A Sketch-Based Modelling system using Convolution Surfaces

Anca Alexe[1], Loïc Barthe[1], Marie Paule Cani[2], and Véronique Gaildrat[1]

[1]IRIT - CNRS, UPS Toulouse, France
[2]GRAVIR - CNRS, INRIA, UJS, INP Grenoble, France

Abstract

This paper proposes a user-friendly modelling system that interactively generates 3D organic-like shapes from user drawn sketches. A skeleton, in the form of a graph of branching polylines and polygons, is first extracted from the user's sketch. The 3D shape is then defined as a convolution surface generated by this skeleton. The skeleton's resolution is adapted according to the level of detail selected by the user. The subsequent 2D strokes are used to infer new object parts, which are combined with the existing shape using CSG operators. We propose an algorithm for computing a skeleton defined as a connected graph of polylines and polygons. To combine the primitives we propose precise CSG operators for a convolution surfaces blending hierarchy. Our contributions also include the extension of the inverse convolution kernel - more efficient than the standard pseudo-Cauchy kernel - to variable radii segment skeletons. Our new formulation has the advantage of requiring no optimisation step for fitting the 3D shape to the 2D contours. This yields interactive performances and avoids any non-desired oscillation of the reconstructed surface. As our results show, our system allows non-expert users to generate a wide variety of free form shapes with an easy to use sketch-based interface.

Keywords: sketch based modelling, implicit surfaces, convolution surfaces, CSG

1 Introduction

The complexity of user interaction is the main obstacle to the use of standard modelling systems. This impacts both the user and the possibilities of expression this system provides. Achieving a simple and faithful translation of the user's idea without requiring sophisticated input and a long training process remains a challenge for the modelling software. One of the simplest and user-friendliest modelling metaphors is drawing. This kind of communication is useful in educational applications such as teaching, and already has industrial purposes such as story boarding. It is generally used in the early stages of design, because drawing a sketch is both much faster than creating a 3D model, and more convenient for expressing ideas. However, the obvious drawback of 2D sketches is their limitation to a single viewpoint. The user cannot move around the drawn object, nor view it from different angles, except in [6] where the sketch cannot be used for extracting a 3D shape. The aim of the sketch-based modelling is to combine the simplicity of 2D sketching with powerful 3D capabilities. Since the first sketch based interface [18] the concept has been largely developed and explored, from architectural design [8] to artistic design [12] and free form shapes [10, 11, 16]. The latter are difficult to model with sketches, though among the most interesting because of the large modelling possibilities they provide.

