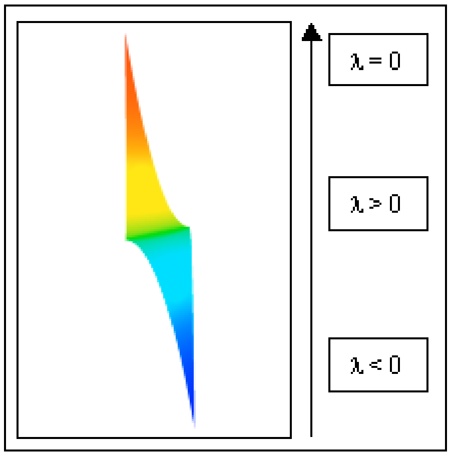


Figure 3: Color scale for the function value.

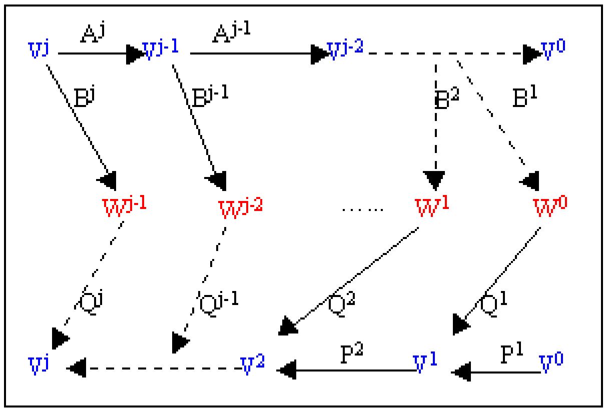


Figure 5: Wavelet Transform. Analysis step (top) and Synthesis step (bottom).

control coefficients to deal with. Note that this problem also exists when manipulating classical polygonal meshes as well as usual Bspline patches, and it has been widely studied in that context. One ubiquitous solution - initially proposed by Parent in 1977 [5] - is to hide the underlying control parameters (that may be mesh vertices, spline control coefficients, or whatever) and show only high-level sculpting tools to the user. More precisely, each tool is defined as a filter that is applied to a whole set of control parameters. This paradigm has been applied to zero-set surfaces defined by trivariate scalar functions in [9]. The resulting method is very powerful but may encounter problems in some cases. For instance, suppose we want to model a teapot. The easiest way is to create first a body without holes, and then create the holes by removing matter. If one considers only the surface of the object, it may be difficult to create this hole, because one cannot access to the inside part of the object. In our implementation, we choose to keep direct access to individual control coefficients, while allowing the use of high-level painting tools to edit a cluster of coefficients as a whole. Moreover, to enable manipulation of even a higher level, our implementation includes two other modelling tools, a wavelet transform (detailed below) and a constructive tree (detailed in the second section).

2.3 Wavelets Transform

Wavelets are a very powerful mathematical tool, that enables a decomposition of a vector space V into several equivalent representations. At a given level k, V will be described as a combination of a body space V^k and a detail space W^k, in other words, the information stored in V is split between V^k and W^k, without any loss. An introduction to the wavelet theory can be found in [12]. The process going from V^k and W^k to V^{k+1} is called synthesis, and the reverse process is called analysis. Both processes require two linear filters: A and B for the analysis step, P and Q for the synthesis one. Fig. 5 presents how to apply these filters to obtain a multiresolution decomposition.

The vector space we use in our case is the cubic Bspline volume defining function F. Thus to apply the multiresolution scheme, we simply have to define the four filters A,B,P and Q (the details of their calculation can be found in [12]).

Fig. 4 shows different steps of modelling a ”monster” using a wavelet transform. First, the Bspline volume is defined with a grid of $5 \times 5 \times 5$ control points (Fig. 4a), corresponding to the set V^1. Then, a single refinement (synthesis step) is applied to generate the set V^2; as it can be seen, the volume does not change (Fig. 4b). Some scalar values are then changed in set V^2 to generate Fig. 4c. The same two steps (synthesis step and edition step) are then done in space V^3 (Fig. 4d). Now, let us consider that the ”ears” of the monster are too close to the arms. By applying a wavelet transform (analysis step) and editing the coefficients at level V^2, one can change the position of the ears (Fig. 4e). If the ears are moved far enough from the arms in V^2 when going back to V^3, the arms are exactly recovered (Fig. 4f). The monster is now complete, with the correct position for the ears and the arms, Fig. 4g shows the result in 3D.

