Interactive Modelling from Sketches using Spherical Implicit Functions

A. Alexe
IRIT – Paul Sabatier University
118 route de Narbonne
31062 Toulouse 04, France
33 (0)5 61 55 83 29
alexe@irit.fr

V. Gaildrat
IRIT – Paul Sabatier University
118 route de Narbonne
31062 Toulouse 04, France
33 (0)5 61 55 83 29
gaildrat@irit.fr

L. Barthe
IRIT – Paul Sabatier University
118 route de Narbonne
31062 Toulouse 04, France
33 (0)5 61 55 74 31
lbarthe@irit.fr

ABSTRACT

We present an interactive modelling technique, which reconstructs three-dimensional objects from user-drawn two-dimensional strokes. We first extract a skeleton from the 2D contour, and the skeleton is used to define an implicit surface that fits the 2D contour. The reconstructed 3D shape has a natural aspect, it is very smooth and can easily be edited and modified using strokes or performing operations on the skeleton. This method is very accessible for non-specialist users and it allows fast and easy shape prototyping.

Categories and subject Descriptors:

I.3.5 [**Computer Graphics**] Computational Geometry and Object Modelling.

General Terms

Algorithms, Design.

Keywords

Geometric Modelling, Sketches, Implicit surfaces.

1. INTRODUCTION

When designing a three-dimensional object, one starts from a mental representation. By mental representation we understand a very high level description of the object, i.e. when the user wants to model a man he knows that he has to model four limbs and a body, etc. This mental representation has to be transcribed into the computer as a three-dimensional shape representation using some interaction device. Generally neither the interaction device nor the interaction metaphor of the modelling software are intuitive enough for non-expert users and the conception of the complete object remains a fastidious succession of shape creation and editing operations.

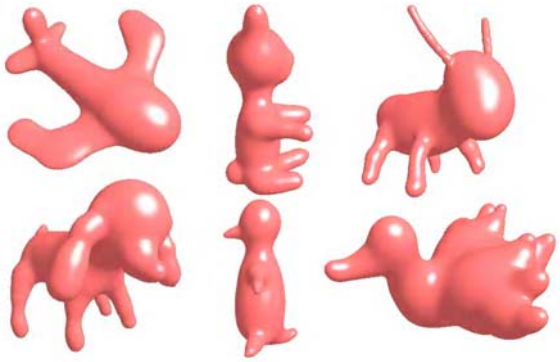

Figure 1. Examples of 3D objects designed with our system using 2D strokes, the average modelling time being 10 minutes per object.

From a general point of view the creation procedure can be decomposed into two distinct parts, which require different techniques and functionalities.

First of all, one has to build a prototype of the object i.e. a coarse approximation of the final shape. Second, the prototype is edited in order to accurately finalize the coarse representation and to add details.

Nowadays, several efficient multi-resolution techniques such as subdivision surfaces [52], [23] or normal meshes [17] allow the refinement of three-dimensional shapes once a coarse representation has been built. Concerning the prototyping procedure, actual modelling software can provide very advanced tools, allowing expert users to produce complicated and realistic prototypes, however the drawbacks are the following:

- They use complex and non-intuitive theoretical models (parametric patches, CSG, etc.). The user has to

follow a long and tedious learning process before being able to efficiently exploit these concepts.

- They use abstract notions that are too far from the real world (geometric primitives, numerical parameters, parametrical functions, key frames, etc).

The complexity of WIMP-like graphical interfaces (*Windows Icons Menus Pointer*), makes them difficult to use and increases the complexity of the user's task.

Since long time, experts showed [30], [43] that more time is spent searching through commands and menus than concentrating on the modelling task itself.

Classical modelling software still leaves unsolved some fundamental problems such as: how to quickly build mock-ups and prototypes; how to make the interaction with the user more intuitive and more natural; and finally, how to outpace the low level of abstraction.

However, we do not intent to provide a complete modelling tool, as it would be very difficult to compare our software with classical modelling software and its infinite number of modelling possibilities. We rather propose a complementing technique, which explores gesture based modelling of free-form shapes and which is mainly addressed to non-expert users.

Sketching is a simple means of expression, accessible by everyone (See [40] for a discussion on sketch-based interfaces.). It is useful for quickly materialising ideas and sharing them with the working team. It is also an excellent stimulant for creativity and innovation. For these reasons, several research projects have focused on the design of 3D shapes from 2D sketches. We will start by briefly presenting the related works on 3D shape reconstruction from a point set and then we will discuss the previous works on sketch-based modelling.

Related work on the 3D surface reconstruction: The problem can be formulated as it follows: starting from a set of points approximating a 2D contour, find a 3D closed surface with a suitable thickness which best fits the point set.

Related ideas on 3D shape reconstruction with implicit surfaces can be found in [27], [19], [8], [25], [4], [3], [31]. Other papers use globally supported RBF functions [41], [10], [44] or locally supported RBFs [28], [24], [32]. We note that all the cited papers are not meant to solve our problem, but the more general problem of 3D reconstruction from a 3D cloud of points. However they can be easily adapted to this particular case. Although RBFs seem to be the state of the art in this field, they require a large amount of data for an accurate smooth reconstruction and therefore they are not suitable for interactive use. Other inconvenients of this technique will be further discussed.

Related work on sketch-based modelling: The first modelling system that used sketches was introduced by Zeleznik [50]. SKETCH is based on gesture recognition and a "dictionary" of basic 3D primitives, and is limited to isothetic objects with sharp angles.

