WarpCurves: A Tool for Explicit Manipulation of Implicit Surfaces

Masamichi Sugihara[a,b], Brian Wyvill[a], Ryan Schmidt[c]

[a] *University of Victoria, Canada*
[b] *Intel Corporation, Canada*
[c] *University of Toronto, Canada*

Abstract

We introduce *WarpCurves*, a technique for interactively manipulating an implicit surface using curve-based spatial deformations. Although implicit surfaces have several advantages in 3D modeling, current workflows are limited by the compositional nature of implicit modeling. Wide classes of surface features that are easy to create with the direct manipulation tools available for explicit surface representations are difficult to reproduce using volumetric implicit operations. We describe a novel spatial deformation that can be used to approximate direct surface manipulation. With our method an artist first draws a curve on the current surface to indicate the feature region-of-interest. Deformations applied to this handle curve are transferred to the implicit surface via an automatically-constructed C^2 continuous space mapping. Additional curves can be added in a hierarchical manner to create complex shapes. Our technique is implemented as a node in the BlobTree hierarchical implicit volume representation, and hence can be used along with other volumetric nodes (operators) such as blending and CSG. Our results show that surface deformations which would be difficult to reproduce using existing volumetric operations can be quickly constructed using warp curves, making them a valuable addition to the implicit modeling toolbox.

Keywords: implicit surfaces, curve-based deformation, spatial deformation, direct manipulation

1. Introduction

Implicit surfaces offer several modeling advantages compared to parametric and point-sampled representations [5]. Since blending and CSG operators can be defined with simple formulations, smooth transitions and topological changes can be easily handled. By organizing these operators in a hierarchical structure called the BlobTree [37], complex solid models can be created. Tools like ShapeShop [27] have demonstrated that even novice users are capable of creating 3D models with implicit surfaces.

Despite these benefits, volumetric implicit modeling requires the artist to think in terms of shape composition, rather than surface manipulation. While composition is suitable for creating initial forms, direct manipulation of the 3D surface can be much more efficient when attempting to refine the shape, alter features, and add details. Schmidt & Singh [25] noted that artists preferred explicit representations for such tasks, but their mesh-based "Surface Tree" is incompatible with implicit operators. Hence, our goal is to enable explicit manipulation of implicit surfaces.

To integrate into the BlobTree functional hierarchy we must treat the input surface as a "black box", so direct manipulation must be formulated as a spatial deformation. Furthermore, unlike with point-based surfaces the "reverse" deformation is required [38], and should be very efficient to compute if it is to be used in interactive contexts. Several *space warp* techniques have been developed for implicit surfaces [21, 38, 23, 28], but these cannot be easily adapted to the task of surface manipulation. Recently Sugihara et al [33] described a space deformer based on a curve handle which can approximate small surface deformations, but no attempt is made to have the surface explicitly track the curve.

We present *WarpCurves*, a curve-based spatial deformation technique which can be used to apply local and global deformations to implicit models, including approximate direct manipulation of an implicit surface. Our interface is inspired by curve-based deformation techniques such as Wires [31] and FiberMesh [19]. Similar to FiberMesh, warp curves can be placed by sketching on the surface and can be manipulated using peeling techniques [12]. As the artist modifies the warp curves, our method constructs appropriate inverse warps which result in the surface approximately tracking the curve. These C^2 continuous variational warps are automatically bounded to limit their spatial influence. We demonstrate our techniques with the BlobTree [37], but the general approach is applicable to any implicit (or explicit) surface. The resulting spatial deformation can be integrated as a node in the BlobTree, allowing blending and CSG operators to be applied before or after deformation (Figure 1). Our contributions can be summarized as follows:

- A novel and flexible curve-based spatial deformation technique which can be integrated into procedural implicit modeling as an operator node.
- An artist-oriented curve-based interface supporting direct manipulation of implicit surfaces.
- A new technique to automatically bound the influence of curve-based spatial deformations.

