Matisse : Painting 2D regions for Modeling Free-Form Shapes

Adrien Bernhardt, Adeline Pihuit, Marie-Paule Cani, Loïc Barthe

EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2008)
C. Alvarado and M.- P. Cani (Editors)

Matisse : Painting 2D regions for Modeling Free-Form Shapes

A. Bernhardt[1], A. Pihuit[1], M. P. Cani[1], L. Barthe[2]

[1]Grenoble Universities (LJK-CNRS) & INRIA, France
[2]University of Toulouse (IRIT-CNRS), France

Abstract

This paper presents Matisse*, an interactive modeling system aimed at providing the public with a very easy way to design free-form 3D shapes. The user progressively creates a model by painting 2D regions of arbitrary topology while freely changing the view-point and zoom factor. Each region is converted into a 3D shape, using a variant of implicit modeling that fits convolution surfaces to regions with no need of any optimization step. We use intuitive, automatic ways of inferring the thickness and position in depth of each implicit primitive, enabling the user to concentrate only on shape design. When he or she paints partly on top of an existing primitive, the shapes are blended in a local region around the intersection, avoiding some of the well known unwanted blending artifacts of implicit surfaces. The locality of the blend depends on the size of smallest feature, enabling the user to enhance large, smooth primitives with smaller details without blurring the latter away. As the results show, our system enables any unprepared user to create 3D geometry in a very intuitive way.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Object Modeling(implicit surfaces); I.3.6 [Computer Graphics]: Methodology and techniques(sketch-based interaction).

1. Introduction

Designing 3D shapes is not easily accessible to the public: sculpting the model one has in mind from a piece of clay or wood requires some skills, while sketching, if easier, will only provide a single 2D view of the shape. Digital shape modeling, supported by impressive advances in shape representation and editing, is an interesting alternative. However, standard modeling systems are far from being usable by un-trained users: they usually require some understanding of the underlying geometrical representation, are often limited to some indirect control (e.g. manipulating control structures or weight values) and require a tedious training process before mastering the interface. Most novice users would get discouraged before being able to enjoy a creative design task.

Recently, systems based on 2D sketching have been proposed to ease the modeling of 3D shapes. However, keeping the simplicity of use of real sketches - where 3D somehow emerges from the sketch - without introducing restrictions on the shape being modeled, is very challenging. This paper presents a solution where smooth free-form shapes of any topology can be constructed by progressively painting and refining them from different viewpoints. An example is shown in Figure 1.

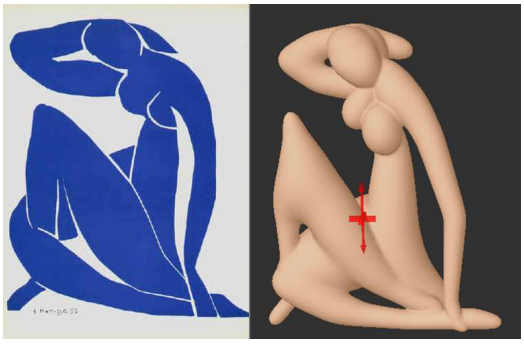

Figure 1: *Left: Original art-work from Matisse. Right: 3D model created with our system using 10 sketching steps from different viewpoints. The Matisse art-work served as a loose inspiration.*

1.1. Related Work

Sketch-based modeling systems can be divided in two categories. The first category groups systems dedicated to a specific application. This includes modeling clothes [TWB*06], trees [IOI06, OOI07], hair [WBC07] or architectural shapes [Ske]. In these systems, 3D is inferred using some a priori knowledge of the object being modeled. The second category, which includes the seminal paper Teddy [IMT99], groups systems for modeling general free-form shapes from the sketch only. Let us review this second category, to which our work belongs.

The basic idea is to reconstruct a 3D shape from contour curves, interpreted as silhouettes. In most cases, the silhouettes are restricted to closed curves and considered to be planar, which restricts reconstructed shapes to those with a flat medial axis. In contrast, [KH06] allows the user to draw complex contours including cusps and branching point, from which more general 3D shapes can be inferred. Most of these systems allow adding extra shape components from other viewpoints, for instance through cutting and extruding the model [IMT99]. To capture shapes of more general topological genus, some systems allow the creation of holes within the planar sketch of the shape, using conventions such as sketching a contour in clockwise order inside a counter-clockwise outer contour [SWSJ05, KH06]. However, this requires the user to learn specific sketching rules. Our painting metaphor reuses the idea of modeling planar silhouette components extended from different viewpoints. However, we use a painting instead of a sketching metaphor, making the creation of components of arbitrary genus straightforward.