Multiresolution offers a valuable improvement of the modelling process. The decomposition of a shape into different levels of detail is intuitive to the user. Furthermore, it also avoids undesirable swellings. For instance, modelling the example shown in Fig. 6 (a donut-like with a ball in the hole, and a column) with a single Bspline volume may be very difficult without the multiresolution scheme. On the left part of the Fig. 6, the whole solid was directly modelled at a high resolution. Such a process generates some swelling artefacts on the column and small undesirable angles on the donut. This is only due to the too large number of control points, and the difficulties it causes in manipulation of them. On the right part the donut, the column, and the ball were modelled at different levels. This time, swellings and other artefacts are removed.

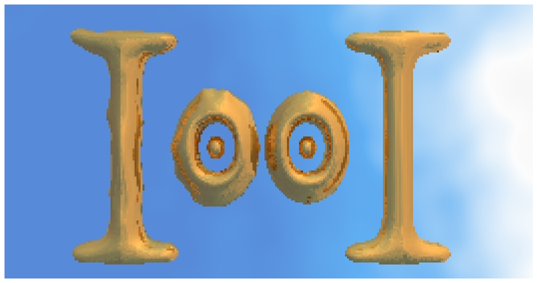


Figure 6: Modelling in a high resolution leads to some swellings. A multiresolution solves this problem.

3 Constructive Tree

In this section, we propose to build a FRep constructive tree using the function representation of geometric solids, and more especially using the primitive proposed above. Actually, many primitives, such as sphere, block, ellipsoid or convolution surfaces can be used

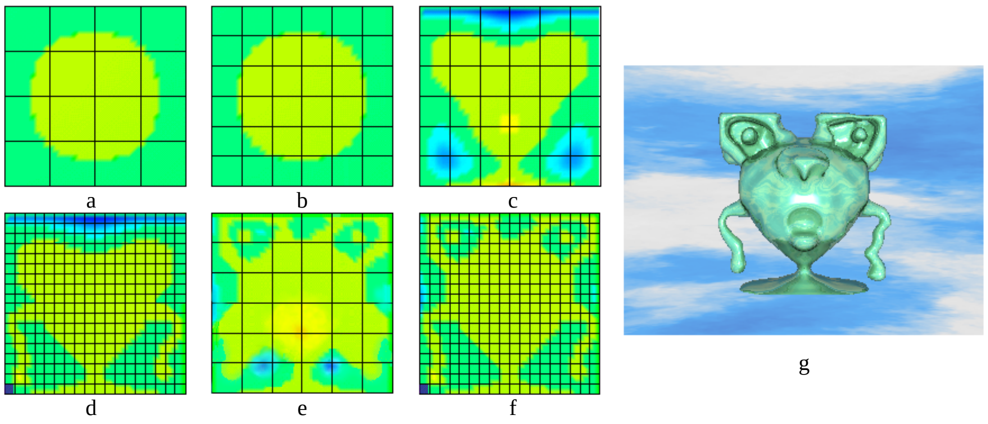


Figure 4: Different steps (from a to f) of the modelling process of a "monster" using multiresolution. The final result in 3D is shown on the right.

with it, as well as many operations, such as set-theoretic, blending, twisting and others. A more complete description of available nodes and leaves for this constructive tree can be found in [8]. In the remainder of this section, we propose first to describe the hierarchical refinement of a Bspline volume we use, similar to the one proposed in [3] and [9]. Then we discuss a new transformation, which can be used to modify the current model at any level of the constructive tree.

3.1 Refinement: sum, union, blending

Refinement of a Bspline volume has been first introduced in [3] for parametric spline surfaces. The proposed method consists of building a hierarchical tree of surfaces. Starting from a root surface, sub-surfaces, called overlays are locally defined. Each overlay can also be refined with new sub-surfaces of finer resolution, to generate a tree of surfaces, where the resolution of each overlay is function of the depth of the tree. This approach has been applied to Bspline volumes in [9]. The principle is to use an octree to sculpt a model at different levels of details. This feature is very useful. For example, if one wants to model a teapot, he may first model the body of it, and then, in a finer resolution the spout, the ear and the cover. To obtain the whole model, a sum between sub-volumes is achieved during a tree traversing process. To preserve a C2 continuity, in the case of the cubic Bspline, the two first and last rows and columns of scalar coefficients are set to zero. Even if this condition is enough to preserve the continuity inside the resulting function during the overlays addition process, the sum operation may lead to unexpected result.

