EUROGRAPHICS Symposium on Sketch-Based Interfaces and Modeling (2009), pp. 1–8
C. Grimm and J. J. LaViola Jr. (Editors)

Editing Level-Set Models with Sketched Curves

paper1005

Abstract

Level set models are deformable implicit surfaces where the deformation is controlled by a speed function in the level set partial differential equation (PDE). These models are widely used in computer graphics applications due to their implicit definition, low-level volumetric representation and the powerful numerical techniques used to produce the PDE-based deformation. We present a set of interactive sketch-based level-set surface editing operators. These operators allow a user to sketch curves above or on a level-set surface in order to edit the surface's shape. Once the curves are sketched the surface interactively evolves to locally fit to the curves. A user may then modify the curves in order to refine the shape of the model. The mathematics, algorithms and techniques needed to implement numerous sketch-based level set modeling capabilities are described. The speed functions that produce the surface deformations within the context of solving the level-set PDE are detailed. Several examples are presented to demonstrate the flexibility and usefulness of the editing operators.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling

1. Introduction

Surface models, e.g. triangle meshes, NURBS, and subdivision surfaces, have been the most widespread modeling representation used within computer graphics and visualization for several decades. In these models topologically 2D surfaces existing in 3D Cartesian space have been explicitly represented with 2D structures, such as triangles and spline patches. While these have been the predominant models for quite some time, implicit models, which represent surfaces as iso-surfaces of a 3D scalar field, continue to become more prevalent and important within such disparate disciplines as special effects and medicine/biology.

Level set models are one type of implicit model and are defined as an iso-surface, i.e. a level set, of a dynamic implicit function ϕ. The surface is deformed by solving a partial differential equation (PDE) on a regular sampling of ϕ, i.e. a volume dataset. Level set methods provide the techniques needed to change the voxel values of the volume in a way that moves the embedded iso-surface to meet a user-defined goal. Level set models offer numerous benefits in comparison to other types of geometric surface representations. They are guaranteed to define simple (non-self-intersecting) and closed surfaces. Thus level set editing will always produce a physically-realizable (and therefore manufacturable) object. Level set models easily change topological genus, making them ideal for representing complex structures of unknown genus. They are free of edge connectivity and mesh quality issues, and therefore do not require remeshing during editing operations. They support straightforward solid modeling operations and calculations, while offering a powerful surface modeling paradigm, a paradigm built on well-developed mathematics and numerical methods.

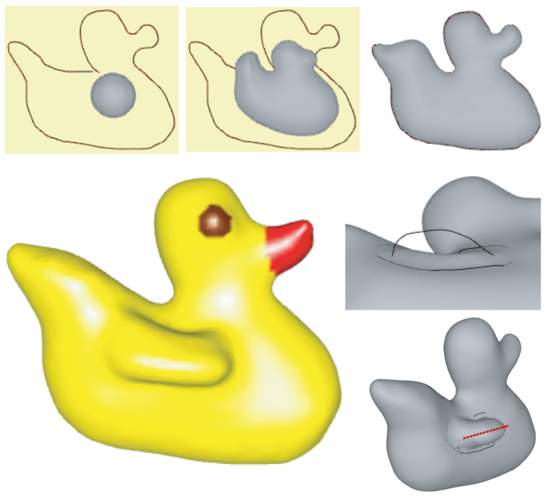
Figure 1: *A rubber duck is created from a level-set sphere and a set of sketch curves.*

Sketching communicates ideas rapidly through the creation of approximate geometric forms, and requires low overhead, no need for precise input or specialized knowledge. Sketching also provides easy-to-use methods for specifying an initial shape, as well as an intuitive approach for the correction and revision of the shape. Most current sketch-based editing systems use 3D meshes for their underlying surface representation. While these types of geometric models are widely used and are supported by numerous modeling systems, meshes limit the range of deformations during sketch-based editing operations, especially when the operations produce drastic model changes, such as extreme squeezing or spreading. In these cases, mesh-based systems need to perform a remeshing operation, which can be time-consuming and can slow system response; thus impeding user interaction and productivity.

A sketch-based modeling system based on implicit models remedies this problem by removing the inefficiency of remeshing during surface deformation. Additionally, there is a need for sketch-based modeling techniques for level-set models, as they become more widely used in entertainment and medical applications. To address this need we have developed an approach for interactive sketch-based editing of level-set models. This approach allows a user to sketch a curve on, above or near a level-set model. The model then evolves in response to the user curve-based input to create a surface that locally matches the shape of the curves. The curves may then be modified, with the level-set surface adjusting to the curve changes.

Contributions. The mathematics, algorithms and techniques needed to implement a variety of sketch-based level-set modeling capabilities have been implemented. These capabilities include local and global surface editing using single or multiple curves that are specified on or above a level-set surface. We have defined level-set speed functions to implement the capabilities within an interactive modeling framework. Our work provides a general, expressive and interactive set of editing operators for PDE-based implicit models. Some of these operators are utilized to create a rubber duck from a level-set sphere and a set of sketch curves, as seen in Figure 1.