Let us now discuss the methods used in previous systems to fit the sketched contours. A first group of methods belong to variational modeling, and are thus based on optimization: points on the sketched contours, projected into 3D, are used as position constraints for a 3D surface, using either variational implicit modeling [KHR02, ZS03, CEC*05] or Laplacian mesh editing [NISA07]. In [SWSJ05], a 2D variational field function is fitted to each contour, with choice between extrusion, revolution or standard inflation to get a 3D shape. Components sketched from different viewpoints are merged using the blobtree blending mechanisms.

A second approach for fitting a 3D shape is to rely on geometric analysis of the contour to reduce the need for an optimisation: a skeleton (or medial axis) is extracted from the contour and used to generate a 3D shape [IMT99, TZF04, AGB]. A difficulty in reconstructing such a shape is gettig a smooth surface. Recent techniques use the skeleton for defining an implicit surface, modeled as blobs [AGB] or as a convolution primitive [TZF04, ABCG05]. On one hand, blobs often create oscillations on the shape. On the other hand, the use of convolution smoothes thickness variations so much that directly fitting the contour is difficult. [ABCG05] is supposed to solve this problem but never explains how it is done. We suspect that ad-hoc correspondances tables were precomputed. [TZF04] instead introduces an extra optimization step.

In this work, we use same convolution surface model (namely, a closed-form solution for the Cauchy-kernel) but we propose a closed form formula based power transform of radius values and adapted iso-values to limit the smoothing of thickness variations. It makes our system fully interactive.

1.2. Contributions

This paper presents an interactive modeling system designed for novice users. Smooth, free-form shapes of arbitrary topological genus are progressively created by blending surfaces created from different viewing angles and zoom factors. Our main contribution is the introduction of a region-painting metaphor, that makes the creation of shape components of arbitrary topological genus straightforward. The main features of our painting system are:

- a mechanism to reconstruct a region with an implicit surface without any optimization step;
- improved, automatic ways of inferring the depth of a shape component from the elements already being drawn;
- a new local blending mechanism, *which only blends components where the user painted some overlap*, and whose parameters are automatically chosen to prevent small details from being blurred into larger shapes.
- an adaptive meshing algorithm which meshes each primitive according to the size of smallest features and locally re-meshes the surface after blending operations.

The remainder of this paper develops as follows: section 2 presents our solution for reconstructing a painted region. Section 3 details the way new shape components are combined with existing ones. Section 4 reviews the interface and presents results. We conclude and discuss future work in section 5.

2. Converting a painted region into a 3D shape

This section details the process we use for modeling a 3D shape from 2D user input. After explaining the benefits of using painted regions rather than silhouettes, we detail the way a convolution skeleton is extracted from a region, and how the convolution surface parameters are chosen so that the shape fits the region.

2.1. Painting regions instead of sketching silhouettes

In most sketch-based systems [IMT99, SWSJ05, AGB, KH06], the user is required to sketch a 2D contour, from which a 3D shape is inflated. Except in [KH06], the contour is a simple closed line, interpreted as a flat silhouette, which restricts the family of shapes one can build. Complex shapes can however be created from this mechanism, by blending flat-silhouette components built under different viewpoints.

In this work, we re-use the idea of defining shape components that all have a flat silhouette, since it makes both design by novices and 3D reconstruction much simpler than with complex contours. However, to ease the user task, we use *a painting metaphor* rather than a sketch-based one: the user selects a brush of a given size and is free to paint either a stroke (closed or not) or to fill a region, which can be done quickly using a paint bucket. An eraser is provided to edit the painting (see the mouth in Figure 2).

Using a painting rather than a sketching metaphor brings several benefits: from the user side, long and thin elements can be painted in a single gesture and edition using an eraser is very intuitive. From the system side, there is no need of monitoring the user input, such as checking that a well defined closed contour has been sketched; moreover, no extra interface is required for enabling the creation of shapes of arbitrary topological genus: complex regions such as the one in Figure 2 will be very naturally defined using the painting metaphor.