Teddy software [21] performs reconstruction using the chordal axis [38], from which a mesh is computed. The 2D contour is sampled regularly and the resulting planar set of points is triangulated using a constrained Delaunay triangulation [2]. A chordal axis [38], which connects the middles of the internal edges of the triangulation is then built and used as a skeleton. The final 3D shape is a polygonal mesh reconstructed by elevation of this skeleton. Teddy demonstrates the efficiency of 2D sketching for 3D free-form shape prototyping, but it also exhibits some limitations such as the difficulty to edit the mesh and to rearrange parts of the 3D object. The quality of the provided mesh is also poor, and a post-processing treatment on the mesh is necessary [20]. By using implicit surfaces (i.e. a smooth representation) we overcome these inconvenients.

Karpenko et al. [22] use variational implicit surfaces. [13] base their work on Karpenko's idea but add a more complete over-sketching utility. [51] enrich the method by the possibility to "paint" strokes on the surface or "in the air" to add more expressiveness. The surface is defined by functions that interpolate a set of points. In this context, the main problem is the extrapolation of the 3D shape thickness. The first task is to create a complete approximation of the 3D object by a set of points, which is needed by the interpolation function (the initial stroke is not sufficient). Hence, new points have to be created in both sides of the profile in order to build the 3D object thickness. This process is done by projection, which limits the form of the user-defined contours to ellipse-like shapes. The user creates several simple shapes, which are then blended in order to build the final object. The main limitation of this approach comes from the diminished range of the possible 2D contours, due to the difficult reconstruction of the 3D shape thickness. We note that the moving least squares techniques has the same drawback, so the same observations apply to it as for variational implicit surfaces. Also, in [22] the user is forced to explicitly blend the different parts of the object, whereas ideally this process would be automatic and transparent. Both of these drawbacks are avoided by our approach.

A more recent approach reconstructs a cylindrical convolution surface from the user's stroke [45]. Convolution cylindrical implicit surfaces are a smooth structure and they are very close to our approach. As we show in section 3.2.3, convolution cylinders are not adapted to reconstruct surfaces that fold onto themselves, as near the folding a visible unwanted blending appears that cannot be entirely removed by the optimisation process. Also, the authors report unwanted oscillations of the resulting surface, which are produced by the optimisation process, as it can be seen in their figures.

Owada et al. [33] use a volumetric voxel based structure, which allows them to model the intern object cavities. As done in [45], the same functionality can be realized with our approach, using negative implicit functions, and it will be developed as a future extension.

We point out that in the different methods presented so far, the user cannot control the local 3D shape thickness (except in [45]). Since the shape's thickness is computed automatically and may not correspond to the user's intention, this is an important issue.

Our approach and contributions: We prefer to keep the double representation skeleton/3D-shape and hence allow the 3D shape to be defined and modified using 2D strokes, but also edited using very simple operations on the skeleton, such as deleting, copy/cut-paste, etc. Our structure is similar to the one used in [36] to recover animation movements from video sequences. In our case there is a single image, which is the user's stroke, and we construct what would correspond to a "natural" shape from this image. The user has the possibility to modify it by various strokes and skeleton operators. Skeleton manipulations are of interest because they allow simple and intuitive rearrangements of the object's parts such as arms, legs, etc. For skeleton based editing see [49] Moreover, a smooth shape representation is preferable to a direct polygonal mesh extraction.

Therefore we use Teddy's skeleton reconstruction procedure (modified with an adaptive sampling of the 2D stroke that allows better detail capturing than uniform rough sampling techniques) and once the skeleton is extracted, we reconstruct the 3D shape with blended implicit spheres, placed along the skeleton, which fit the 2D contour. We also preserve the skeleton structure, as a graph, the nodes being the implicit spheres.

Our reconstruction technique is based on the one used by Muraki [27] and [8], but we make it interactive by performing several simplifications which will be exposed later. The reconstructed surface is very smooth (C^7 continuous in our case; the reasons why this high class continuity is necessary will be explained in section 3.2.2), the spheres radii automatically extrapolate the 3D shape thickness and the user can edit it once the 3D shape is reconstructed, and finally simple editing operations can be performed on the skeleton.

This approach is, as the previous sketch-based approaches, suitable for modelling simple, organic-like shapes, and it can be used by non-expert users, since it does not require knowledge of the underlying surface model. Only the skeleton and the stroke tool (a digital pen, a mouse, etc.) are available to the user. It can be applied in fast prototyping and modelling for story telling. The computing times are about 2-3 seconds per stroke, with about a few dozens blobs per stroke, which is the case for all the models that we show in Figure 1. Therefore, the reconstruction times are comparable to those obtained by our predecessors, but the quality of the surface is much better (i.e. smoother and less oscillations), as we produce high-class continuity surfaces, which can be compactly stored, as a set of blobs parameters. We propose a new spherical kernel, which is well suited for our reconstruction needs. We also enriched the modelling tools with skeleton manipulation operators. Our method overcomes the limitations imposed to the input contour in [22]. Compared to cylindrical convolution functions (as used in [45]), spherical implicit functions have a smaller influence zone, and therefore they follow more faithfully the shape of the stroke, and the surface remains smooth. Besides they are faster to evaluate than cylindrical convolution functions. We also propose a technique to considerably reduce the oscillations of the resulted surface.

Paper organisation: In the next section, the adaptive 2D stroke sampling and the skeleton extraction procedure are presented. Section 3 describes the automatic reconstruction of the 3D implicit surface that fits the 2D contour and Section 4 presents the different creation/editing operations available with our system.