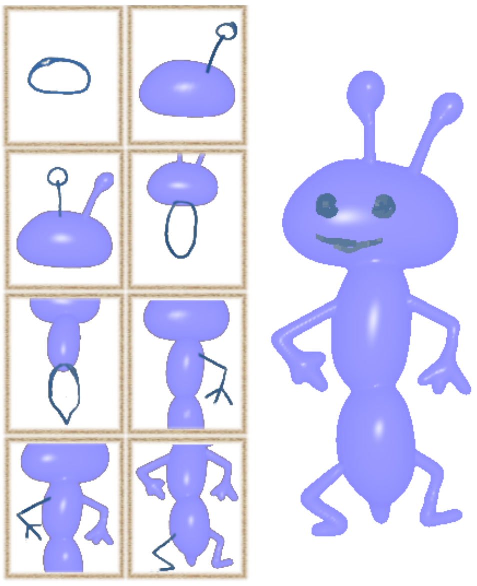

Figure 1: *Modelling a cartoon character with our system.*

The main difficulty in reconstructing a 3D model from a 2D contour is extrapolating lacking information. There are two main approaches for constructing smooth, rounded shapes from 2D contours. The first one consists in perspective projections of the contour point samples to reconstruct the 3D geometry. These points are then interpolated using variational implicit surfaces [11, 19, 7]. The second is to construct a skeleton from the 2D contour and use it to generate a 3D shape [10, 16, 1]. The main drawback for the first approach is that the surface has to be recalculated every time it is edited and the time taken to compute the coefficients for the variational implicit surfaces increases with the number of points. Also, small details are lost when blending the object parts because preserving them would require too many constraints and too much computation. Therefore we prefer the second approach. Previous research in this field has raised some difficulties. One of these is the necessity of an optimisation step to adjust the implicit surface to the drawn contour. This leads to a better contour approximation in terms of error but the surface oscillates [16] [1]. Moreover it is time consuming and in the context of sketch based interface providing very accurate reconstruction is not necessary. Indeed, the user drawn contour is seldom noisy so we rather aim at getting a smooth shape with close appearance to the contour. Removing the optimisation step saves time and reconstructs a smooth non-oscillating surface. Of course the contour approximation constraints have to remain satisfied. Another drawback of most of the previous approaches is that the shape thickness is automatically inferred so the result may differ from what the user wanted. For example if the user draws the shape of the palm of a hand, the fingers will be smoothly reconstructed as cylinders, whereas the palm will look like a sphere, far from the users expectation. In [16] the problem is addressed by asking the user to provide an additional information about the cross section's profile. This increases the complexity of the interface and for this reason the technique might not be intuitive enough for non-expert users. In [1] it is possible to control the thickness of the resulted shape by scaling the basis function used.

Our contribution

We propose a representation that allows for a great variety of topological shapes, a richer collection of sketch-based operations, an adaptive level of detail for sketch modelling with precise control of the result up to small details, while keeping a very simple and friendly user interface. For this purpose we reconstruct the 3D shape using convolution surfaces [5] with both polylines and polygons skeletons. The primitives are composed with CSG blending in a blending hierarchy. In order to be suitable for interactive modelling, the convolution surface has to be fast to evaluate. For this reason we use a convolution kernel that gives a closed form solution for the convolution integral. At present only the pseudo Cauchy [14] kernel satisfies this condition for the primitives that we use (segments and triangles). In this paper we show that an extension of the inverse kernel [17] provides a closed form solution segment with linearly variable radius and for triangles. This extended inverse kernel is faster that the pseudo Cauchy kernel, which however provides an easier blending control. This is especially important since we do not use any optimisation process to fit the surface to the contour. The two kernels are analysed and compared.

Section 2 presents our system from the user's point of view. Section 3 details the use of convolution surfaces. Section 4 presents the CSG composition model and Section 5 presents the application from the system's point of view, i.e. the algorithms and the techniques used. Section 6 shows and discusses some results. It also draws the conclusions and perspectives of our work.

2 From the user viewpoint

The purpose of our system is to enlarge the possibilities offered by the paper-pencil 3D modelling metaphor, while keeping a simple and intuitive input interface. The modelling process iterates the following steps until modelling is complete:

1. The user draws one or several strokes
2. The strokes are interpreted to reconstruct a 3D object part
3. The part is added to the current object (or subtracted if carving)

2.1 User input overview

As the user draws a stroke, its thickness and colour intensity vary proportionally with the pressure on the digital pen, as to imitate the irregular density and thickness of the strokes produced by a real pen. Several strokes accumulated in the same pixel result in a darker colour for that pixel. The other end of the pen is used as an eraser. As long as the stroke has not been reconstructed, the user is free to erase and modify it. This way the user's input is allowed to be noisy and irregular, as it is naturally on paper (see Fig. 6 (a)). The Undo command is also available.

2.2 Creating a new shape

The user draws a contour on the graphic tablet using the digital pen. Once the contour has been completed the user presses the digital pen against the tablet. This produces the 3D reconstruction of the stroke (see Fig. 2 (a),(b)).

2.3 Extending or carving a shape

The mechanism is the same as for creation. The first surface point hit by the user, which must be a point on the object, gives the depth of the shape to be constructed. When the stroke is complete, the user presses the stylus if he wishes to add the shape to the existing object, or the eraser (at the opposite pen's end) of he wishes to carve it into the object. The shape is reconstructed in such a manner that the projection of the shape on the screen fits the contour that has been drawn by the user (see Fig. 2 (c),(d),(e) and (f),(g),(h)).

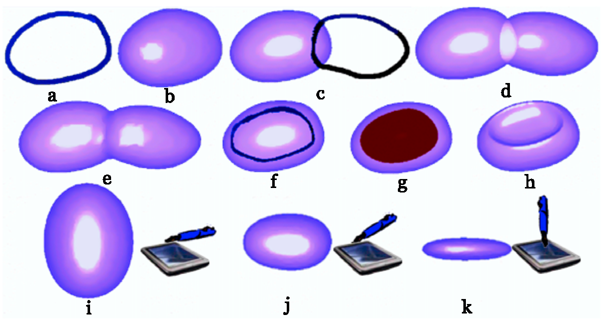

Figure 2: *(a), (b) Creating a part. (c),(d),(e) Adding a part to an object. (f),(g),(h) Carving. (i),(j),(k) Thickness control (side view).*

2.4 Thickness control

The user controls the thickness of the shape as follows: if the pen is held orthogonal to the tablet while pressing to reconstruct, the shape is a flat one (see Fig. 2 (k)), otherwise it is a thick one (see Fig. 2 (i)), the thickness being dictated by the pen's bend.