Figure 2: *Painting with different sizes of brush (a), filling the surface with paint bucket (b) and erasing for adding the mouth and correcting the eyes (c).*

2.2. Converting 2D regions into convolution skeletons

The region the user paints is stored in a texture image of a fixed size for further processing. The second step is to extract the *medial axis* of this region (defined as the locus of maximal circles included within the region) and convert it into a graph of branching poly-lines, usable as the skeleton for a convolution surface [AJC02].

Rather than processing a constrained Delaunay triangulation followed by a chordal axis extraction as in many previous works [IMT99, TZF04, AGB], we use a fast processing method (namely iterative erosion) to extract a set of pixels that approximates the medial axis. [Hal89]. The resulting *skeleton image* (Figure 3 (b)) is stored in a second texture. A benefit of this method is that the resolution of the skeleton is fixed with respect to the texture image: in practice, they will correspond to skeletons of various resolutions in the world referential, since the user freely zooms in and out before painting. Our method thus straightforwardly adapts the resolution of the skeletons to the size of shape features.

In addition to the skeleton image, we compute a Weighted Distance Transform (WDT) that uses a scan-line algorithm to propagate the distance from the border of the region: after initializing distance values to zero outside the region and one inside, two passes of forward and then backward scan are applied to each pixel P, in scan-line order:

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

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

where a and b are metric weights (we use the 3-4 metric of Figure 3) and N_i = 1..8 are the pixel's neighbors. This provides each pixel, including those of the skeleton image, with the radius of the maximal disc centered at this pixel and included into the region (see Figure 3 (c)).

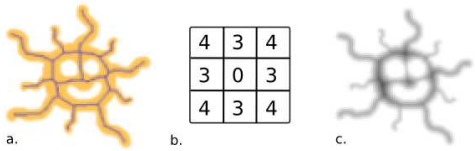

Figure 3: *Skeleton image (a), Mask used for WDT (b) and Distance image (c)*

The last step is to convert the skeleton and distance images to a convolution skeleton, that is a graph of branching poly-lines with adequate weight at vertices.

The pixel-based skeleton is first re-sampled and converted into poly-lines at an adequate resolution. To be sure that both the shape of curved branches and the variation of radii along branches are well captured, we use a sampling resolution which depends on *both geometry and brush size*. We first create vertices at skeleton pixel positions that correspond to branching points between three branches of more (pixels with 3 neighbors or more), and at pixels with a single neighbor, corresponding to the extremities of branches; then we refine each branch by adding vertices at skeleton pixels corresponding to the local extrema of curvature along the branch. Finally, we refine the branch further if needed, using a resolution depending on the brush radius. Experimental tests show that resampling the branch is necessary if the radius varies over a threshold of 10% of its maximal value. A result is shown in Figure 4 (a).

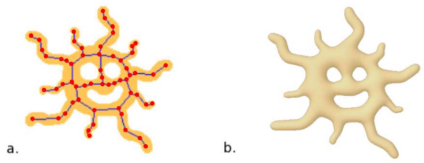

Figure 4: *Convolution skeleton (with branches) extracted from a region (a) and the resulting convolution surface (b). Note that the user gets a smoothed version of his design.*

This process outputs a graph of branching poly-lines, which will be used as a *convolution skeleton*. A radius value r_i corresponding to the pixel value in the WDT is stored at each vertex S_i of this skeleton.

2.3. From the skeleton to a convolution surface

Our assumption while reconstructing a 3D shape from a 2D region is that the local thickness of the shape in the third dimension should be similar to its thickness in the painting plane. The validity of this assumption will be discussed in Section 4. Implicit surfaces are a natural choice for reconstructing a smooth surface from a skeleton. Moreover, this representation will ease the blending of shape components designed from different viewpoints. To be able to use graphs of poly-lines as skeletons while avoiding the well known bulging problems at joints [BW97], we base our method on convolution surfaces. More precisely, we use the closed-form formula for the convolution integral of the Cauchy kernel, also used in [TZF04]. The implicit surface is defined as the set of points P such that: $F(P) = \mathbf{T}$ where $\mathbf{T}$ is a given isovalue and the field function F is the integral of the kernel function h along the skeleton S_k:

$$F(P) = \int_{S_k} h_S(P)dS \tag{1}$$

Since an integral over a given support is equal to the sum of the integrals over a partition of this support, F is computed by summing the integrals along the different line-segments composing the skeleton. To get a fast yet precise computation, we use the Cauchy kernel, which yields a closed form solution for the convolution integral [She99]:

$$h_S(P) = \frac{1}{(1+s^2d^2(P,S))^2} \tag{2}$$

where *d(P,S)* is the distance between the point P and the skeleton point S, and s is a tangent parameter that allows to control the kernel width, and thus the way this primitive will blend with others.