2. Previous Work

SKETCH [ZHH96] introduced a gesture-based interface for the rapid modeling of CSG-like models consisting of simple primitives. The user sketches the salient features of a 3D primitive and the system instantiates the corresponding 3D model in the scene. An improved sketch-based modeling system, Teddy [IMT99], uses 2D user strokes to construct 3D polygonal surfaces. This highly interactive system translates simple user strokes to actions such as paint, erase, extrude, cut and smooth to create 3D models. They later developed a framework to create visually smooth surfaces from their sketch-based modeling environment [IH03]. This work was extended in [NISA07] (FiberMesh) to use a set of 3D curves to define the surfaces. For a given set of curves, the system automatically constructs a smooth surface by applying functional optimization. Another application created by the same approach is Plushie [MI07], an interactive design system for 3D plush toys. In all of these applications a relatively coarse mesh (1000-2000 vertices) is used to achieve interactive performance. [ONNI03] use a binary volume dataset to overcome the topological restrictions of Teddy. While providing a new capability, this system is limited by the spatial and bit resolution of the underlying volume dataset.

Wires [SF98] is a deformation technique which uses curves (wires), placed in close proximity to a polygonal surface, as handles to deform the surface locally. This technique has been applied to animation of facial expressions, cloth animation and surface stitching. Lawrence and Funkhouser [LF04] utilize a painting paradigm for local surface deformations, where user-applied "paint" defines instantaneous surface velocities. They initially implemented this technique using level-set surfaces, but later switched to polygonal surfaces in order to achieve interactive rates and improve spatial resolution. Cherlin et al. [CSSJ05] use interpolating parametric surfaces in their sketch-based modeling framework. While complex models can be created, the authors state that it is time-consuming to use. Layered procedural surfaces may be created and manipulated with Surface Trees [SS08], a hierarchical representation of surface patches and surface editing operations. This approach merges sketch-based interaction with a 3D analog of the intuitive layer-based metaphors found in 2D graphic design tools.

A number of surface editing operators for level set models have been described in [MBWB02]. These include sharpening, smoothing, blending, embossing, volumetric CSG and morphological operations. The algorithms and numerical techniques used to implement these operators are detailed in [MBW*05]. Baerentzen and Christensen [BC02] used the Level Set Method in their volume sculpting system, which included such capabilities as add/remove material, smooth/unsmooth and morphological operations. Another PDE-based volumetric sculpting system [DQ04, DQ05] defined smooth surfaces as a solution to a fourth order elliptic PDE subject to geometric and physical constraints, such as curvature and normals. The surface is deformed by solving the PDE with the given boundary conditions.

Wyvil et al. [WGG99] created an implicit modeling system that combines CSG operations with blending and warping. They use a tree-based representation, called the BlobTree, where leaves of the tree are the primitives and inner nodes are the operations, i.e. warp, blend, union, intersection and difference, as well as Barr deformations [Bar84]. ShapeShop [SWSJ05] uses BlobTrees as the underlying shape representation for a sketch-based editing framework. A curve-based primitive is introduced that may be "inflated" or extruded. Sketch-based models are produced by combining this primitive with CSG and blending operations. This work was extended by the addition of a curve-based free-form deformation capability [SdGWS08]. Karpenko et al. [KHR02] and BlobMaker [AJ03] use variational implicit surfaces as model representation for free-form modeling. These systems suffer from time requirements for solving the variational function. Other implicit sketching systems have used convolution surfaces [TZF04] and spherical implicit functions [AGB04], and also have significant computational requirements.

3. Level Set Methods

The Level Set Method [OS88,OF02] is a technique for tracking the evolution of a deforming interface/surface. It represents the deforming surface as an iso-surface

$$S = \{x \mid \phi(x) = k\}, \tag{1}$$

where $k \in \Re$ is the iso-value, $x \in \Re^3$ is a point in space on the iso-surface and $\phi : \Re^3 \to \Re$ is a scalar function. In other words S is the set of points which form the k^{th} iso-surface of ϕ. The choice of k is arbitrary. Deformations are defined by the change of S over time. The dynamic level set equation defines k to be constant (usually 0) and ϕ to evolve with time (Equation 2).

$$\phi(x,t) = k \tag{2}$$

We can transform this equation into a partial differential equation system which can be solved by well established numerical methods. Taking the time derivative of both sides of Equation 2 produces

$$\frac{\partial\phi(x,t)}{\partial t} + \nabla\phi(x,t)\frac{ds}{dt} = 0. \tag{3}$$

Equation 3 is a Hamilton-Jacobi type equation and defines an initial value problem for the time-dependent scalar function ϕ. There are several numerical approaches for solving the dynamic level set equation [OF02]. Equation 3 can be rewritten as

$$\frac{\partial\phi}{\partial t} = -\nabla\phi \cdot F(x, D\phi, D^2\phi \ldots), \tag{4}$$

where F is a user-defined speed term which depends on a set of order-n derivatives of ϕ as well as other functions of x. F defines the speed of the level set surface at point x in the direction of the local surface normal $(\nabla\phi/|\nabla\phi|)$. The surface is deformed over time by moving it either in or out in the direction of the normal. The magnitude and direction of the movement is specified by the F function, which is defined over the entire volume. Figure 2 presents the values produced by F at specific points on a surface given a set of sketched curves. A set of surface normals scaled by F are drawn as blue arrows.