2.5 Adaptive modelling

The user can model smaller details, by zooming to get closer to the object. The large object parts will smoothly blend with each other, while the small details (e.g. eyes, nose of a character) will have a sharper blending.

2.6 Painting and annotating

The user can paint directly on the objects or in space next to them. In this way additional information or annotation can be added to the model.

3 Convolution surfaces

3.1 Definitions

A convolution surface is expressed by the equation

$$F(p) = \mathbf{T}$$

where T is the constant iso-surface value and $F(p)$ is a potential function from R^3 to R defined by the convolution product of a geometric skeleton function $g(p)$ and a kernel function $h(p)$:

$$F(p) = \int_S g(\mathbf{r})h(p - \mathbf{r})d\mathbf{r}$$

A consequence of the basic convolution definition is that a skeleton convoluted with a kernel produces the same result as the sum of its composing primitives convoluted with that same kernel. In order to allow a closed form solution for polylines, it is sufficient to have a closed form solution for segments. In a similar manner, a closed for solution for polygons can be obtained if the kernel has closed form solution for triangles. To allow for varying radius segments we extend the inverse kernel[17] as:

$$h(p) = \frac{w(p)}{d(p, S)}$$

where w is the weight attached to the skeleton at the point p and d is the distance from the point p to the primitive's skeleton. One must be careful when $d = 0$, as the function is not defined for that value, so we clip it at $1/\epsilon$ with a small value for ϵ.
The pseudo Cauchy kernel, as presented in [14] is given by:

$$h(p) = \frac{1}{(1 + s^2 d^2(p, S))^2}$$

The closed form solution for the pseudo Cauchy kernel has been given in [16] for linearly varying radii segments and in [14] for triangles. The next section analyses the extended inverse convolution kernel.

3.2 Segments with varying radii

We consider the case of a segment with the weight varying linearly from w_0 to w_1 and we denote h the distance from the point p to the segment (see Fig. 3 (a)). a_1, a_2 are the distances from H to A_1 respectively from H to A_2. u is the segment axis, d is the vector from H to the point p and $x = du$. The integral to be calculated is:

$$F(p) = \int_0^L \frac{Mt + N}{\sqrt{d^2 + t^2 - 2tx}} dt$$

where $M = \dfrac{w_1 - w_0}{a_1 - a_0}$ and $N = \dfrac{a_1 w_0 - a_0 w_1}{a_1 - a_0}$

This leads us to the following closed form solution:

$$\begin{aligned} F(p) &= M\sqrt{d^2 - 2xL + L^2} \\ &+ (xM + N)ln\left[L - x + \sqrt{d^2 - 2xL + L^2}\right] \\ &- Md - (xM + N)ln(d - x) \end{aligned}$$

The following table compares the number of operations required to compute the convolution integral for both kernels.

Kernel	*	/	+	-	Other
Pseudo Cauchy	25	3	11	6	1sqrt 2atan
Inverse	16	0	8	11	2sqrt 2ln

Table 1: *Evaluation for variable radius segment skeleton.*

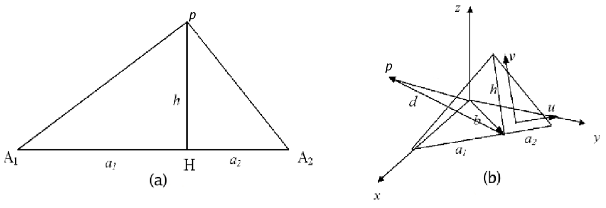

Figure 3: *Notations used in computing the convolution (a) with a line segment (b) with a triangle.*