As shown in [TZF04], the closed form solution for the convolution integral of the Cauchy kernel can be extended to line-segments with a varying radius. Using the formulation from [TZF04], we express the field value at a point p as:

$$F(P) = w_H F_H(P) + \frac{w_T - w_H}{l} F_T(P) \tag{3}$$

where w_H (resp. w_T) is the weight of the segment's *head* (resp. *tail*), $F_H(P)$ (resp. $F_T(P)$) is a field value only depending on the branch's *head* (resp. *tail*) and l is the branch length.

2.4. Choosing convolution parameters

Intuitively, one would except that in using the desired surface radii as weights w_H and w_T in the formula above, a surface of the right thickness would be constructed. Unfortunately, some extra tuning is required: due to the integral formulation of convolution surfaces, the resulting surface is thicker than the weight values, due to the summed contribution of all points S along the skeleton. Previous work solved the problem by either manually tabulating the correspondence between weights and thicknesses and pre-inverting the table [ABCG05] or by using iterative optimization to find the weight values that best fit the region [TZF04].

We propose a simpler solution. Based on the fact that we work with texture images of fixed size, and thus that the size of the shape to reconstruct is always within the same range of values, we compute convolution weights by scaling the radius values at vertices using:

$$w_i = (C * r_i)^3 \tag{4}$$

where r_i is the desired radius along the branch, and C is a constant factor depending on the image size that normalizes distances. In practice, $C = \frac{1}{(width\ (in\ pixels)\ of\ the\ image) * 3}$ works well.

We first experimented with $w_i = C * r_i$ but curvatures of the reconstructed shape were much too smooth compared to the original drawing. We introduced the power of 3 to anticipate for convolution's smoothing. This formula for the weights scales the field function to the right range of values.

Our last enhancement is to insure that the surface fits the region by selecting the iso-surface of the field function: we compute the field value at points located on the contour, near the extremity of each branch. This gives us the iso-values the surface should go through. To best fit the region, we set the iso-value T to an average of these values.

The convolution surface resulting from this step is depicted in Figure 4 (b). As expected, the surface is smooth but only approximates the painted region. The quality of this reconstruction will be discussed in Section 4 (including the remaining problem in case of high thickness variations along a branch).

2.5. Meshing a surface component

The convolution surface resulting from the process we just described is tessellated into a mesh using [Blo94]. We set the grid resolution to half of the smallest radius value along the convolution skeleton, which corresponds to the size of the smallest feature to reconstruct.

To make field queries more efficient, we truncate the field function generated by each line-segment outside of a bounding region, using the fact that integrals of the Cauchy kernel quickly fall to zero at a distance from the skeleton. In our implementation, this is done by computing an axis parallel bounding box around each line segment augmented with discs corresponding to the local surface radius; then the size of the box is multiplied by a constant factor, and the field value is considered to be zero outside of this larger box.

3. Combining different shape components

3.1. Positioning new shapes into the scene

The Matisse system uses 2D painting from different viewing angles to generate 3D shapes. When someone paints or draws on a sheet of paper, he or she can imagine the relative depth position of the scene elements and thus "see" a 3D scene. To design an intuitive sketch-based system, we have to guess this relative depth position in a plausible way.

Frontal painting at the right depth : The first idea is to paint on a plane facing the camera, perpendicular to the viewing direction. When the user clicks on the scene using the painting tool a frontal painting plane is created. Our strategy for selecting a depth for this plane when no other information is provided is to always keep the same depth value in the camera's referential. This way, the distance to the painting plane adapts to the zoom level: when the user zooms in, he or she can paint closer details, while large background elements can be easily painted when he or she zooms out.