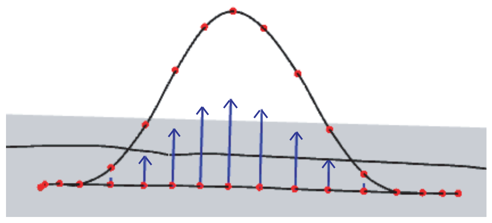
Figure 2: *Values of the F function produced by a set of sketched curves. Surface normals scaled by F are drawn as blue arrows.*

4. Foundational Technology

The sketch-based editing operators utilize previously-developed foundational technology. This technology includes an interactive level-set modeling framework and a localized editing technique for Catmull-Rom splines.

4.1. Interactive Level-Set Modeling Framework

Our interactive level-set modeling framework consists of four major components: (1) the level-set library that solves the level-set PDE on a narrow band, (2) the OpenGL user interface (UI), (3) the data structures that hold the volume and narrow-band information, and (4) the routines that translate user input into speed functions for the level-set PDE [EB09a].

The first component of the framework utilizes the VISPACK level-set library [Whi08] to efficiently solve the level-set PDE. We have developed an editing user interface using QT within an OpenGL application that has been integrated with the VISPACK library. The application accepts user actions and translates them into speed functions for the level-set equation. We have also enhanced the narrow-band implementation[†] in VISPACK to further improve its computational capabilities. The original VISPACK library computes the level-set equation over the entire surface at every time step. Additional data structures that achieve real-time evaluation of the level-set equation have been implemented. These data structures restrict computation to only those limited areas of the surface which are being modified.

4.2. Localized Editing of Catmull-Rom Splines

We employ Catmull-Rom (C-R) splines in our modeling system to provide an interactive and easy-to-use method for curve editing. C-R spline's ability to interpolate control points is an important feature, one that allows us to accurately translate user input into a mathematical representation. Other spline representations could also be utilized for

[†] Rather than track all the level sets, the narrow-band method focuses only on those grid points which are located in a narrow band around the zero level set.

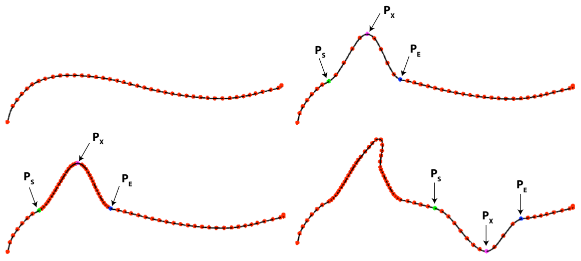

Figure 3: *Localized C-R spline editing. The purple point P_X is modified. The active window starts at the green control point P_S and ends at the blue control point P_E. Note that the window may be asymmetric and the resolution of control points can vary on different parts of the curve.*

curve definition. In order to overcome the limitation that moving a C-R control point only modifies the immediate neighborhood around the control point, we developed an approach for the localized, interactive editing of C-R splines [EB09b]. To provide greater flexibility, control and expressiveness, we have implemented techniques that expand and generalize the result of modifying one C-R control point. The techniques allow the user to define the range and type of influence that manipulation of a single control point may produce on a C-R curve, thus creating a versatile and powerful localized curve editing capability.

An active window is defined around the control point selected by the user. The extent of the window can be changed by the user any time during editing and does not need to be symmetric. The active window defines the portion of the curve that will change after a single control point editing stroke. The movement of this single control point is distributed to every control point within the window. We also provide a variable-resolution editing feature that lets the user specify the control point resolution within the active window. This enables the user to increase the density of the control points where more detail is needed, and makes that part of the curve more controllable. Figure 3 shows an example of localized editing of a C-R spline. The active window boundaries are highlighted in green and blue, and the point being modified in purple.

5. Sketch-Based Editing Operators

Several sketch-based techniques have been implemented to edit level-set surfaces. The deformations can be applied locally to a user-defined region or globally to the entire surface. A single closed curve on the surface can be used to identify a specific region of interest (ROI) to be deformed. The user then draws one or more curves on or over the surface to define the outlines of the final model. The surface within the ROI moves towards these curves with the speed functions described below. The curves may then be modified to further shape the surface.