3.3 Triangles skeletons

In order to compute the convolution with a triangle skeleton, lets consider the perpendicular from the point opposed to the triangle's basis, which is its longest edge. This point is considered the origin of the triangle's system of local coordinates. The axes u and v (see Fig. 3 (b)) are along the triangle's longest edge respectively along the triangle's height. Let a_1 and a_2 the distances from the origin to the triangle's vertices located on the same edge. d is the vector from the origin of the coordinate system to the point p. x and y are the coordinates of a point situated within the triangle. The distance from the point p to the point (x, y) of the triangle is:

$$d^2(r, S) = (d - (ux + vy))^2 = d^2 + x^2 + y^2 - 2(xu + yv)$$

and the integral can be written:

$$F(p) = \int_0^{h - \frac{hx}{a_2}} \int_0^{a_2} h(d(r, S))dxdy$$

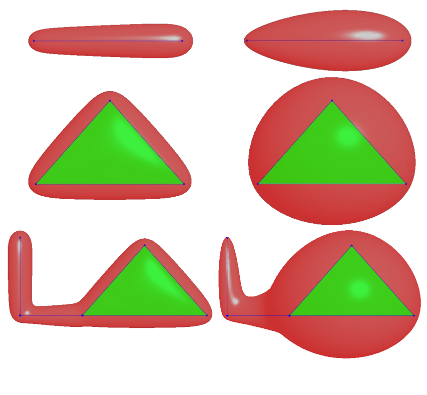

Figure 4: *Convolution with linearly varying radius segments and with triangles. Left: the pseudo Cauchy kernel. Right: the extended inverse kernel.*

The closed form solution of this integral is given in Appendix A. The number of necessary operations

Kernel	*	/	+	-	Other
Pseudo Cauchy	43	4	24	11	4 sqrt 6 atan
Inverse	32	5	28	7	3 sqrt 3log

Table 2: *Evaluation for triangle skeleton.*

compared to that necessary for the pseudo Cauchy function are given in Table 2.

3.4 Comparison

The previous subsections show that the weighted inverse kernel is faster to evaluate than the pseudo Cauchy kernel. However, this function must be used carefully, as it drops very abruptly, causing a pronounced and difficult to control blending (see Fig. 4). The pseudo Cauchy kernel drops slowly producing a much controllable blending and also has a tangent parameter s which allows even better blending control. Any of the two can be used in our reconstruction algorithm, but the inverse function sometimes induce errors in reconstruction due to its excessive blobbiness.

4 CSG operators

It is important to have a composition model that allows precise control of the blending result, especially since we do not use an optimisation step. The use of positive/negative functions composed by summation makes impossible the control the resulted surface because there is no mean to impose the surface to pass through a given point. The exact control can be achieved only with CSG composition. The model from [3] has the advantage of being general and also n-ary defined. We propose a CSG model that derives from it, but does not modify the field value even after hierarchical composition. Otherwise several field modification in the same region would dramatically affect the field value and the surface would become uncontrollable after several steps in the blending hierarchy tree. To achieve precise control, we apply the blending function only once to each primitive, in the hierarchical tree's leaves. A notable advantage of applying a blending function to an unbound function is that its influence in space will be limited, therefore speeding the function computation because there is no need to evaluate the function outside its radius of influence. The blending operator must be rewritten to be suitable for our kind of functions (positive inside the surface and negative outside the surface). We use a blending function of the Stolte type [15] defined as:

$$g(x) = \begin{cases} 0 & , x > R \\ 2 & , x < -R \\ \frac{1}{R^{mn}} (R^m - x^m)^n & , x \in [0, R] \\ 2 - \frac{1}{R^{mn}} (R^m + x^m)^n & , x \in [-R, 0] \end{cases} \quad (1)$$

$\forall m \in \mathbf{N}$, m odd, and $\forall n \in \mathbf{N}$, n even. See Fig. 5 for a plot of this function. The n parameter gives the sharpness of the blending. m is constant our case and $m = 1$. R is the function's influence radius. As stated in [3] R must be chosen so that $R = f(p)$ where p is a point that we want on the surface. The