2. Sampling and extraction of the skeleton from the 2D stroke

First the user sketches a stroke, which is automatically closed. Then we apply an average filter, in order to remove noise from the input. In previous approaches, the contour can be then sampled following two strategies: whether the edges have uniform length ([21], [22]), or the edges length is adaptively established, between a minimum and a maximum value, whenever the angle variation is smaller than a threshold value [45]. Another solution for adaptive sampling is the popular Douglas Peuker algorithm for line simplification [14]). But the Douglas Peucker algorithm would produce a highly non-uniform repartition of samples along the contour, which will have undesirable effects on the reconstruction.

We think that an adaptive sampling technique is recommended, since it helps to capture the coarse details in the strokes and to skip the finer ones. At the same time, we need to guarantee a minimum uniformity of the samples repartition, which will result in placing the implicit spheres close enough to form a stretched smooth blending.

For these reasons, we prefer to start from a regular sampling, and then adapt it wherever this is necessary, in order to remove the oscillations in the resulting surface.

We used a wavelet-based compression of the stroke, at several levels. To compress one level, we replace every two points by one point, its coordinates being the average of the two points coordinates. The process is repeated several times (4 times in our case, which means that every 16 pixels in the contour produced a point in the sampling polygon). The coordinates are stored in tree structures, every point memorizing the two points that produced it. The leafs of the trees are the pixels of the initial contour, and the roots are the maximum compression level.

Whenever one sample is considered not sufficient, the corresponding tree expands, and the two "sons" replace the initial point. To determine if the sampling is sufficient enough, we have to compute a first approximation of the reconstructed surface, and to evaluate it. The process is described in section 3.2.2. For our computations, we start with the maximum level of compression.

The next step is the construction of a constrained Delaunay triangulation [2] (Figure 3 (b)) of the polygon points (Figure 2 (a)). This is followed by the computation of the polygon skeleton, using the chordal axis [38] (Figure 2 (c)). This is a close relative of the medial axis [11] but it is locally defined and it allows pruning of insignificant branches. We recall that the chordal axis links the middles of the internal edges of the Delaunay triangulation. A special treatment is applied to

branching points. Our skeleton extraction algorithm and pruning are based on the ones described in [21].

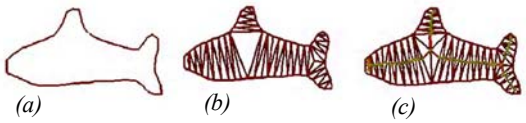

Figure 2. (a) Initial contour. (b) Delaunay triangulation. (c) Skeleton computation.

3. RECONSTRUCTION OF THE 3D IMPLICIT SURFACE

3.1 Theoretical background

The implicit surface that reconstructs the 3D shape is generated by the blend of implicit spheres. A single implicit sphere is defined by its centre c_i. Then by setting $r_i = d(p,c_i)$, the distance from a point $p \in R^3$ to the centre c_i, we define a potential function $f_i(r_i)$. Functions f_i decrease smoothly following a Gaussian-like curve, from 1 to 0, as r_i varies from 0 to infinity (Figure 3). An influence radius R_i bounds the function's contribution. When $r_i \geq R_i$: $f_i(r_i)=0$ (bounded primitives) or $f_i(r_i) \approx 0$ i.e. $f_i(r_i) < \varepsilon, \forall r_i \geq R_i$ with ε negligible. The sphere's surface is defined by the set of points $p \in R^3$ such that $d(p,c_i)=e_i$ and $f_i(e_i) = C$, where C is a constant within the interval (0,1), generally 0.5.

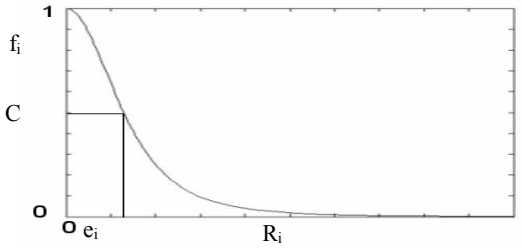

Figure 3. Plot of a Gaussian-like potential function fi used for our reconstruction.

The volume bounded by the sphere is the point set of R^3 such that $f_i(r_i) \geq C$. The most important property of these implicit spheres is their capacity to automatically blend when their potentials are summed (N is the number of spheres):

$$f(p) = \sum_{i=0}^{N-1} f_i\ (p) \qquad (3.1)$$

The result of the blending is a smooth surface defined by the potential function f. The first implicit model of this type was the blobby model [5], which is based on an exponential function. In order to reduce the computational cost of the exponential and to provide bounded primitives, several polynomial functions f_i have been proposed: SoftObjects [47], Metaballs [29], W-shaped polynomial [37], the "pseudo Cauchy" function [39], Stolte's function [42], keR model [8], etc. We denote these spherical potential functions f_i as "blobs".

Other powerful modelling by blending techniques exist, such as the blob-tree [46]. Our structure might be regarded as a single n-node from a blob-tree, the operator being the summation of the primitives. The F-Rep [35] are more general implicit functions but they are very complex to evaluate and therefore not suitable for interactive modelling.

3.2 Reconstruction and fitting procedure

3.2.1 First surface approximation

Once the skeleton has been computed, blobs are placed in every skeleton point as illustrated in Figure 4. The edges of the blobs graph are automatically computed from the skeleton's structure. We now have to choose the potential functions f_i. For reasons that are discussed in detail in the next section, we propose the following kernel:

$$f_i\ (p) = \frac{\left(r_i^2 - R_i^2\right)^8}{R_i^{16}} \qquad (3.2)$$

The reconstructed 3D surface is then defined by the following equation:

$$f(p) = \sum_{i=0}^{N-1} f_i\ (p) = \sum_{i=0}^{N-1} \frac{\left(r_i^2 - R_i^2\right)^8}{R_i^{16}} = C \qquad (3.3)$$

This equation has a set of N parameters R_i (one influence radius R_i per blob f_i) that have to be determined. These parameters have to be computed so that the 3D surface best fits the drawn stroke. We cannot directly determine the parameter values for which the 3D surface is sufficiently close to the contour, but we can compute a first approximation and then use an adjustment (minimisation) procedure in order to converge to an optimal solution.