Fig. 7 shows a simple example in the case of a 1D solid. The first step is to model two segments AB and CD with a single Bspline curve (Fig. 7a). Then, we operate a refinement in order to add an another segment between them. This is achieved by adding an overlay (a Bspline curve as well). As it can be seen, on the extremities of the sub-curve, the continuity is preserved, and it reaches smoothly the main curve. The result of the sum between those two curves is an 1D solid composed of three parts, AB, CD and EF (Fig. 7c). Now, suppose that we want to translate the added part EF. As it can be seen, if one applies a simple sum, this part disappears, and one has to change the overlay value and by the way return to the modelling process to change the scalar coefficient of the control points (Fig. 7e). On the other hand, one can consider this overlay as a curve by itself (Fig. 7b). Thus overlays can be used as a simple primitive for an FRep tree, and a union can be applied to them. The result is similar to the sum operation (Fig. 7d). But now, when the overlay is translated, the expected result is obtained (Fig. 7f). The function used for the union is similar to the one defined previously for the intersection :

$$g(u)|g(v) = g(u) + g(v) + \sqrt{g(u)^2 + g(v)^2} \tag{5}$$

Fig. 8 presents a 3D example, namely a teapot modelled by using different operations. The sum operation (left) leads to a hole inside the body of the teapot, whereas a union operation (middle) preserves its shape. A light blending has been applied in the right picture. It shows some material have been added to the junction between the two parts, and produces a smooth transition.

3.2 Space Mapping

In this section, we propose to define a local deformation which helps modify, for any given point, the shape defined by a constructive tree. Such transformation may be applied at any level of the constructive tree. For instance, if one considers a simple tree obtained by a blending union of two spheres, it becomes possible to transform the material added by the blend.

To define such transformation, let us first consider a simple translation. Let f be a defining function, and (dx, dy) a vector of translation. Then, the transformation is defined as :

$$T : f(x, y) \rightarrow f(x - dx, y - dy) \tag{6}$$

This operation is global (i.e., applied to the whole object). Now, let us define the space mapping. Consider a 2D solid and the displacement of a point A toward a point A'. To define a local deformation centred around point A, one has to respect the two following properties:

- (dx, dy) have maximum values at A'.
- (dx, dy) drops to zero when (x, y) is far from A'.

Then, the linear displacements dx and dy can be changed to two functions $dx(x, y)$ and $dy(x, y)$. Any bell-shaped curve respects both properties. The following definition can be used:

$$\begin{cases} if \gamma > \epsilon & \\ & x = x - e^{-\gamma} \times (x_{A'} - x_A) \\ & y = y - e^{-\gamma} \times (y_{A'} - y_A) \\ else & \\ & x = x \\ & y = y \end{cases} \tag{7}$$

where

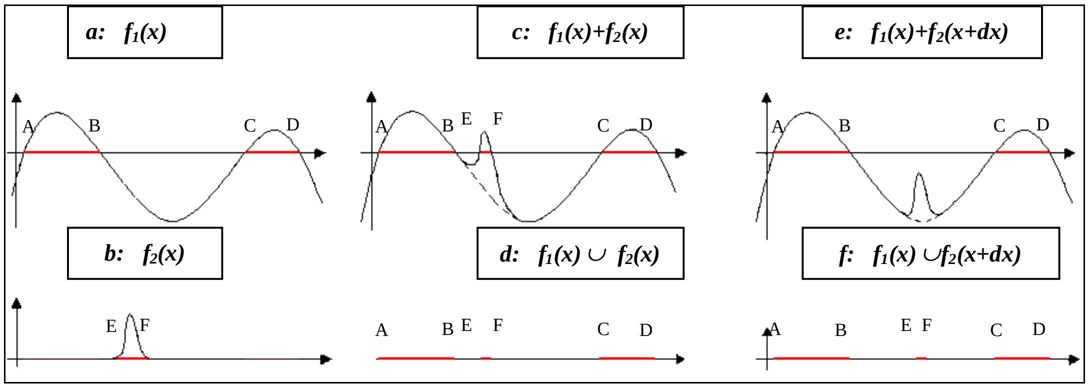


Figure 7: Difference between a sum and a union between two curves, in the case of a 1D solid.