In most cases, the user has already started painting a shape and would like to add another component or some details to it from a different viewpoint. We then use a relative positioning mechanism: when the painting starts on an existing shape, we create a frontal plane at the depth of the point the user clicked on similarly to [CHZ00]. Since the skeleton of the new component is located in this plane, this results in slightly overlapping shape components, which allows for smooth blending between sub-shapes, as explained later in this section.

In order to give better feedback to the user when browsing the scene, we always display an almost transparent frontal plane which indicates where the next drawing will take place if the user clicks at the current mouse position. This virtual plane is rendered like clear plastic so that the meshes behind it are lighter (see Figure 5 (left)). Whatever the painting plane, the shape is created at the scale of the painted image. Then it is positioned in 3D at the right depth and scale, so that it superimposes with the painted region [CHZ00].

Figure 5: *Semi-transparent display of the frontal plane the user has selected while sketching a nose (left) and the result (right)*

Bridging between existing shapes : In some cases, the user would like to paint a connection between several existing shape components. In this case, the painting is not expected to take place in front of the camera, but in an oblique plane connecting these elements [CHZ00]. In our implementation, we allow such a bridging mechanism (see Figure 6). It can either be used for bridging between two shapes (e.g. painting a wire between two trees) or between two points on the same shape (see the arm on the right of Figure1). We select the two points giving the depths to interpolate as midway between the front and the back depth of the shape at the point the user clicks on. Since passing through two points is an under-constrained problem, we keep the painting plane vertical with respect to the current camera view. When the user paints on an oblique plane, his painting is

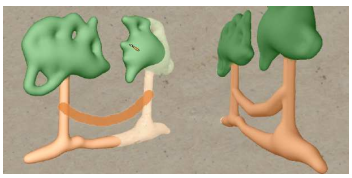

Figure 6: *Bridging between two shapes: sketch (left) and rotated result (right)*

distorted with respect to the shape component he is creating. To keep the benefits of the constant size texture image used in skeleton computation, we still compute the skeleton as if the region was facing the camera. Then we project it in the oblique painting plane while editing radius values so that the shape's frontal projection still corresponds to the painted region. The resulting oblique skeleton generates the convolution surface.

3.2. Soft blending

After reconstruction, new parts are blended to the object. Since our objects are reconstructed using truncated convolution surfaces, they can be considered as implicit surfaces with compact support so the whole range of blending operators can be used. This goes from a simple sum [Bli82,BW90] to more advanced arc-of-an-ellipse [BWG04] or displacement [HL03] blends.

Since our system allows the user to create large components as well as very small ones in the same framework and with an adapted resolution, the blending operator we choose must overcome the well-known problem of small objects "absorption" when blended with significantly larger ones [WW00], i.e. the small part being completely embedded into a smooth bump on the large object. A first solution to this was proposed in [BWG04]. But it only holds for a large object blended with one or several smaller ones of equivalent size, which is not usable in our case. We rather use the second version of the arc-of-an-ellipse operator also described in [BWG04]. It provides an accurate control of the transition size independently on each blended primitive, making it especially well suited to our purpose.

Let us denote M_1 and M_2 the meshes representing a pair

of objects to combine, noted O_1 (defined by function F_1) and O_2 (defined by function F_2). The operator we use restricts the blending to a local volume between two isovalues : V_1 (resp V_2) the value of function F_1 (resp F_2) at the blending boundary on O_2 (resp O_1) (see Figure 7). The idea is to carefully choose these isovalues so that the smallest surface component is not blend into the larger one. To do this we evaluate F_1 (resp F_2) in a point P_1' (resp P_2') at an adequat distance from object's O_1 (resp O_2) along the field gradient. So we have $V_1 = F_1(P_1')$ and we choose :

$$P_1' = P_1 - w_{2,min} \frac{\nabla F_1(P_1)}{\|\nabla F_1(P_1)\|} \quad (5)$$

Where P_1 is choosen on O_1 closed to the intersection and $w_{2,min}$, the minimum in the weights of object O_2 skeleton (see section 2.4), gives a good estimation of O_2 smallest detail.

3.3. Local application of the blend

A well known problem with implicit modeling is unwanted blending. When the user creates a complex model by successively blending several primitives, the later will very often blend all along. For instance, modeling an arm which blends with the body of a character at the shoulder but remains separated elsewhere is not easy. It usually requires the use of complex mechanisms such as blending graphs and decay functions [AJC02]. Fortunately, the controlled blending operators we use in our application enable us to set up a simpler, intuitive solution.