The first approximation is computed by considering each blob as being isolated. The equation that defines a single blob with a radius e_i is written as:

$$f_i\ (p) = \frac{\left(e_i^2 - R_i^2\right)^8}{R_i^{16}} = C \qquad (3.4)$$

C being the same constant as in (3.3) and e_i being a fixed value that represents the radius in isolation of the i^{th} blob (i.e. the distance from the blob's centre to the surface). We consider the radius e_i as being the average of the distances between the blob's centre c_i and its neighbouring contour points (see Figure 4). The radius e_i set in this way in equation (3.4) allows us to determine the initial value of the parameter R_i. This process is applied on each blob and we obtain the initial potential function f.

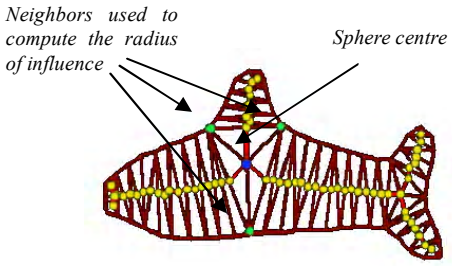

Figure 4. Computing the initial values for the radius of influence Ri.

With parameters R_i computed as described above, one could notice a slight "bloating" effect on the surface, which moves away from the contour, as shown in Figure 5 (a).

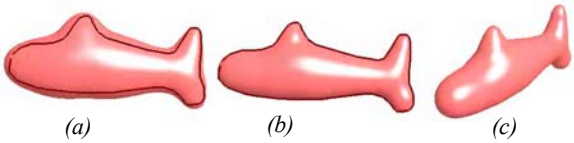

Figure 5. (a) Initial surface. (b) Surface after adjustment. (c) The same object viewed from a different angle.

This is natural since the influence radius of each blob is computed without considering the influence of the neighbouring blobs. When all the blobs' contributions are summed (Eq. (3.3)), the potential values of functions f_i are accumulated and as the blobs blend the surface begins to bloat. Moreover, blobs are often very close, and the "bloating" effect can be quite significant. Our experiments show that a better first approximation of the contour by f is obtained by multiplying the radius e_i by a factor of 2/3. By reducing the initial radii we reduce this bloating effect, and we win a few steps in the minimisation process, hence speeding up the process.

We notice that the distance between two neighbouring blobs never exceeds the sum of their radii and hence, they are guaranteed to blend.

3.2.2 Discussion on the smoothness of the resulting surface

Two main criteria have been taken into account to guide our kernel choice: the less oscillations of the reconstructed surface (Figure 6), and the low computational cost of the potential function.

C^1 or even C^2 continuity proved not to be sufficient for generating a perfectly smooth surface. As the Figure 6 (b) demonstrates, small oscillations are visible on the fish's tail, which mark the transitions between blobs.

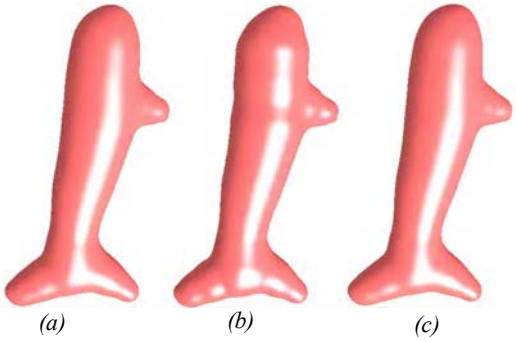

Figure 6. (a) Stroke reconstructed with blobby model [5](C^∞ function). (b) Stroke reconstructed using Soft Object function [47] (C1 function). (c) Stroke reconstructed using our Stolte-like function (C7).

This happens when the potential function "falls" too abruptly to zero. The oscillations vanish when the function's continuity class increases (i.e. the potential function falls less rapidly). Our experiments produced very smooth surfaces when using computationally expensive unbounded C^∞ functions (Figure 6 (a)) and less smooth surfaces (Figure 6 (b)) when using computationally cheap C^a continuous bounded polynomial functions ($a \leq 2$). A function providing a good compromise would have a high-class continuity, while remaining computationally cheap. This has been obtained with a bounded function (Eq. 3.2) similar to the W-shaped or Stolte's function, but with a higher degree to achieve a better smoothness (C^7 continuous) and therefore less oscillation.

As it can be observed in Figure 6, there is no visible difference between a surface generated using the blobby model, and a surface generated with our function. This is the first solution that we propose for smoothing the surface. However when the surface has tiny cylindrical regions (Figure 7), or small details, this might not be enough.

Another solution would be simply to sub-sample the stroke, in order to get a higher number of contour points, which would generate a higher number of blobs. But this solution adds more blobs everywhere on the stroke, when actually only some regions need more blobs. In this case we use a third solution.

We consider that two blobs are too far from each other if the distance between their centres exceeds the minimum of the two blobs radii, multiplied by an adjusting factor (which is 1.2 in our case). In this case the sampling contour points in that zone are expanded, every point being replaced by the two points on the superior tree level that we used for compression (see section 2). In that way we refine all the zones were blobs are not close enough. This process is done before parameters adjustment. If the trees have been expanded, the skeleton and the first approximation of the implicit spheres need to be recomputed. This process takes less that one second, and it is repeated a number of times that cannot exceed the maximum level of compression (the average is two times). This maintains the modelling time interactive.