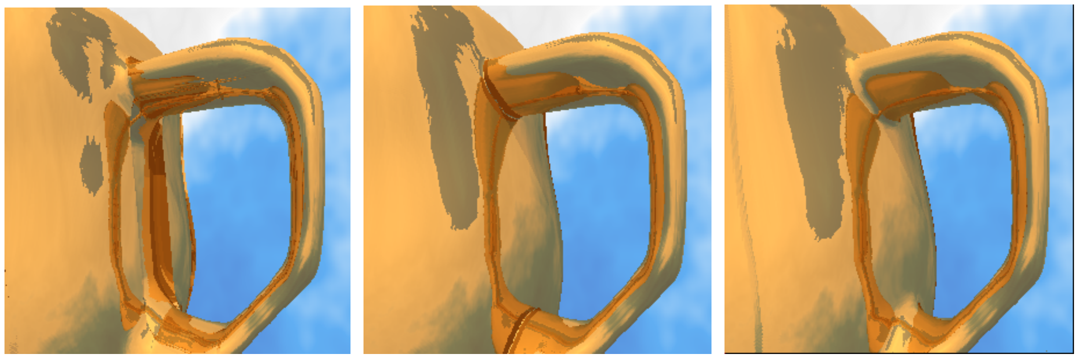


Figure 8: Difference Difference between sum, union and blending. Extension to the 3D case. From left to right, the applied operation is sum, union and blending.

$$\gamma = \frac{(x - x_A)^2 + (y - y_A)^2}{p} \; and \; p \in]0, +\infty[\qquad (8)$$

As it can be observed, the displacement is maximum when the considered point is placed on A'. We set explicitly the displacement to zero when (if greater than an ϵ', because our interest is only in the "bell" of the curve, and its other parts might introduced some undesirable small positive value). Note that the point A is arbitrarily selected, and thus, a space mapping can be defined for any given point in the space. If one consider a set of displaced points, the definition (2) becomes :

$$\begin{cases} x = x - \sum_{i=1}^{n} e^{-\gamma} \times (x_{A'_i} - x_{A_i}) \\ y = y - \sum_{i=1}^{n} e^{-\gamma} \times (y_{A'_i} - y_{A_i}) \end{cases} \qquad (9)$$

where A and A' are two set of n points, containing respectively original points A_i and moved point A'_i.

The parameter p is a real value given by the user, where it defines the area of influence of the space mapping, (and thus the shape of the "bell"). Fig. 9 shows results for different values of p in the 3D case. As it can be seen, different levels of deformation, local or global, can be obtained. The base shape is an ellipsoid. Each line represents a different value of the p parameter. The first row represents the effect of the displacement of a single point inside the solid, and the second row, the displacement outside.

For any given point in the space, one can apply this transformation. If more than one point is moved, then a sum of all the displacements is applied.

One problem may raise if a point is moved too far from its original position. Indeed, the parameter of the equation defines a radius of influence of the point. If the point is too far, its potential may not be sufficient to influence the original shape. To overcome this problem, one may increase the value of parameter p, but the result will be a more global deformation. The solution we propose is to build a stack of transformations, where small transformations are sequentially applied. In our implementation, the object defined by the constructive tree is first polygonized using the algorithm described in the previous section. Then, to apply local deformation, a vertex of the polygonal mesh is selected and moved to a new location. Local re-polygonalization is then achieved in real time, up to a certain limit (if parameter p is small enough). As it can be seen in Fig. 10, this space mapping is comparable to a carving scheme.