Our sketch-based operators require the definition of a 3D

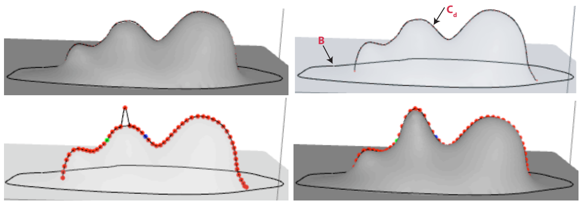

Figure 4: *Top: Two curves are sketched, one on and one above the surface. The surface grows to fit to both cross-sections. The final result is displayed with a surface drawn translucently on the right. Bottom: A control point is modified (left). The surface grows to fit to the modified curve (right).*

curve using a conventional 3-button mouse. To address this issue, we offer two methods for specifying 3D curves. The first method provides a *helping-plane*. During editing operations the 2D input produced by mouse strokes can be mapped onto an arbitrary plane within the scene. If utilized, the plane can be added at a point of interest on the model and displayed in the scene with a translucent color. Initially, the plane's normal is set to face the user, however, it can be changed to an arbitrary orientation with a mouse interaction. 2D input is mapped onto the plane during curve sketching operations. A helping-plane (displayed in translucent yellow) is used to define the shark tail in Figure 9, and the initial plane of the shamrock and duck in Figures 1 and 8. In the second method the 3D curve is defined to lie on a plane perpendicular to the view direction. The curve begins at the point where the first mouse click intersects the level-set surface.

5.1. Sketching a Single Cross-section

With this operator a curve may be used to define a cross-section of a local shape change. The user draws a closed boundary curve (B) on the surface to define an ROI and another curve (C_d) that defines a cross-section of the desired shape. Every point within the ROI moves towards C_d with a speed function defined in Equation 5 until the surface reaches C_d. We created a "mountain" on the surface with C_d defining the peak and B defining the extent of the foothills (Figure 4). Once the evolution starts, a third curve (C_s) is created on the surface. This curve is represented as a dense set of points and is the projection of C_d onto the surface perpendicular to the view direction. C_d is also represented with a similar set of dense points and a one-to-one correspondence is defined between each pair of points on the two curves. At every step of the evolution, C_s morphs toward C_d and the surface grows to meet the new C_s, until C_s (and the surface) reaches C_d.

The red dots on the cross-section curve in Figure 4 are control points that can be manipulated by clicking and dragging. New control points can also be added to the curve. After a control point is moved or added, the curve is recalculated and the level-set equation is solved once more to

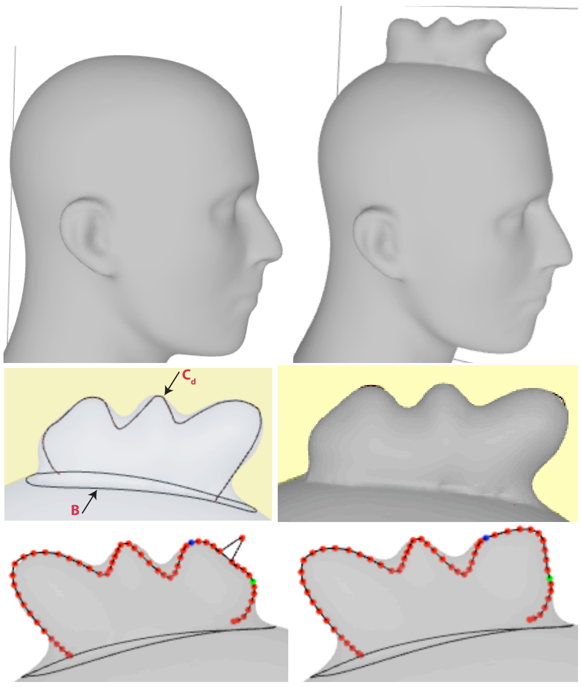

Figure 5: *Top: One cross-section curve is used to create a mohawk for the mannequin head. Middle: Two curves define the shape of the mohawk. The surface fits to these curves. Bottom: The cross-section curve is modified for further refinement of the final shape. The surface is drawn translucently to provide a clearer view of the curves and control points.*

fit to the new cross-section. An initial bump defined by two sketched curves is shown in Figure 4 (top). A control point on the upper curve is pulled upwards. Figure 4 (bottom-right) presents the resulting cross-section curve and the surface that evolves to fit to the curve.