In Figure 7 (c) the surface exhibits oscillations in tiny regions, i.e. where the blobs are not close enough to form a stretched blending. In this case the minimisation process fails to remove the oscillations and hence we use our blobs insertion procedure. After the adjustment of the local blob density, 32 contour points have been added, in less than 2 seconds and the oscillations on the surface have been considerably reduced.

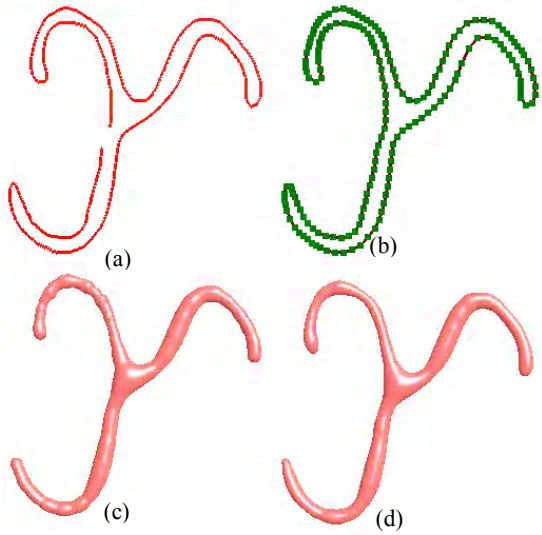

Figure 7. Smoothening the surface: (a) Initial stroke. (b) Stroke adaptively sampled by expanding the sampling trees in the tiny zones. (c) Surface reconstructed with the regular sampling. (d) Surface reconstructed with the adaptive sampling. Note the reduction of the shape oscillations.

The next step is the minimisation process, which adjusts the parameters R_i so that the 3D implicit surface fits the contour.

3.2.3 Adjusting the function parameters

As in previous approaches [27], [8], [45], the adjustment tries to minimise an energy function E that characterises the distance between the surface and the contour points. This is done by summing the squares of the difference between the value of the function f at contour points and the value C that f should return at these points in order to interpolate them. M is the number of contour samples:

$$E = \sum_{j=0}^{M-1} \left(f(p_j) - C \right)^2 \qquad (3.5)$$

To perform the adjustment, several non-linear least squares minimisation methods (Gauss-Newton, Levenberg-Marquart) have been tested. The best performance in both speed and convergence were obtained using the dogleg trust region method [26]. Only two seconds are necessary on an AMD Athlon 1,3GHz for the method to find an almost zero energy value (i.e. smaller than 0.1 which is sufficient in our case) when the number of blobs is around fifty. This number of blobs is largely enough for the models that we present. Every stroke is reconstructed individually, as the modelling process is incremental. The surface before and after the adjustment is shown in Figure 5.

A more complete energy function would also contain a tension term within the energy formula [12], which would minimize the mean curvature over the contour points. However, this would considerably slow down the minimisation process (i.e. by a factor of 100 in our tests), and it would not be suitable for interactive times. As our experiments showed, it is not necessary to use a more complete energy formula.

Finally the surface is polygonized for rendering and displayed using a classical Marching Cubes algorithm [6]. Figure 8 compares the mesh obtained with our reconstruction algorithm with the one produced from the same contour using the algorithm from Teddy [21].

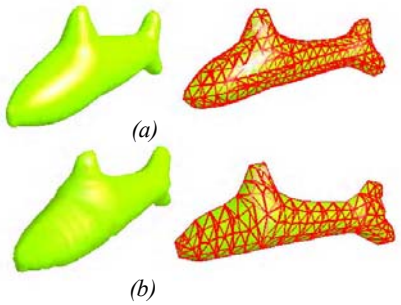

Figure 8. (a) Implicit surface produced by our algorithm. (b) Mesh produced with the algorithm described in [21]. The resultant surface is much smoother when using implicit surfaces.

One can see that our method provides a much smoother surface. A post processing of the mesh (for example, with subdivision surfaces) would also be an option, but in that case we will loose the compact structure of the implicit surfaces, and the mesh should be post processed every time the skeleton operators are applied, which is not convenient.

As we stated in the introduction, the reconstruction with convolution cylinders is not suitable for "folding" strokes. This is shown in Figure 9. We use the same convolution function as in [45], and we compare the stroke reconstructed using their method with the same stroke reconstructed with blobs. In order to remove the unwanted blending at the interior of the fold, the contribution from the cylinders in this zone is diminished, but that causes the surface to move away from the stroke at the extern part of the fold. This is the reason why we preferred to use blobs because they have smaller regions of influence, so they follow faithfully the curve of the fold, as shown in Figure 9 (b) and (d).

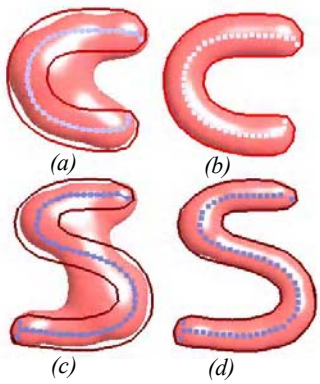

Figure 9. (a), (c) Folding stroke reconstructed with convolution cylinders as described in [45]. (b), (d) The same folding stroke reconstructed with blobs (the Stolte-like function 3.2)

4. EDITING OPERATIONS

The double shape representation (skeleton/3D-shape), gives some advantages to our method. Both of these structures are intuitive to use for shape editing. In addition to the editing possibilities provided by the use of 2D strokes, the user can perform simple operations directly on the skeleton points.