4 Implementation

We have implemented an interactive modeller using Tcl/Tk for interfacing and MAM/VRS [14] for graphics. In order to allow an intuitive modelling process, we choose to design a single user interface where all operations can be presented. With this interface, one can:

- Sculpt an object with a Bspline volume.
- Add traditional primitives: spheres, blocks, ellipsoids, etc.
- Apply three kinds of operations: first there are usual affine transformation (rotation, translation, scaling), second there are set-theoretic ones (union, intersection, subtraction) and blending versions of them, finally there are general deformations (twisting, tapering, stretching, space mapping, etc).

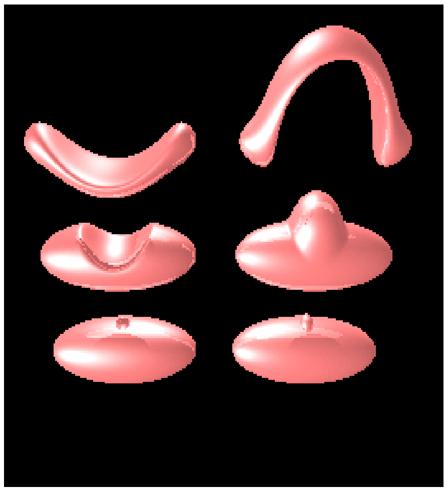


Figure 9: Example of space mapping with displacement inside the solid (left) and outside (right) with three different values of p.

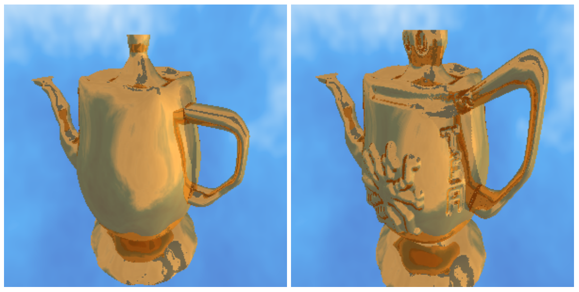


Figure 10: Application of the space mapping: an original teapot and a carved teapot.

- Modify any node or leaf of the constructive tree at any time during the modelling process (for instance, change the blending union parameters).

Fig. 11 is a snapshot of the modeller taken during a modelling session. The window is divided into two main areas. The lower one is the representation of the final object, evaluated by the tree traversing procedure, the upper part contains the description of one node or leaf belonging to the tree. This upper part can contain two different kinds of information. If the selected object is a primitive or a node, it contains the relevant parameters (for instance, radius and center for a spherical primitive). If the selected object is a Bspline volume, the upper part changes into a canvas containing a set of control points. As it was mentioned previously, the sculpting process of a Bspline volume is achieved by changing the control coefficients belonging to the same plane. Thus, the user can select such a plane before using some 2D painting tools to change the values. As it can be seen in the figure, the upper right part contains the visualization of this plane where the value of the control coefficients are shown using a temperature color map. In a real modelling process, this selected patch should appear in the lower part, in order to locate where material is going to be added.

The resulting solid is shown in Fig. 14, and details of the back side are shown in Fig. 13. The corresponding constructive tree is presented in Fig. 12. It is composed of various objects and operations :

•Ellipsoids (3)	•Spheres (2)
•Block (2)	•Bspline Volumes (11)
•Solid Noise	• Blending / Union
•Others	

Note that currently, only the sculpting steps are achieved in real time during the modelling process. When one uses a blending operation or applies a twist deformation, the amount of update it requires is usually too large to offer real-time repolygonization (but it is still done in less than 60 seconds for objects of an average level of complexity). For instance, to generate the complex model presented in Fig. 11, an 80 grid has been used, and the polygonalization took 90 seconds on a Pentium 400MHz. In our implementation the object can be saved as an HyperFun script to be used in the HyperFun software environment [13]. This language is a high level language supporting exchange of FRep models.

5 Conclusion

In this paper we have presented a new modelling technique for 3D solids that mixes two usual modelling paradigms, namely interactive sculpting and constructive modelling. The solids are defined by a 3D scalar function evaluated by the tree traversing process. The leaves of this constructive tree are given by 4D uniform rational cubic Bspline volumes, which are combined using set-theoretic, blending and other R-function operations. The shape can be manipulated by changing the scalar control coefficients of the spline volumes. Additional deformations can be achieved by moving arbitrary points in the coordinate space and applying space mapping at any level of the constructive tree. The isosurface of the 3D scalar function can be polygonized and visualized in real time to get an interactive modelling tool.