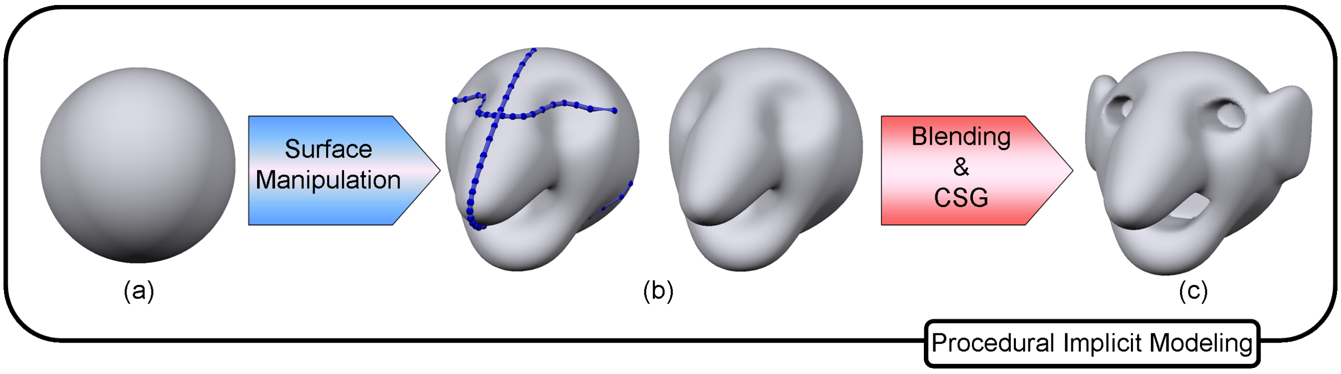


Figure 1: An overview of the WarpCurves tool. The implicit point primitive (a) is deformed using 3 warp curves to create facial features (b). Soft-CSG operators are then used to add eyes and a mouth, and the ears are added with blending operators (c).

2. Related Work

Deformation has a long history in computer graphics, so we restrict our review to spatial and curve-based deformation. For a more detailed overview of the deformation literature, we refer the reader to recent surveys of the area [18, 10, 32].

A well known artist-oriented deformation technique is free-form deformation (FFD), which is based on spatial deformation, originally introduced by [29]. Conceptually, the target object is embedded in a 3D grid or *lattice*, which is used to define a deformation field. As the artist manipulates the control points of the lattice, the embedded surface is deformed. Several extensions have been developed to make a 3D lattice more controllable [7, 16, 20]. Implicit fields are also used as deformation fields of FFD [8], although the target object of this technique is limited to point-based surfaces.

Recently, cage-based methods have been proposed which embed the target object within an arbitrary non-convex polyhedral cage, and then generate a spatial deformation based on deformations of the cage [14, 13, 15, 3]. The cage is a closed surface and typically a coarse mesh of the target object is used. Thus, a cage is geometrically and topologically more flexible than a lattice as a deformation controller. In general, FFD-style techniques provide simple and intuitive control over spatial deformation, but it is difficult to control changes to the surface, as the target object is not edited directly. Also, they are designed for point-based surfaces and do not easily extend into the implicit domain.

Curve-based techniques are often more efficient when adding or modifying surface features via deformation. The Wires system [31] binds the target object with several curves called *Wires* and then constructs a bounded spatial deformation according to the displacements of the Wires. This approach is highly interactive, easy to incorporate into procedural models, and intuitive for artists, and hence is used extensively in commercial modeling tools. Since the Wires can be aligned with important features of the target object, it is easy to directly manipulate specific parts of the surface. This idea has been extended to edit man-made objects while automatically preserving salient features (iWIRES [11]). In sketch-based modeling, curves are also used to define the features of the free-form surfaces [19]. All of these techniques provide efficient, artist-oriented interfaces, and have inspired our approach, however they are limited to point-based surface representations because they are difficult and/or inefficient to invert.

Sculpting interfaces can also be used to deform surfaces. Swirling Sweepers [1] and zero-divergence vector field deformation [36] are efficient and highly intuitive tools for direct manipulation of discrete 3D surfaces. These techniques can be applied to functional surfaces, but involve a relatively expensive path integration for each point. During interactive sculpting this cost is amortized over many frames, but in the implicit context each evaluation of the scalar field would require a prohibitively expensive reverse-integration along the entire path. A recent curve-based sculpting technique for level-set models [9] has similar limitations, and also involves a temporal component in the form of per-frame partial differential equation (PDE) evolution. In this case the result is not strictly a spatial deformation and cannot be directly applied to functional implicit models.

To be applied to an arbitrary implicit surface, a deformation must be formulated as a spatial deformation and the ”inverse warp” (the map from the deformed to initial surface) is required. Early work adapted the Barr deformations [2] for several implicit representations such as the function-representation (FRep) [21], skeletal implicit surfaces [38], and the BlobTree [37]. The work of Wyvill and van Overveld [38] also presented deformation fields suitable for animation, such as those which squash or stretch implicit models as they pass through the field. A variational warp was applied to implicit surfaces in [24] and this technique was incorporated into a general framework for FRep deformations called extended space mapping [23]. A deformation field is constructed by interpolating the displacements of control points. Since this technique interpolates an arbitrary set of control points, the control points can be placed on the important features of the target object. Due to a global feature of the variational technique, however, local influence cannot be guaranteed. A more controllable deformation technique for FRep was developed by [28] which defines another FRep object as a deformation field. However, such deformation fields must be defined manually by the artist, which is time consuming and can be unpredictable. Also, because the inverse warp is required, defining deformation fields is more compli-