Figure 10 synthesizes most of the operations implemented so far. The automatic blending property of implicit surfaces preserves the surface smoothness and the definition of the surface by blended spheres ensures that the object is and remains solid through editing operations. Some images of objects modelled with our system are shown in Figure 1. Each object was modelled within 10 minutes average time.

4.1 Extrusion

In order to perform an extrusion, the user must first select the blob to be extruded (dark point in Figure 10 (b)). The next step is to draw the extrusion profile by sketching a stroke that is not necessarily closed, as shown in Figure 10 (b). The system then processes the stroke as in creation mode. The skeleton of the stroke is extracted and implicit spheres that approximate the stroke are produced. Finally, the newly created skeleton is connected to the existing one, and the user sees the extruded part that he created.

4.2 Changing the thickness of one or more blobs

In order to modify the thickness of one or several blobs the user first selects the centres of the blobs to be scaled using a selection rectangle (dark points in Figure 10 (c)). Then he moves the mouse or the digital pen up or down, depending on the desired effect: flattening or fattening. Inserting a factor in the squared Euclidean distance computation does scaling.

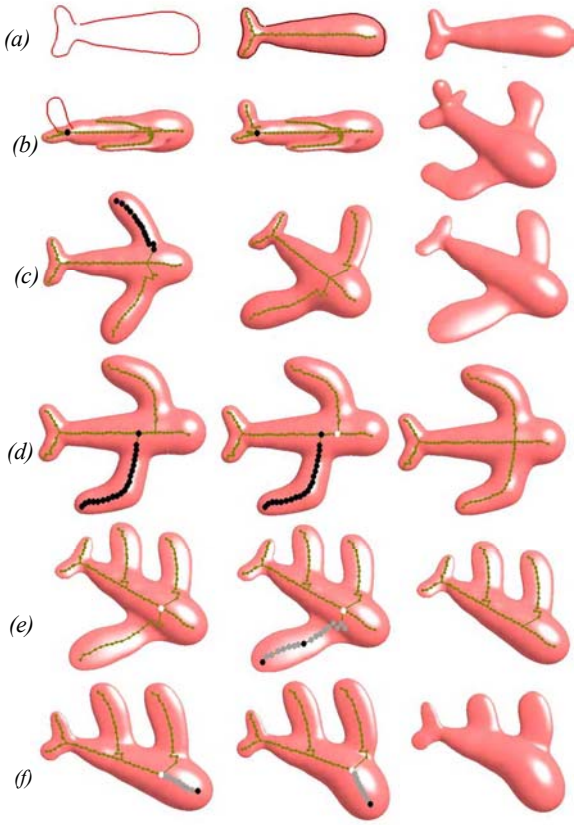

Figure 10. Operations implemented by our system: (a) Creation of a shape from its profile. (b) Creating the plane's tail by extrusion. (c) Flattening the wings and the tail by scaling the blobs. (d) Cut-paste of a skeleton part. (e) Suppression of a skeleton part. (f) Rotation around an articulation point.

For example, in order to perform scaling on the z axis (when the contour is defined in the (x,y) plane), the squared distance $r^2=x^2+y^2+z^2$ is replaced by $r^2=x^2+y^2+\gamma z^2$, where γ is the scaling factor. In order to flatten, γ is chosen to be less than one and for fattening, γ is chosen to be greater than one. When $\gamma \neq 1$ the spherical blob becomes an ellipsoid, hence providing the thickness control. Examples are shown in Figure 10 (c). As done in [45] we can think of using anisotropic distance functions [47], [34], [7], [10]. However, this obliges the user to draw the thickness for a set of blobs and we are not sure that this is an efficient tool to provide. Since we want our modelling metaphor to remain as simple and as intuitive as possible, the control of the shape's thickness in a more appropriate manner will be the topic of our future research.

4.3 Rotation around an articulation

The user selects first the point on the skeleton, which will be the rotation pivot (white point in Figure 10 (f)). Since the skeleton is a connex graph, there are at least two possibilities for the selection of the part of the skeleton to be rotated. The

second selected point identifies the part to be rotated around the pivot (dark point in Figure 10 (f)). The selected branch can now be interactively oriented in the desired direction. All the blobs located in this area (grey points Figure 10 (f)) go through the same rotation transform. The property of automatic blending of the implicit surface guarantees that the smooth surface aspect will be preserved through this transform. In order to avoid the unwanted blending (for example, an arm brought too close to the body will blend with the body) we use the technique described in [18], i.e. we start by computing the contribution from the closest blob, then we progressively add the contribution of this neighbours, and the process is repeated recursively until the contribution becomes negligible This does not work for the folding surfaces in Figure 9 (so it would still not solve the unwanted blending that appears when using the technique from [45]).

4.4 Copy-paste

First the user has to select the skeleton branch to be copied, doing the same procedure as for rotation. In order to paste the blobs at another location, the user clicks on the blob in which he wishes to paste the branch, as shown in Figure 10 (d) (white point). The copied branch will be pasted in that position, i.e. all the blobs on the branch will be copied, and then translated with a vector, equal to the difference between the old branching point and its new paste location. In the example shown, the user performed a cut-paste operation. The skeleton graph is then updated with the new connection. Then the new branch can be reoriented in a new direction (for instance, symmetrical with the old branch, to form symmetrical arms, legs, ears, etc.).

4.5 Suppression of a skeleton part

For this operation, the selection is done in the same way as for rotation: first the pivot is selected, then the user clicks any point on the branch that he wants to select (see Figure 10 (e)). This identifies all the blobs to be deleted. Pressing the « delete » key suppresses all the selected blobs, except the pivot. The skeleton graph is updated.