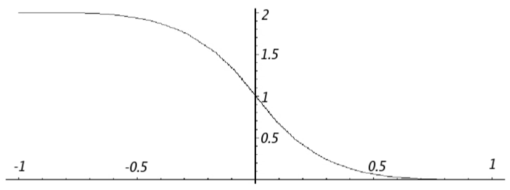

Figure 5: *Blending function.*

function's definition on the interval $[0, R]$ describes the primitive's behaviour regarding the CSG union, while the definition on the interval $[-R, 0]$ describes its behaviour regarding difference. With these conventions, the union between several leaves i.e. primitives that have never been blended can be written as:

$$F(p) = -1 + \sum_{i=0}^{n} g_i \left(-f_i \left(p\right) + T\right) \quad (2)$$

where f_i is the convolution function describing the i^{th} part, g_i is the blending function associated with the i^{th} part. The union operator between composed nodes and leaves is defined as:

$$F(p) = -1 + \sum_{j=0}^{m} \tilde{g}_\jmath \left(F_j \left(p\right)\right) + \sum_{i=0}^{n} g_i \left(-f_i \left(p\right) + T\right) \quad (3)$$

where $\tilde{g}_j = F_j(p) + 1$
The difference between leaves is given by:

$$F(p) = 1 - g\left(f_j(p) - T\right) - \sum_{i=0}^{n} g_i \left(-f_i \left(p\right) + T\right) \quad (4)$$

The difference between several trees and leaves is given by:

$$F(p) = 1 - \sum_{j=0}^{m} \tilde{g}_\jmath \left(F_j \left(p\right)\right) - \sum_{i=0}^{n} g_i \left(-f_i \left(p\right) + T\right) \quad (5)$$

where $\tilde{g}_j = -F_j(p) + 1$
In our case we do not need to subtract trees from leaves (the user does not need to design a complex shape and then subtract it from the object), so this case will not be treated here. Our model allows for a blending hierarchy that does not modify the field at the leaves of the tree, because the blending function is applied only once. As a result the operations are precise ensuring the smoothness of the resulted shape.

5 From the system's viewpoint

5.1 User input

The stroke's thickness and the colour's alpha component are linearly mapped to the pencil pressure to give the appearance of a carbon pencil drawing. This way, a higher pressure on the pen gives a thicker stroke and a more intense colour.
To provide the erasing operation while still being able to undo operations, we store every user drawn stroke within a stack. When rendering, if the stroke is an erasing one, drawing into the colour buffer is disabled. The stencil test is set to compare the fragment value to an ERASED marker. If the fragment fails the stencil test, then the stencil value is set to ERASED, otherwise it stays unchanged. This way, if a pixel is erased, all strokes that were covering it previously are not displayed. Still, they are all memorized. The *Undo* command erases the last stroke from the stack. If it is an erasing stroke, all the strokes drown underneath the erased stroke will become visible again. Painting is achieved by unprojecting the strokes from the window space into object space, and then displaying them with smooth brush-like points.

5.2 Skeleton from 2D contour

A pressure threshold indicates that the drawing is finished. When the stylus pressure has reached this threshold, the strokes image is recovered as a 2D bitmap. The image is then compressed and reduced in size using a pixel averaging technique, in order to smooth the input. This also reduces the amount of computation for the skeleton.
An overview of the reconstruction algorithm is given in Fig. 6. In order to perform the skeleton extraction we iteratively construct a connected pixels skeleton, invariant to object geometric transformations and allowing the inverse transform. The pixel skeleton is then sampled and the polygons are triangulated in order to obtain a segments and triangles graph. This will be used to define a convolution surface as in [2].
We detail all the algorithm steps for completeness.
1. The object is separated from the background (Fig. 6 (b)).
2. The Weighted Distance Transform (WDT) is computed (Fig. 6 (c)). This is done in two passes in a forward and backward scan, by applying the following operations to each pixel object P:

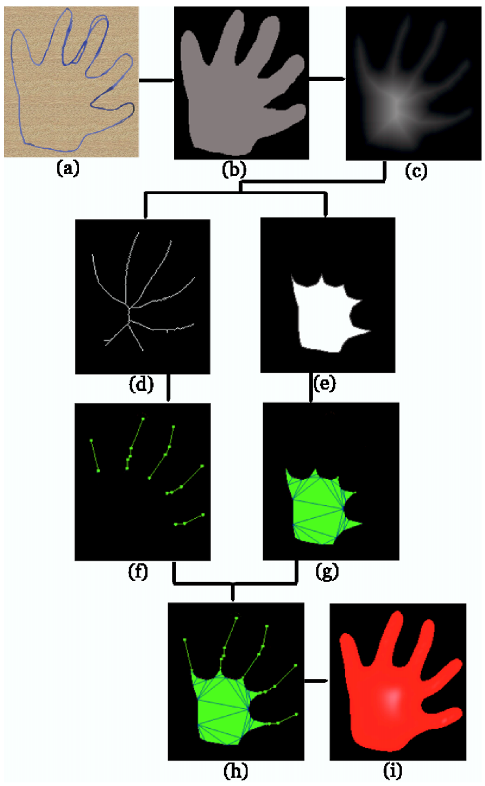

Figure 6: *Algorithm overview : (a) User strokes. (b) Identification of the object boundaries. (c) WDT transform. (d) Iteratively thinning with preservation of the CMDs. (e) Iteratively thinning without preservation of the CMDs. (f) Sampling the segments. (g) Sampling and triangulating the polygons. (h) Final skeleton graph. (i) Final convolution surface. Overall computation time : 1.7 s.*

Forward scan:

$$f_1(P) = min(N_5 + a, N_4 + b, N_3 + a, N_2 + b)$$

Backward scan:

$$f_2(P) = min(P, N_1 + a, N_8 + b, N_7 + a, N_6 + b)$$

where a and b are the metric weights (we use the $3-4$ metric) and N_i $i = 1..8$ are the pixel's neighbours. The result is shown in Fig. 6 (c). We then detect the set of Centres of Maximal Discs (CMDs) which will be used in skeleton extraction. A pixel P on the WDT is a CMD if $P + a > N_{2i-1}$ and $P + b > N_{2i}$ $i = 1..4$

3. The object is thinned iteratively keeping all the CMD's. This produces a connected pixel graph (Fig. 6 (d)), which is pruned to eliminate the insignificant branches. This graph will provide the skeleton segments.

4. The second pass is performed on the initial WDT image and thins the object without preserving the CMD, the result being an eroded shape of the object (Fig. 6 (e)). This image will be used to compute the skeleton's polygons.

5. The image from Fig. 6 (e) is subtracted from (d) and the result is a collection of free-form lines of one pixel width. Each line is sampled using the Douglas-Peuker compression algorithm [9] which produces a minimum number of segments with respect to a given minimum distance between the initial polyline and the sampling polyline (Fig. 6 (f)). The sampling polyline preserves the structure and peaks and avoids contour smoothing

6. To compute polygons, the object contour is then recovered from the eroded image (Fig. 6 (e)) using the turtle algorithm [13] and then sampled (Fig. 6 (g)). If the image has several disconnected object parts, then each part produces a polygon. The polygons are split into triangles using constrained Delaunay triangulation (Fig. 6 (g)).

7. The graph connections can now be computed from the two images in Fig. 6 (f) and (g). This produces the final skeleton graph (Fig. 6 (h)).

5.3 Creating a shape

The graph previously obtained defines a convolution surface which can be controlled by setting weights at every sampled point. The weights are assigned according to the distance from the point to the contour, which can be directly read using the WDT image. All points that are at the same distance from the contour will have the same weight. An exact fit of the user drawn contour would require an optimisation step, which might cause the surface to oscillate. We prefer to obtain an approximate fit of the user drawn contour and a smooth non-oscillating surface. For this purpose, we have fitted the weights manually for a large number of cases, considering the fact that they vary with the distance to the contour, but also with the distance from the user's point of view. We then computed a polynomial function which best interpolated all the found values and use it to automatically assign weights without any adjustment afterwards. This technique yields very satisfying results in practise. The implicit shape that reconstructs the given stroke is obtained by the convolution of the skeleton graph with a convolution kernel. A shape's capacity of blending with the previously existing shapes is described by its blending function from Eq. 1. We set the blending parameter n according to the level of detail. Small details produce sharper blending. The unblended surface is then polygonized and displayed. R can be taken to be the function's field value in any point issued from the latter polygonization. The overall time for performing the previous steps for the hand example in Fig. 6 is 1.7 sec., the most expensive step being the implicit surface polygonization [4]. The blending hierarchy is automatically updated.