First of all, we consider that the user has expressed his wishes about regions to be blended by painting some local overlap between shapes (see Figure 7 left). Once a new shape component is constructed, we only blend it where its mesh overlaps with others. We achieve this through some processing on the meshes that represent the implicit surfaces to be blended: we compute the union of the meshes (a first approximation for the mesh of the blended shape we are looking for), together with the contour(s) of the intersection (see Figure 7 (a)), expressed as a list of segments. Note that there may be several separated such contours if the new component overlaps with the existing shape in several distinct regions (in our implementation this is done using the LGPL GTS library).

Starting on each contour, we remove from the union mesh the region affected by the local blending operations described in the last section. This is done by testing, at each vertex, the value of the field function representing the blended shape. If this value is not close to zero, the vertex belongs to the region to re-compute. Faces entirely constructed on such vertices are removed (see Figure 7 (b), (c)). Then a triangle strip is created to fill the gap between the remaining parts of the mesh. This strip is progressively refined by recursively subdividing triangles into 4 triangles where needed according to a curvature criterion: new points at the middle of edges are created and converged to the new, blended implicit surface each time their field value is smaller than the threshold.

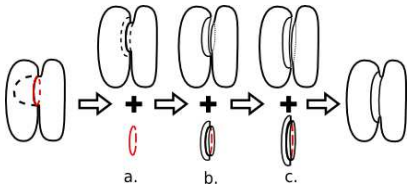

Figure 7: *Local blending of two shapes : intersection contour extraction (a), intersection removal (b), blending surface remeshing and growing (c).*

A benefit of this mechanism is that, although we do not use any blending graph structure, blending remains local: two shapes blend in the region where the user made them overlap (slightly extended according to the parameters of the local blending operators). If other parts of the same surfaces come close to each other elsewhere, but without having their meshes overlap, they will not blend in that other region. This makes the system much more predictable than usual implicit modeling, although the resulting mesh is no longer the isosurface of a usually defined field function.

4. Results and performances

4.1. Overview of the interface

Matisse provides the user with many tools not only for painting and for exploring the 3D scene but also for saving and re-using his work. These tools are:

- a paint bucket, a brush and an eraser of adjustable size
- a color-selection panel
- zoom, rotation and translation, usable at any time
- *undo*, *redo*, *save*, *load* and screenshot buttons

In everyday life, after drawing the global shape of an object, the average person's behavior is to rotate the sheet of paper to ease sketching in another direction, or to come closer for adding details. To keep this simplicity of use, our system does not require the user to click any button to reconstruct a 3D shape before further sketching: the 3D shape is automatically computed and displayed while he or she changes the point of view by zooming or moving into the scene. This is done without freezing the interface since the computation of the surface is done in threads. Thus, as in everyday life, the user is only concerned about painting and moving his drawing for filling it out. The resulting, intuitive interface can immediately be used without a learning stage.

4.2. Performances

In Matisse adding a surface component requires 3 main steps : skeleton computation, initial meshing, and blend-

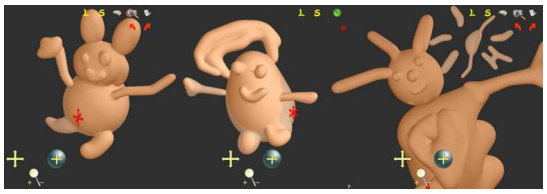

Figure 8: *Creations by 18, 67 and 12 year old novice users*

ing.Table 1 gives some performances times on a 1.8 Ghz centrino computer (times given in milliseconds). Note that the meshing mechanism we used (Section 2.5) was far from optimized. Results would be made much faster by improving this step.

Model	Implicit Reconstruction	Skeleton Meshing	Blending with existing component
Sun	120	6500	-
Star's body	70	1180	-
Tree's trunk	30	1040	-
Tree's branches	100	5360	750
Tree's bridge	20	310	740

Table 1: *Performance times for the Star (Fig 5), the Sun (Fig 2), Trees and Bridge (Fig 6)*

4.3. Informal validation by end-users

Since *Matisse* is designed for the general public, we tested the system with two kinds of users: novice users and traditional artists. The category of novice users was composed of about 10 teenagers and seniors, using the system with a simple mouse on a laptop. Their main feedback was to ask for colors, since they were provided with an earlier, mono-color version of the system. These users enjoyed both the simplicity of painting and the 3D nature of the shapes. Teenagers got accustomed very quickly to navigating through the scene and zooming to add local details. They never complained about the computed 3D shapes, although each surface component did not exactly fit their input so that they often had to bend several components to fit a slope with extreme thickness variations (see Figure 9). However, they would have liked a 3D eraser tool to remove parts of the surfaces that they disliked.