5. CONCLUSION AND PERSPECTIVES

We presented a system for modelling 3D shapes from 2D user-sketched strokes. Using the Igarashi et al. skeleton extraction procedure [21] we propose reconstructing the 3D surface in a different manner. Instead of directly extracting a 3D mesh, we use the skeleton to place implicit spheres, which are automatically blended in order to reconstruct a smooth surface. This surface can then be polygonized for interactive rendering. Our method takes advantage of its double representation: the 3D shape and the skeleton. It provides the same functionalities that previous approaches have and in addition it provides a smoother, compact structure surface and it offers new editing tools such as thickness control of the 3D shape, (copy/cut)-paste operation on the skeleton, etc. It becomes possible to rearrange parts of the object without re-sketching them, and our modelling tool remains very simple and accessible to anyone while providing an efficient shape prototyping method.

The shape can be compactly stored using the spheres' centres and radii, and it can be polygonized with the desired resolution, in order to be visualized or to be exported to other modelling software where small details can be added.

Our model is compatible with the general mesh format provided by classical modelling software and it could be integrated into a classical modelling frame using for example the HybridTree approach [1].

The main drawback of our approach is the difficulty in representing sharp edges (i.e. they are smoothed by the implicit function reconstruction).

Perspectives include the implementation of an adaptive and incremental polygonization algorithm, in order to repolygonize only the parts that have been modified [16], [15]. Now that we have demonstrated the efficiency of our reconstruction approach, we will develop more editing operators in order to provide a full panel of simple and intuitive tools that can be supported by our shape representation.

6. REFERENCES

[1] Allègre R., Barbier A., Galin E., Akkouche S. *A Hybrid Shape Representation for Free-form Modeling.* Research Report, Liris, Lyon University, 2004, RR-2004-009.

[2] Aurenhammer F. *Voronoi diagrams - A survey of a fundamental geometric data structure.* ACM Computing Surveys, 1991, 23: pp. 345-405.

[3] Bernadini F., Bajaj C. L., Chen J., Schikore D.: *Automatic reconstruction of 3D CAD models from digital scans*. International Journal of Computational Geometry & Applications *9*, 4, pp. 327–369.

[4] Bajaj C. L., Bernadini F. Xu G. *Automatic reconstruction of surfaces and scalar fields from 3D scans*. Proceedings of ACM SIGGRAPH 95, pp.; 109–118.

[5] Blinn J. F. *A Generalisation of Algebraic Surface Drawing*. ACM Trans Graphics, Vol. 1, No. 3, July 1982, pp. 235-256.

[6] Bloomenthal J. *An Implicit Surface Polygonizer.* Graphics Gems IV (P. Heckbert, ed.), Academic Press, New York, 1994, pp. 324-349.

[7] Blanc C., Schlick C. *Extended Field Functions for Soft Objects*. Proc of Implicit Surfaces'95 pp. 21-32.

[8] Bittar E., Tsingos N., Cani M.-P. *Automatic Reconstruction of Unstructured 3D Data: Combining a Medial Axis and Implicit Surfaces.* Computer Graphics Forum, 14(3): pp. 457-468, August 1995.

[9] Carr J. C., Beatson R. K., Cherrie J. B., Mitchell T. J., Fright W. R., McCallum B. C. *Reconstruction and representation of 3D objects with radial basis functions*. Computer Graphics (SIGGRAPH 2001 proceedings), pp. 67-76, August 2001.

[10] Crespin B., Blanc C., Schlick C. *Implicit Sweep Objects.* Eurographics' 96, 15(3), pp.165-174.

[11] Choi H. I., Choi S. W., Moon H. P. *Mathematical Theory of Medial Axis Transform*. Pacific Journal of Mathematics, Vol. 181, No. 1, pp. 57-88, November 1997.

[12] Desbrun M., Cani M.-P. *Active Implicit Surface for Animation*. Graphics Interfac, June 1998, pp. 143-150.

[13] De Araujo B., Jorge J. *BlobMaker: Free-form modelling with variational implicit surfaces*. Proceedings of 12º Encontro Português de Computação Gráfica, Porto, 2003 pp. 17-26.

[14] Douglas, D.H., Peucker, T.K. *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.

[15] Ferley E., Cani M.-P., Gascuel J.-D. *Practical Volumetric Sculpting*. The Visual Computer no. 8 vol. 16 pp. 469-480 , December 2000.

[16] Galin E., Akkouche S. *Incremental Techniques for Implicit Surface Modeling*. Computer Graphics International'98 312-321, June 1998.

[17] Guskov I., Vidimce K., Sweldens W., Schröder P. *Normal Meshes*. Computer Graphics Proceedings (SIGGRAPH 2000), pp. 95-102, 2000.

[18] Hornus S., Angelidis A., Cani M.-P. *Implicit Modelling Using Subdivision-curves*. The Visual Computer, 2-3 (19), pp. 94-104, May, 2003.

[19] Hoppe H., DeRose T., Duchamp T., McDonald J., Stuetzle W. *Surface reconstruction from unorganised points*. Proceedings of ACM SIGGRAPH 1992, pp. 71–78.

[20] Igarashi T., Hughes J. F. *Smooth Meshes for Sketch-based Freeform Modeling*, *ACM Symposium on Interactive 3D Graphics*. ACM I3D'03, 2003, pp. 139-142.

[21] Igarashi T., Matsuoka S., Tanaka H. *Teddy: A Sketching Interface for 3D Freeform Design*. ACM SIGGRAPH 1999, pp. 409-417.