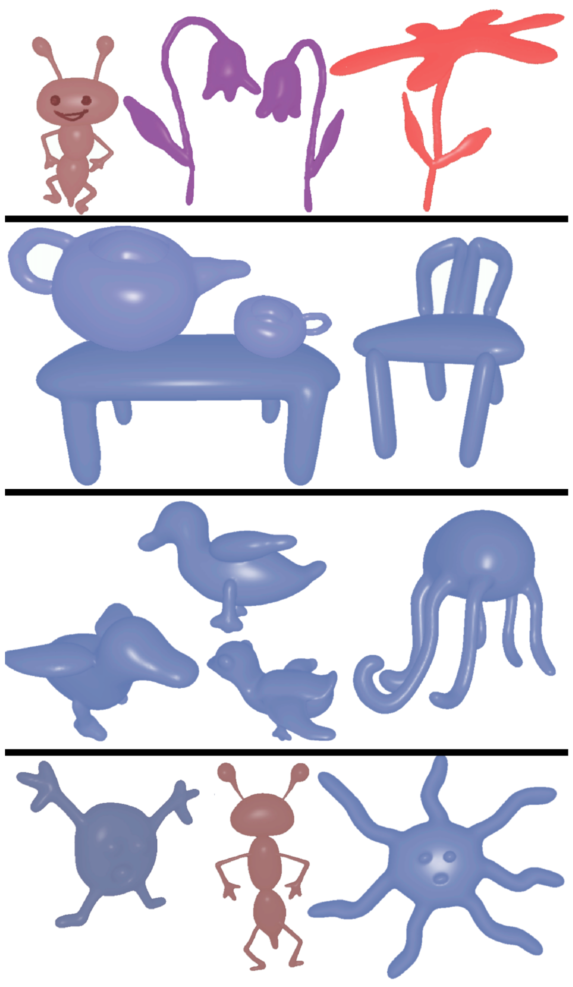

Figure 7: *Objects modelled with our system. The user took 2 to 5 minutes overall modelling time and 3 to 9 strokes for each object.*

5.4 Thickness control

The control of the object thickness orthogonal to the view direction is achieved by providing every implicit shape with a scaling factor. The function is composed with a scale in the depth direction. The scaling factor to be assigned to the current drawn shape, is computed from the sum of the two angles formed by the pen with X and Y tablet axes. The angle is measured when the pen been pressed to indicate reconstruction.

5.5 Adaptive modelling and rendering

The level of detail of the skeleton remains constant in the image space, but it is automatically adapted to the level of detail of the 3D shape. The latter is computed from the distance between the object and the camera. The level of detail determines the blending parameters, the skeleton weights and the size of the polygonization cell for the shape to be reconstructed. The polygonization for the previous reconstructed stroke is computed and displayed immediately, while a process in the background computes the final surface polygonization. Meanwhile the user continues his modelling task. If the polygonization is available, the final mesh is displayed, replacing the two meshes of the disconnected components. If the object has been updated, the polygonizing process is notified and it restarts the computation. This allows maintaining interactive rates and rapid application response during the modelling process so that the user feels free to pursue his modelling activity.

Object	Strokes	Overall time (min)
Table	3	2
Chair	4	2.5
Teapot	4	3
Mug	3	2
Ducks	3	2
Ants	9	5
Sun	4	2
Open flower	4	3
Closed flower	3	2
Octopus	7	3

Table 3: *Number of strokes and modelling time needed by the user to model the objects in Fig. 7.*

6 Results and conclusions

Convolution surfaces allow much better shape representation than standard skeleton based implicit surfaces, due to their possibility to represent flat surfaces, as well as a large topological variety. Fig. 7 shows some scenes with objects entirely modelled with our system. The system provides a real simplicity for the non-expert user, for example three strokes only are necessary to create each one of the birds (with symmetry enabled for the wings and legs). The figure also shows flat surfaces (table and chairs). The shapes have no oscillations and no bulges. The CSG composition is a generalised composition more flexible and accurate than the simple sum, allowing a better blending control, from smooth to sharp transitions. The small details of the objects are well preserved due to the parametrable CSG. For example, the sun's eyes and mouth are small details compared to the face but they are well preserved by the blending. The shape may have various topologies (ex. chairs, teapot) and can be carved (teapot, mugs). The applications of our system are educational, but also story boarding for films making (ex. cartoons, see Fig. 7) where the scenarios writer is not necessary a 3D designer and fun. These applications could be extended to the design the internal structure of organic shapes because the composition model is suitable for this. We would also like to focus on accelerating the polygonization time for generating the final implicit surface and we will investigate adaptive polygonization in the future.