We are currently adapting the interface of our system so as it can by controlled by a Data Glove device. The ultimate goal is to propose a virtual sculpting environment where a sculptor can create an object without changing (or changing as few as possible) the gesture he is used to for his creative process. Another work currently in progress, is to create a similar tool for solid texturing of 3D objects [6].

References

[1] S. Coquillart. A sculpting tool for 3d geometric modelling. *Computer Graphics, Siggraph*, 24:205–212, 1988.

[2] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide.* Second Edition, Academic Press, 1990.

[3] R.D. Forsey and R.H. Bartels. Hierarchical bspline refinement. *Computer Graphics*, 22(4):205–211, 1988.

[4] W.E. Lorensen and H.E. Cline. Marching cubes : A high resolution 3d surface construction algorithm. *Computer Graphics, Siggraph*, 21(4):163–196, july 1987.

[5] R. Parent. A system for sculpting 3d data. *Computer Graphics*, 11(8):138–147, 1988.

[6] A. Pasko, V. Adzhiev, and B. Schmitt. *Constructive Hypervolume Modelling*. Technical Report TR-NCCA-2001-01, 2001.

[7] A. Pasko, V. Adzhiev, A. Sourin, and V. Savchenko. Function representation in geometric modelling: concept, implementation and applications. *The Visual Computer*, 11(8):429–446, 1995.

[8] A. Pasko, V.V. Pilyugin, and V.V. Pokrovski. Geometric modelling in the analysis of trivariate functions. *Computers and Graphics*, 12(3/4):457–465, 1988.

[9] A. Raviv and Elber G. *Three dimensional freeform sculpting via zero sets of scalar trivariate functions.* Technical Report CIS9903, 1999.

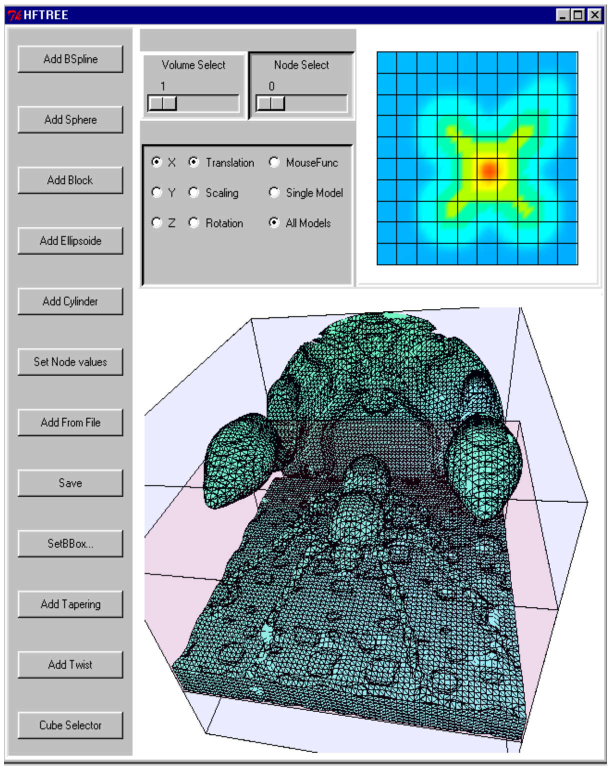
[10] B. Schmitt, A.Pasko, and V. Savchenko. Extended space mapping with bźier patches and volumes. *Implicit Surfaces '99*, pages 25–31, September 1999.

[11] B. Schmitt, M. Kazakov, A.Pasko, and V. Savchenko. Volume sculpting with 4d spline volumes. *CISST'2000*, 2:475–483, September 2000.

[12] E.J. Stollnitz, T.D. Derose, and D.H. Salesin. *Wavelets for Computer Graphics : Theory and Applications.* Morgan Kaufmann, 1996.

[13] Url. *HyperFun Web Site*. http:://www.hyperfun.org.

[14] Url. *MAM/VRS Web Site*. http:://www.math.uni-muenster.de/informatik/mam/.



Volume Select 1

Node Select 0

[✓ X] Y Z

[✓ Translation] Scaling Rotation

MouseFunc Single Model [✓ All Models]

Figure 11: Snapshot of the GUI using concepts proposed in this paper.