[22] Karpenko O., Hughes J. F., Raskar R. *Free-form Sketching With Varational Implicit Surfaces*. Eurographics 2002, TR2002-27, June 2002.

[23] Kobbelt L., Campagna S., Vorsatz J., Seidel H.-P. Interactive multi-resolution modeling on arbitrary meshes. SIGGRAPH 98.

[24] Kojekine N., Hagiwara I., Savchenko V. *Software tools using CSRBFs for processing scattered data*. Computers & Graphics 27, 2 (April 2003).

[25] Lim C., Turkyyah G. M., Ganter M. A., Storti D. W. *Implicit reconstruction of solids from cloud point sets*. Proceedings of the third ACM symposium on Solid Modeling and Applications, ACM Press, 1995, pp. 393–402.

[26] Mizutani E. *Powell's dogleg trust-region steps with the quasi-Newton augmented Hessian for neural nonlinear least-squares learning*. Proceedings of the IEEE International Conference on Neural Networks, Washington, DC, 2, pp. 1239-1244.

[27] Muraki S. *Volumetric shape description of range data using blobby model*. Computer Graphics, 25(4): 227-235, July 1991.

[28] Morse B. S., Yoo T. S., Rheiengans P., Chen D. T., Subramanian K. R. *Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions*. Shape Modeling International 2001, 89–98.

[29] Nishimura H., Hirai M., Kawai T., Kawata T., Shirakawa I., Omura K. *Object Modelling by Distribution Function and a Method For Image Generation*. The Transaction for the Institute of Electronics and Communication for Engineers of Japan, 1985, Vol. J68-D, Part 4, pp. 718-725.

[30] Norman D. A. *User Centred System Design*. Lawrence Erlbaum Associates, Inc., 1986.

[31] Ohtake Y., Belyaev A., Alexa M., Turk G., Seidel H.-P.: *Multi-level Partition of Unity Implicits*. ACM TOG (Proc. SIGGRAPH 2003), 22(3): pp. 463-470.

[32] Ohtake Y., Belayev A. G., Seidel H.-P. *A multi-scale approach to 3D scattered data interpolation with compactly supported basis functions*. Shape Modeling International 2003.

[33] Owada S., Nielsen F., Nakazawa K., Igarashi T. *A Sketching Interface for Modeling the Internal Structures of 3D Shapes*. Proceedings of 3rd International Symposium on Smart Graphics, Springer, Heidelberg, Germany, July 2-4, 2003, pp.49-57.

[34] Parent R., SINGH K. *Polyhedral shapes as general implicit surfaces primitives*. OSU-CIS Technical Report, OSU-CISRC-5/94-TR24, 1994.

[35] Pasko A., Adzhiev V., *Function-based shape modeling: mathematical framework and specialized language. Automated Deduction in Geometry*. Lecture Notes in Artificial Intelligence 2930, Ed. F. Winkler, Springer-Verlag, Berlin Heidelberg, 2004, pp. 132-160.

[36] Plänkers R., Fua P. *Articulated Soft Objects for Video-based Body Modeling*. International Conference on Computer Vision, Vancouver, Canada, July 2001, pp. 394-401.

[37] POV-Ray , http://www.povray.org/

[38] Prasad L. *Morphological analysis of shapes*. CNLS Newsletter, 139: 1-18, July 1997.

[39] Shersyuk A. *Convolution surfaces in Computer Graphics*. Ph.D. dissertation, School of Computer Sciences and Software Engineering, Monash University, Australia 2000.

[40] Sanchis N., Jorge J. A. *Direct Modelling: from Sketches to 3D Models*. 1st Ibero-American Symposium on Computer Graphics, July 2002.

[41] Savchenko V. V., Pasko A. A., Okunev O. G., Kunii T. L. *Function representation of solids reconstructed from scattered surface points and contours*. Computer Graphics Forum 14, 4, pp. 181–188.

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

[43] Suchman S.A. *Plans and Situated Actions: The Problem of Human-Machine Communication*. Cambridge Univ. Press. 1987.

[44] Turk G., O'Brien J. *Modelling with implicit surfaces that interpolate*. ACM Transactions on Graphics 21, 4 (October 2002), pp. 855–873.

[45] Tai C.-L., Zhang H., Fong C.-K. *Prototype Modeling from Sketched Silhouettes based on Convolution Surfaces.* Computer Graphics Forum, 2004. *To appear.*

[46] Wyvill B., Galin E., Guy A. *The Blob Tree, Warping, Blending and Boolean Operations in an Implicit Surface Modeling System.* Implicit Surfaces' 03, June 1998.

[47] Wyvill B., McPheeters C., Wyvill G. *Data Structure for Soft Objects*. The Visual Computer, Vol.2, No. 4 August 1986a, pp. 227-234.

[48] Wyvill B., Wyvill G. *Field functions for implicit surfaces.* The Visual Computer, 5:75 pp. 75-82, December 1989.

[49] Yoshizawa S., Belyaev A., Seidel H.-P. *Free-form Skeleton-driven Mesh Deformations*. Solid Modeling 2003.

[50] Zeleznik R., Herndon K., Hughes J. *Sketch: An Interface for Sketching 3D Scenes.* Proceedings of SIGGRAPH'96, 1996, pp. 163-170.

[51] Zenka R., Slavik P. *New Dimensions For Sketches.* Spring Conference on Computer Graphics 2003 Proceedings, Budmerice, Slovak Republic, 2003.

[52] Zorin D., Schröder P., Sweldens W. *Interactive Multiresolution Mesh Editing*. Computer Graphics (SIGGRAPH '97 Proceedings), pp. 256-268.