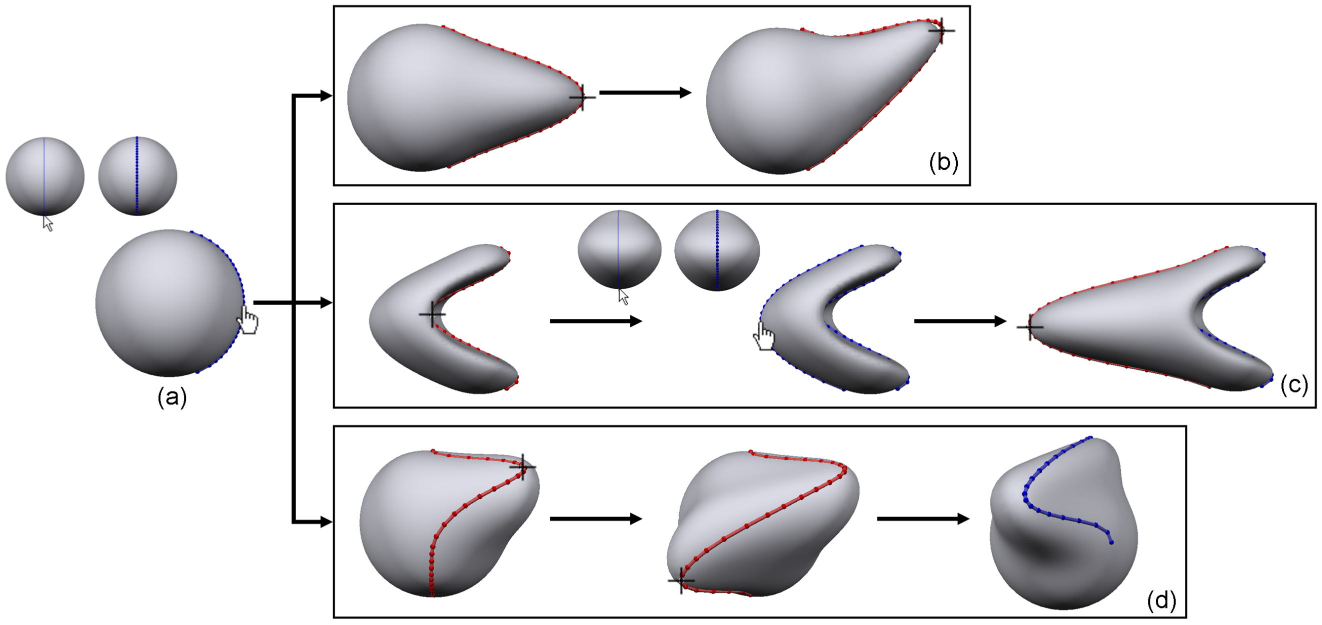


Figure 2: The WarpCurves interface. A curve can be placed anywhere on the surface by sketching (a). The artist can grab a vertex of the curve and drag it to the desirable place parallel to the screen, and accordingly the implicit model is deformed (b). Multiple curves can be used at the same time by simply sketching another curve (c). Several vertices can be manipulated in one curve (d). In our interface, when the cursor touches a curve, it becomes a hand and the curve is available to be edited. The cursor becomes a cross during dragging. The selected curve becomes red while the other curves are blue.

cated than the deformation techniques which can be applied to point-based surfaces.

Recently an FFD-style deformation for implicit surfaces was described in which the artist manipulates a curve handle, which in turn guides the automatic construction of a variational deformation field [33]. If a surface curve is used this approach can provide the appearance of local control over the surface, but the deformation is based on regular 3D lattice and so the deforming surface often will not track the curve handle, and larger deformations inevitably become global warps. In addition, to achieve real-time performance a coarse approximation was necessary. The techniques we present do not share these limitations.

3. Overview

The WarpCurves interface is modeled after FiberMesh [19] as shown in Figure 2. The artist first draws a polyline curve on the portion of the surface which is to be manipulated. The curve can be modified by pulling or pushing individual vertices. The translation of a vertex is transferred to the rest of the curve based on the peeling technique proposed by Igarashi et al [12], which automatically specifies the deformation region of interest (ROI) based on the maximum displacement of the vertex during the interaction. Essentially, as the control vertex is pulled further and further, a correspondingly larger portion of the curve adjacent to the control vertex is deformed. As the curve is manipulated, the surface is modified accordingly by a WarpCurve spatial deformation. The ROI of this deformation is also determined according to how far the artist pulls the curve. Hence, the WarpCurves tool allows the artist to interact with an implicit model as if editing the surface directly, but creates this effect by deforming the underlying scalar field of the implicit model (Figure 3).