The speed function for this operator is

$$F(x) = \frac{d_{up}(x)}{max(d_{up}(x))} * f(d_{out}(x)) * \frac{max(d_{in}(x)) - d_{in}(x)}{max(d_{in}(x))}, \tag{5}$$

$$f(d) = \begin{cases} 1.0 & d > \varepsilon \\ (d/\varepsilon)^2 & d \leq \varepsilon, \end{cases} \tag{6}$$

where x is a point on the surface, $d_{out}(x)$ is the geodesic distance from x to B, and $d_{in}(x)$ is the geodesic distance from x to C_s. The first step of calculating d_{up} for point x involves finding the closest point in the point set representing C_s from x, called x_{cs}. Recall that x_{cs} has a corresponding point on C_d, called x_{cd}. $d_{up}(x)$ is simply the Euclidean distance between x_{cs} and x_{cd}. Both max functions are taken over all points in the ROI. $f()$ is defined in Equation 6, and ensures that the speed function (and therefore the deformation) goes to zero within a distance ε to boundary curve B.

The first term of Equation 5 ensures that the evolution will stop once the surface reaches the C_d curve. Together the last two terms define the speed function as a decreasing function of geodesic distance from the cross-section curve C_d to the boundary curve B. In Figure 5 this operator is used to give the mannequin head a mohawk.

5.2. Multiple Cross-Section Curves

The operator from the previous section was extended to create editing capabilities that use several curves to define a 3D shape. The initial challenge with this approach was to develop an interface that would allow the user to easily sketch his/her ideas. Two techniques were implemented for editing a level set surface by sketching multiple curves with a conventional 2D mouse, and are described in Sections 5.2.1 and 5.2.2. In both approaches the user first draws a closed curve on the surface to define an ROI. Multiple cross-section curves are then sketched to define the 3D shape. Once the initial curves are placed the user then can modify them to specify further details. The model is drawn in translucent colors in this mode to aid curve editing. Once curve sketching is completed, the system may be placed in surface evolution mode. In this mode the surface evolves within the ROI after each curve modification. This mode may also be toggled off to once again allow multiple edits of the curves before the surface is updated.

5.2.1. Sketching over the surface

This method allows the user to edit and deform a level-set model by sketching planar curves over the surface. One challenge when implementing this editing operator occurs when curves overlap in space. The 3D shape is not well-defined when the input curves cross over each other but do not intersect. We have explored several approaches to this problem and have found two reasonable solutions. In the first solution, the surface grows locally until it reaches one of the curves, and then stops at the first curve. The second solution involves blending the influence of each cross-section curve at each point within the ROI. Both methods utilize the speed function defined in Equation 5 and require the calculation of d_{up} and d_{in} relative to each cross-section curve at every point in the ROI.

In the first solution the speed function at point x on the surface is calculated with Equation 5 using the closest curve, where "closest" is defined to be the one with the lowest associated d_{in} value. The shape in Figure 6 (top right) is produced with this method. The second solution uses a blending function to calculate the speed of a point on the surface by combining contributions from multiple individual speed functions. In general this approach drives the surface to a location between overlapping, non-intersecting cross-section curves. We found this approach to produce more pleasing results and has been used for the remaining examples. Equation 7 describes the general structure of the speed function

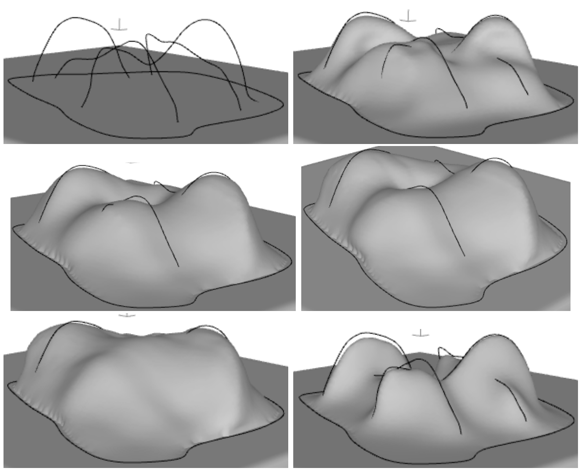

Figure 6: *Sketching cross-section curvess over the surface. Top left: The input curves. Top right: Fitting the surface to the closest curve. Middle left: Blending with* α_c^1. *Middle right: Blending with* α_c^2. *Bottom left: Blending with* α_c^3. *Bottom right: Blending with* α_c^4.

for multiple curves.

$$F(x) = \sum_{c=1}^{N_c} \alpha_c(x) * F_c(x), \tag{7}$$

where N_c is the number of cross-section curves, $\alpha_c(x)$ is one of the four blending functions in Equation 8, and $F_c(x)$ is defined by Equation 5. For both terms the subscript c indicates that the calculation is done relative to cross-section curve c.

$$\begin{aligned}
\alpha_c^1(x) &= \tfrac{1}{2} + \tfrac{1}{2} * cos(\tfrac{d_{in}(x)}{max(d_{in}(x))}\pi) \\
\alpha_c^2(x) &= 1 - \tfrac{d_{in}(x)}{max(d_{in}(x))} \\
\alpha_c^3(x) &= \tfrac{1}{N_c} \\
\alpha_c^4(x) &= \tfrac{1}{d_{in}(x)}
\end{aligned} \tag{8}$$

where $d_{in}(x)$ is the shortest distance between point x and the C_s curve associated with cross-section curve c, and $max(d_{in}(x))$ is computed over all cross-section curves.