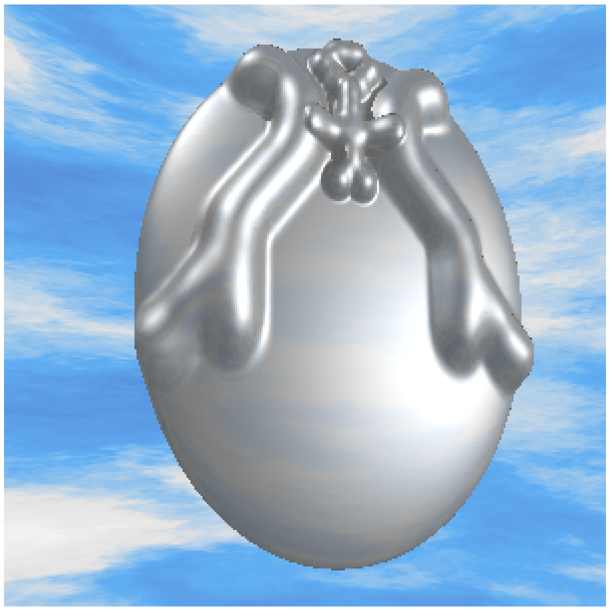


Figure 13: Detail of the back of the Fig 14. It consists of an ellipsoid modified by different space mappings.

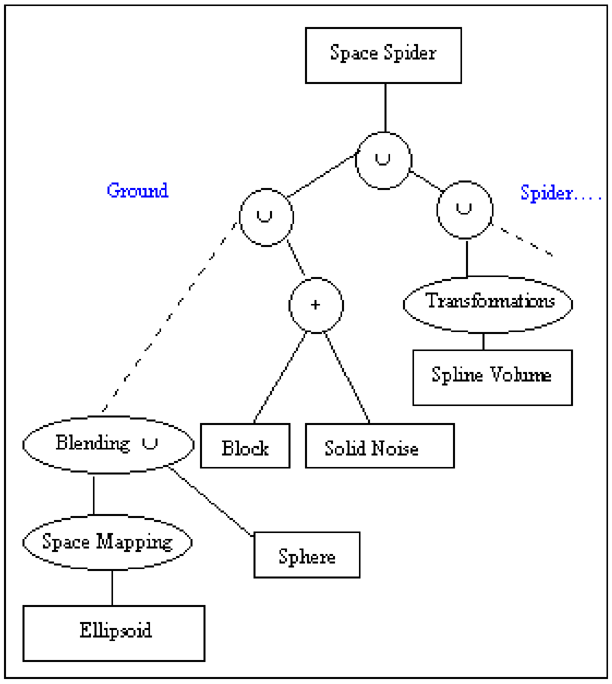


Figure 12: Constructive tree corresponding to the ”space spider” (Fig 14).

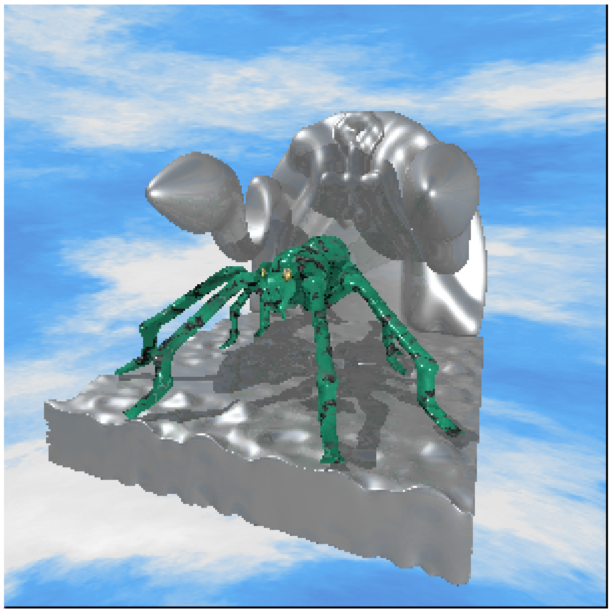


Figure 14: 3D solid modelled by building a tree with simple primitives and sculpted object using BSpline volumes.