A Appendix

Closed form solution for convolution of the extended inverse kernel with triangles.

$$\begin{aligned}
F(p) &= v log\left(\frac{t^2+v^2}{(a_1+u+A_1)(a_2-u+A_2)}\right)\\
&+ \frac{P_1}{Q_1} log\left(\frac{-h^2+a_1u+hu+Q_1W}{a_1^2+a_1u+hv+Q_1A_1}\right)\\
&+ \frac{P_2}{Q_2} log\left(\frac{-h^2-a_2u+hu+Q_2W}{a_2^2+a_2u+hv+Q_2A_2}\right)
\end{aligned}$$

where

$$\begin{aligned}
A_1 &= \sqrt{a_1^2+d^2+2a_1u}\\
A_2 &= \sqrt{a_2^2+d^2-2a_2u}\\
Q_1 &= \sqrt{a_1^2+h^2}\\
Q_2 &= \sqrt{a_2^2+h^2}\\
W &= \sqrt{h^2+d^2-2hv}\\
P_1 &= hu+a_1(h-v)\\
P_2 &= hu+a_2(h-v)
\end{aligned}$$

References

[1] A. Alexe, V. Gaildrat, and L. Barthe. Interactive modeling from sketches using spherical implicit functions. In *AFRIGRAPH*, Stellenbosch, November 2004.

[2] A. Angelidis and M.-P. Cani. Adaptive implicit modeling using subdivision curves and surfaces as skeletons. In *Solid Modelling and Applications*. ACM, june 2002. Saarbrucken, Germany.

[3] L. Barthe, V. Gaildrat, and R. Caubet. Combining implicit surfaces with soft blending in a csg tree. In *CSG Conference Series*, April 1998.

[4] J. Bloomenthal. An implicit surface polygonizer. In *Graphics Gems IV (P. Heckbert, ed.), Academic Press*, pages 324–349, 1994.

[5] J. Bloomenthal and K. Shoemake. Convolution surfaces. *Computer Graphics*, 25(4):251–256, 1991.

[6] D. Bourguignon, M.-P. Cani, and G. Drettakis. Drawing for illustration and annotation in 3d. In A. Chalmers and T.-M. Rhyne, editors, *Computer Graphics Forum*, volume 20 of *EUROGRAPHICS Conference Proceedings*, pages C114–C122. EUROGRAPHICS, Blackwell Publishers, sep 2001.

[7] A. Cuno, C. Esperança, P. R. Cavalcanti, R. F. Cavalcanti, and R. Farias. 3d free free-form modeling with variational surfaces. In *WSCG*, february 2005.

[8] B. de Vries. Sketching in 3d. In *18th Conference on Education in Computer Aide Architectural Design*, June 2000.

[9] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10 (2):112–122, 1973.

[10] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: A sketching interface for 3d freeform design. *Proceedings of SIGGRAPH 99*, pages 409–416, August 1999. ISBN 0-20148-560-5. Held in Los Angeles, California.

[11] O. Karpenko, J. F. Hughes, and R. Raskar. Free-form sketching with variational implicit surfaces. 21(3):585–594, Sept. 2002.

[12] D. F. Keefe, D. Acevedo, Moscovich, Tomer, Laidlaw, H. David, and J. J. LaViola. Cavepainting: A fully immersive 3d artistic medium and interactive experience. In *Proceedings of 2001 Symposium on Interactive 3D Graphics*. ACM SIGGRAPH, 2001.

[13] S. Papert. *Children, Computers, and Powerful Ideas*. Basic Books Inc., N.Y., 1980.

[14] A. Sherstyuk. Kernel functions in convolution surfaces: a comparative analysis. *The Visual Computer*, 15(4), 1999.

[15] N. Stolte. *Espaces Discrets de Haute Résolutions: Une Nouvelle Approche pour la Modélisation et le Rendu d'Images Réalistes*. PhD thesis, University of Toulouse III, 1996.

[16] C.-L. Tai, H. Zhang, and C.-K. Fong. Prototype modeling from sketched silhouettes based on convolution surfaces. *Computer Graphics Forum*, 23(4):855–83, 2004.

[17] B. Wyvill and K. van Overveld. Tiling techniques for implicit skeletal models. *Siggraph Course 11:C1.1-C1.26*, 3, 1996.

[18] R. Zeleznik, K. Herndon, and J. Hughes. Sketch: An interface for sketching 3d scenes. In *ACM Transactions on Graphics, Proceedings of SIGGRAPH 1996*, 1996.

[19] R. Zenka and P. Slavik. New dimension for sketches. In *SCCG 2003*, 2003.