Figure 6 shows how the surface deforms given the input curves in the top left and the different blending functions defined in Equation 8. Although all blending functions generate reasonable results, α_c^1 generates the closest fit to all cross-section curves while producing smooth transitions on the surface in between curves. α_c^2 produces the second best fit to the curves. α_c^3 and α_c^4 are not necessarily satisfactory choices for blending functions because α_c^3 is unable to fit to some of the curves closer to the surface and α_c^4 produces an unacceptable sharp drop-off from the cross-section curves. We use α_c^1 for the remaining examples in the paper.

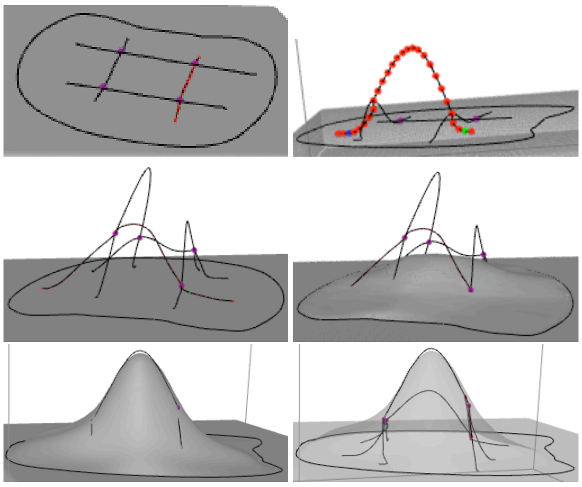

Figure 7: *Sketching on the surface. Top-left: Initial layout. Top-right: moving one control point. Middle-left: Final layout of cross-section curves. Middle-right: Surface evolving to fit the cross-section curves. Bottom row shows two images of the final shape, with the surface drawn transparently to better see the curves.*

5.2.2. Sketching on the Surface

We have also developed a third solution to the intersection problem described in Section 5.2.1 by giving the user control over curve-curve intersections. This method requires all cross-section curves to be drawn on the surface initially. A 3D line intersection algorithm is then used to calculate intersection points between two curves. Short line segments connecting two consecutive points on each curve are tested against each other for intersections. A bounding box optimization technique is utilized to improve running time. Approximate 3D intersection points are calculated using a closest point algorithm. Once the intersection points are calculated the two curves are bound together, and the two curves stay attached to each other at these intersection points during curve editing operations.

Once all curves are drawn and all the intersection points are calculated, the user can modify the curves by pulling on the control points. Any of the cross-section curves can be selected by switching between curves using the arrow keys. Once a control point on a curve is moved, the curve is modified using techniques described in [EB09b]. When a single curve is edited the change is propagated to intersecting curves via intersection points. When an intersection point on one curve is moved to a new location, the movement is interpreted as an editing operation performed on the connected neighboring curves at the shared intersection points. Once curve sketching is complete, the system may be placed in surface evolution mode, and the surface moves within the ROI to fit to the cross-section curves using the speed function in Equations 7 and 8. Figure 7 presents a surface editing session using this approach.

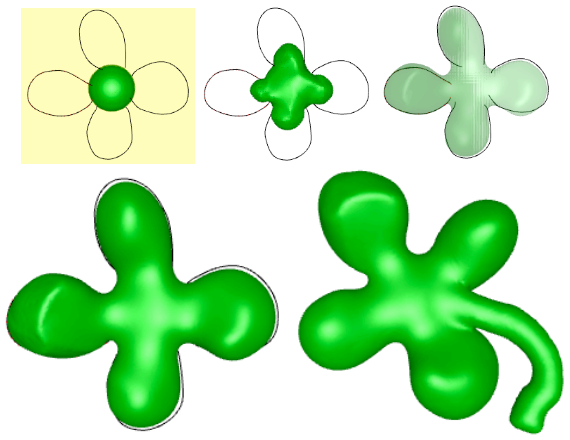

Figure 8: *Global editing example. The sphere is modified with 4 curves to create a shamrock. An intermediate step during evolution is shown in top middle frame and the final result is drawn translucently in top right. The model is further modified to add a stem in bottom right with a point-based editing operator.*

5.3. Global Deformations

All editing operators described so far are local, i.e. they are applied only in a user-defined portion of the surface. These operators can be extended to act globally. In this mode, the operators are applied to the entire surface. The general shapes of a number of models have been specified with this operator. For example the petals of the shamrock in 8, and the initial definition of the shark and duck in Figures 1 and 9. The speed function for this operator is also defined in Equations 5, 7 and 8. Note that the function $f()$ (Equation 6) will always be 1 during global editing, since there is no ROI boundary.