The second category of users was artists, using the system on a tablet: a CG designer and a traditional artist who never worked on computer. The traditional artist was impressed by the functionalities provided, such as *undo/redo* or *load/save*. Both asked for a more precise reconstruction of their input and for the possibility to create flat shapes, such as leaves or petals (see Figure 10). Overall, both enjoyed the simplicity of *Matisse* for creating free-form 3D shapes and asked to come back for a further test session.

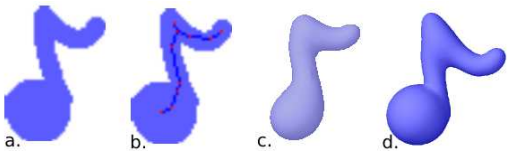

Figure 9: *Our system only provides an approximated reconstruction of regions with extreme thickness variations: painting region (a), convolution skeleton (b) and resulting 3D shape (c), a shape reconstructed by blending 2 components (d).*

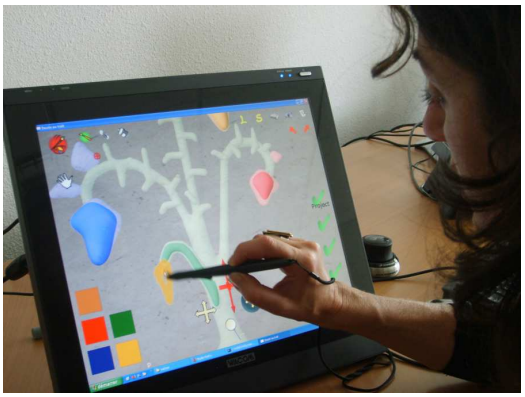

Figure 10: *A traditional artist using* Matisse

5. Conclusion

This paper presented an interactive modeling system designed for novice users. It enables the creation and refinement of smooth free-form shapes by progressively painting the regions they cover from different viewing angles and with adapted zoom factors. An implicit formulation based on convolution surfaces is used to reconstruct each part of the shape and blend it locally to the existing geometry. Our results shows the relevance of combining shapes of arbitrary genus, each reconstructed from a single drawing. *Matisse* can immediately be used by unprepared, novice users.

Being able to remove a region from a shape instead of only adding components was required by several users. This extension should be easy thanks to the implicit surface formalism we are using: we just need to include a difference operator, together with the adequate local re-computation of the mesh. Moreover, we are planning to improve the placement of details sketched over larger shapes by projecting their skeleton onto the underlying surface and extending the bridge mechanism to bridge between three scene elements, removing the ambiguity of a plane defined by two points.

Local blending was achieved thanks to an hybrid implicit/mesh representation enabling to apply the blend in regions where the meshes intersected. A useful extension

would be to provide a similar mechanism while keeping a well-defined, implicit representation for the whole shape.

Future work will also include investigating a more accurate reconstruction method for the painted regions. We would, however, like to keep the method purely geometric, since optimization would reduce performance and be less controllable in terms of smoothness.

Lastly, the thickness of each reconstructed element is related to its local size in the current version of the system and cannot be edited. Generating flat surface components could be done with surface skeletons as in [ABCG05]. However, an intuitive way for the user to select the desired thickness while sketching still has to be thought of.

5.1. Acknowledgments

We are very grateful to Gregoire Aujay for his important contributions to the implementation of Matisse, and to the artist Virginia Alfonso for helping us to validate the system. We also would like to thank the company Axiatec and the PPF "Interaction multimodale" for funding this project.

References

[ABCG05] ALEXE A., BARTHE L., CANI M.-P., GAILDRAT V.: Shape modeling by sketching using convolution surfaces. In *Pacific Graphics* (Macau, China, 2005), Short paper.