6. Results

In this section we present models created using our sketch-based editing tool. Table 1 shows running times and model resolutions. All models have been created on a 3.2 GHz Intel Xeon CPU. Figures 4, 6 and 7 begin with a box model with resolution $161 \times 161 \times 101$. Figure 5 uses the scan-converted mannequin head model at resolution $134 \times 160 \times 186$. Figures 1, 8 and 9 all start with a $20 \times 20 \times 20$ sphere within a $150 \times 150 \times 150$ resolution volume. The final resolution of the bounding box around each model is given in Table 1.

We created three "plastic toys" using the sketch-based modeling system. The shark model in Figure 9 is created using eight curves and a combination of local and global editing operators. The initial body is created from the sphere using a single cross-section curve and the global editing operator. Three additional curves further define the head and the tail and create a slightly curved body. Three fins are added similarly by using a closed curve to define a ROI and

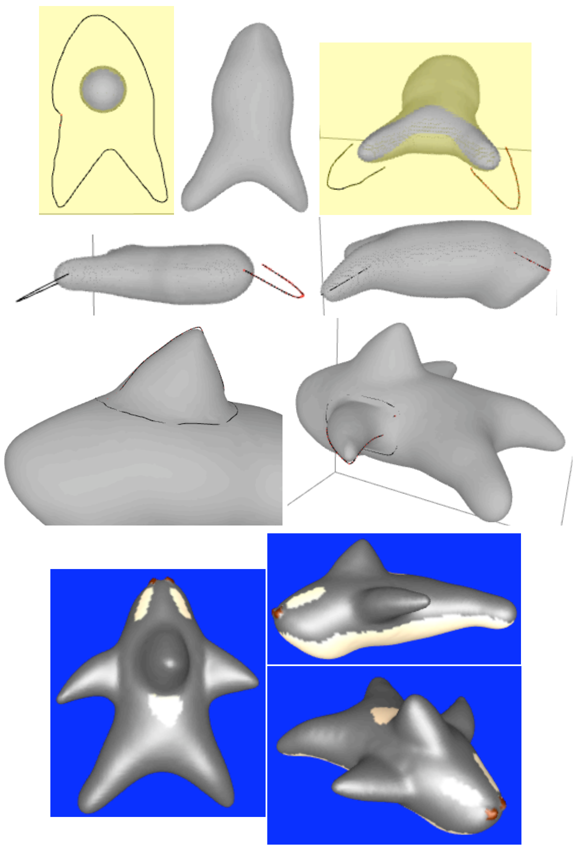

Figure 9: *The sphere and a cross-section curve is used to create the initial shark body. The tail and head are modified using additional curves. The fins are added by locally editing the shark body. The final painted model is shown from three different views.*

a cross-section curve for the fin shape. A painting capability [EB09a] allows color to be placed on the model. The painted and shaded final model can be seen from different views in Figure 9.

The duck (Figure 1) and the shamrock (Figure 8) models are similarly created using sketched curves and the level-set sphere model. Details like the eyes on the duck and the stem of the shamrock are created using point-based free-form editing operators [EB09a].

Model	Resolution	Speed (fps)
Figures 4, 6, 7	$161 \times 161 \times 101$	50-100
Figure 5	$134 \times 160 \times 186$	58
Figure 8	$45 \times 50 \times 35$	83
Figure 9	$85 \times 99 \times 54$	66-90
Figure 1	$79 \times 67 \times 37$	66-90

Table 1: *Model resolution and running times for the final results. Speed is in frames-per-second (fps).*

7. Conclusions and Future Work

We have developed interactive sketch-based level-set surface editing operators. These operators allow a user to sketch curves above or on a level-set surface in order to edit the surface's shape. Once the curves are sketched the surface interactively evolves to locally fit to the curves. A user may then modify the curves in order to refine the shape of the model. The mathematics, algorithms and techniques needed to implement numerous sketch-based level set modeling capabilities have been described. The speed functions that produce the surface deformations within the context of solving the level-set PDE have been detailed. Several examples have been presented to demonstrate the flexibility and usefulness of the editing operators. These initial examples have been created with low-resolution volume datasets, which limit the amount of surface detail that can be specified. Now that the mathematics of the operators have been designed, tested and validated, the next phase of our work will implement these editing capabilities within a high-resolution level-set framework, e.g. DT-grids [NM06]. This will allow us in the future to create level-set models with much finer surface structures and shapes.

References

[AGB04] ALEXE A., GAILDRAT V., BARTHE L.: Interactive modeling from sketches using spherical implicit functions. In *Proc. AFRIGRAPH'04* (2004), pp. 25–34.

[AJ03] ARAUJO B., JORGE J.: Blobmaker: Free-form modelling with variational implicit surfaces. In *Proc. 12th Encontro Portugues de Computacao GraÞca* (2003).

[Bar84] BARR A.: Global and local deformations of solid primitives. In *Proc. SIGGRAPH* (1984), ACM, pp. 21–30.