[AGB] ALEXE A., GAILDRAT V., BARTHE L.: Interactive modelling from sketches using spherical implicit functions. In *AFRIGRAPH '04*, ACM, pp. 25–34.

[AJC02] ANGELIDIS A., JEPP P., CANI M.-P.: Implicit modeling with skeleton curves: Controlled blending in contact situations. In *Shape Modeling International* (2002). Banff, Canada.

[Bli82] BLINN J.: A generalization of algebriac surface drawing. *ACM Transaction on Graphics 1*, 3 (1982), 235–256.

[Blo94] BLOOMENTHAL J.: An implicit surface polygonizer. *Graphics gems IV* (1994), 324–349.

[BW90] BLOOMENTHAL J., WYVILL B.: Interactive techniques for implicit modeling. *Computer Graphics (Proc. of SIGGRAPH 1990) 24*, 2 (1990), 109–116.

[BW97] BLOOMENTHAL J., WYVILL B. (Eds.): *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[BWG04] BARTHE L., WYVILL B., GROOT E. D.: Controllable binary operators for soft objects. *International Journal of Shape Modeling 10*, 2 (2004), 135–154.

[CEC*05] CUNO A., ESPERANÃĞA C., CAVALCANTI P., CAVALCANTI R., FARIAS R.: 3d free free-form modeling with variational surfaces. *WSCG* (2005).

[CHZ00] COHEN J. M., HUGHES J. F., ZELEZNIK R. C.: Harold: A world made of drawings. In *NPAR 2000 : First International Symposium on Non Photorealistic Animation and Rendering* (June 2000), pp. 83–90.

[Hal89] HALL R. W.: Fast parallel thinning algorithms: parallel speed and connectivity preservation. *Commun. ACM 32*, 1 (1989), 124–131.

[HL03] HSU P., LEE C.: The scale method for blending operations in functionally-based constructive geometry. *Computer Graphics Forum 22*, 2 (2003), 143–158.

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: a sketching interface for 3d freeform design. In *SIGGRAPH '99* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 409–416.

[IOI06] IJIRI T., OWADA S., IGARASHI T.: The sketch l-system: Global control of tree modeling using free-form strokes. In *Smart Graphics* (2006), pp. 138–146.

[KH06] KARPENKO O. A., HUGHES J. F.: Smoothsketch: 3d free-form shapes from complex sketches. In *SIGGRAPH '06* (New York, NY, USA, 2006), ACM, pp. 589–598.

[KHR02] KARPENKO O., HUGHES J. F., RASKAR R.: Free-form sketching with variational implicit surfaces. *Computer Graphics Forum 21*, 3 (2002), 585–594.

[NISA07] NEALEN A., IGARASHI T., SORKINE O., ALEXA M.: Fibermesh: designing freeform surfaces with 3d curves. *ACM Trans. Graph. 26*, 3 (2007), 41.

[OOI07] OKABE M., OWADA S., IGARASHI T.: Interactive design of botanical trees using freehand sketches and example-based editing. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses* (2007), ACM, p. 26.

[She99] SHERSTYUK A.: Kernel functions in convolution surfaces: a comparative analysis. *The Visual Computer 15*, 4 (1999), 171–182.

[Ske] Sketchup software: 3d sketching software for the conceptual phases of design. http://www.sketchup.com.

[SWSJ05] SCHMIDT R., WYVILL B., SOUSA M., JORGE J.: Shapeshop: Sketch-based solid modeling with blobtrees. In *SBIM* (2005).

[TWB*06] TURQUIN E., WITHER J., BOISSIEUX L., CANI M.-P., HUGHES J.: A sketch-based interface for clothing virtual characters. *IEEE Computer Graphics and Applications 27* (2006), 72–81.

[TZF04] TAI C., ZHANG H., FONG J.: Prototype modeling from sketched silhouettes based on convolution surfaces. *Computer Graphics Forum 23* (2004), 71–83.

[WBC07] WITHER J., BERTAILS F., CANI M.-P.: Realistic hair from a sketch. In *Shape Modeling International* (June 2007).

[WW00] WYVILL B., WYVILL G.: Better blending of implicit objects at different scales. *SIGGRAPH Sketch* (2000).

[ZS03] ZENKA R., SLAVIK P.: New dimension for sketches. In *Spring Conference on Computer Graphics* (Budmerice, Slovak Republic, 2003).