[BC02] BAERENTZEN J., CHRISTENSEN N.: Volume sculpting using the level-set method. In *Proc. International Conference on Shape Modeling and Applications* (2002), pp. 175–182.

[CSSJ05] CHERLIN J. J., SAMAVATI F., SOUSA M. C., JORGE J. A.: Sketch-based modeling with few strokes. In *Proc. Spring Conference on Computer Graphics* (2005), pp. 137–145.

[DQ04] DU H., QIN H.: A shape design system using volumetric implicit PDEs. *Computer Aided Design 36*, 11 (2004), 1101–1116.

[DQ05] DU H., QIN H.: Dynamic PDE-based surface design using geometric and physical constraints. *Graphical Models 67*, 1 (2005), 43–71.

[EB09a] EYIYUREKLI M., BREEN D.: Interactive free-form level-set surface-editing operators, 2009. in preparation.

[EB09b] EYIYUREKLI M., BREEN D.: Localized editing of Catmull-Rom splines, 2009. to be published in Proc. CAD'09.

[IH03] IGARASHI T., HUGHES J. F.: Smooth meshes for sketch-based freeform modeling. In *Proc. ACM Symposium on Interactive 3D Graphics* (2003), pp. 139–142.

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: A sketching interface for 3-D freeform design. In *Proc. SIGGRAPH* (1999), pp. 409–416.

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

[LF04] LAWRENCE J., FUNKHOUSER T.: A painting interface for interactive surface deformations. *Graphical Models 66*, 6 (2004), 418–438.

[MBW*05] MUSETH K., BREEN D., WHITAKER R., MAUCH S., JOHNSON D.: Algorithms for interactive editing of level set models. *Computer Graphics Forum 24*, 4 (2005), 821–841.

[MBWB02] MUSETH K., BREEN D., WHITAKER R., BARR A.: Level set surface editing operators. *ACM Transactions on Graphics (Proc. SIGGRAPH) 21*, 3 (2002), 330–338.

[MI07] MORI Y., IGARASHI T.: Plushie: an interactive design system for plush toys. *ACM Transactions on Graphics (SIGGRAPH 2007) 26*, 3 (2007), 45.

[NISA07] NEALEN A., IGARASHI T., SORKINE O., ALEXA M.: Fibermesh: designing freeform surfaces with 3D curves. *ACM Transactions on Graphics (SIGGRAPH 2007) 26*, 3 (2007), 41.

[NM06] NIELSEN M., MUSETH K.: Dynamic tubular grid: An efficient data structure and algorithms for high resolution level sets. *Journal of Scientific Computing 26*, 3 (2006), 261–299.

[OF02] OSHER S., FEDKIW R.: *Level Set Methods and Dynamic Implicit Surfaces*. Springer, Berlin, 2002.

[ONNI03] OWADA S., NIELSEN F., NAKAZAWA K., IGARASHI T.: A sketching interface for modeling the internal structures of 3D shapes. In *Proc. 4th International Symposium on Smart Graphics* (2003), pp. 49–57.

[OS88] OSHER S., SETHIAN J.: Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics 79* (1988), 12–49.

[SdGWS08] SUGIHARA M., DE GROOT E., WYVILL B., SCHMIDT R.: A sketch-based method to control deformation in a skeletal implicit surface modeler. In *Proc. 5th Eurographics Workshop on Sketch-Based Interfaces and Modeling* (2008).

[SF98] SINGH K., FIUME E.: Wires: a geometric deformation technique. In *Proc. SIGGRAPH* (1998), pp. 405–414.

[SS08] SCHMIDT R., SINGH K.: Sketch-based procedural surface modeling and compositing using surface trees. *Computer Graphics Forum 27*, 2 (2008), 321–330.

[SWSJ05] SCHMIDT R., WYVILL B., SOUSA M. C., JORGE J. A.: Shapeshop: Sketch-based solid modeling with blobtrees. In *Proc. 2nd Eurographics Workshop on Sketch-Based Interfaces and Modeling* (2005), pp. 53–62.

[TZF04] TAI C. L., ZHANG H., FONG J. C. K.: Prototype modeling from sketched silhouettes based on convolution surfaces. *Computer Graphics Forum 23*, 1 (2004), 71–83.

[WGG99] WYVILL B., GALIN E., GUY A.: Extending the CSG tree. warping, blending and boolean operations in an implicit surface modeling system. *Computer Graphics Forum 18*, 2 (June 1999), 149–158.

[Whi08] WHITAKER R. T.: *VISPACK*. Tech. Rep. UUCS 08-0011, School of Computing, University of Utah, 2008.

[ZHH96] ZELEZNIK R. C., HERNDON K. P., HUGHES J. F.: Sketch: an interface for sketching 3D scenes. In *Proc. SIGGRAPH* (1996), pp. 163–170.