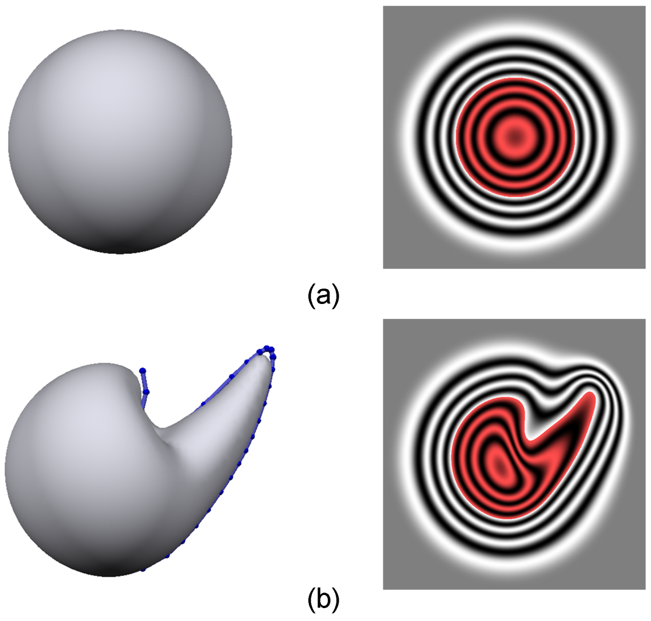


Figure 3: The scalar fields before (a) and after (b) deformation. These scalar fields are sampled from a slice along a plane and the fields which store the field values greater than the iso-value are marked in red. As a curve is manipulated, the system deforms the underlying scalar field of the implicit point primitive and then renders the iso-surface of the deformed scalar field (b).

WarpCurve deformations are designed to be used as an operator which can be incorporated into procedural implicit modeling, as shown in Figure 1. To accomplish this, we implemented an interactive WarpCurves tool within ShapeShop [27], which is a sketch-based modeling system that uses the BlobTree hierarchical implicit volume representation [37]. The BlobTree can represent complex models defined by a tree of primitives and composition operators. The primitives are stored at the leaves and combined at the interior nodes using composition operators such as blending and CSG. A warp node is generally defined as a unary node which takes a scalar field as input and generates a deformed scalar field as output. We also represent our curve-based deformation as a unary node, a Warp composition node which constructs and applies the spatial deformation to its input. The Warp node stores a WarpCurve as the control curve for the deformation. To ensure interactivity, cache nodes [26] are inserted above the constructed Warp nodes. Note that if several WarpCurve operations are applied in the same region of space, the most straightforward structure with one Warp node per curve cannot guarantee that all curves remain on the surface (Figure 4). To address this problem we introduce a *HybridWarp* node in Section 4, which integrates multiple deformations so that the surface tracks all the input curves.

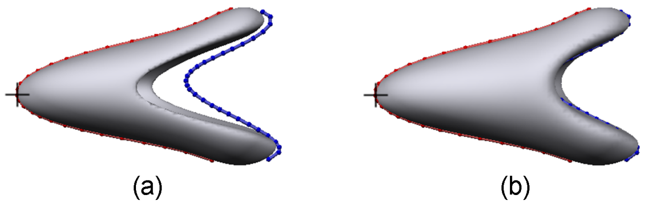


Figure 4: The hybrid warp node. In the current BlobTree structure, the first drawn curve does not have influence when the second drawn curve constructs deformation (a). By using a HybridWarp node introduced in Section 4 in the BlobTree, both of the curves can affect the deformation (b).

As the artist manipulates the input curves, the Warp node constructs two types of fields, the *deformation field* and the *bounding field*, and combines them to create a spatial deformation. The deformation field is an inverse warp computed from the displacements of the curves. We construct the deformation field using the variational warp technique [24, 6] because it can interpolate an arbitrary set of displacements with C^2 continuity. Since the variational technique has global influence (Figure 5a), however, we also construct a bounding field to restrict the warp to a local region. Application of the bounding field is similar to the bounded blending technique [22], however there the artist had to manually define a bounding field using primitive composition. The Warp node automatically constructs a bounding field using the stored WarpCurve as a skeleton. The bounding field has the structure of a skeletal primitive, where the scalar field drops off from 1 to 0 as distance to the skeleton increases. By applying the bounding field to the deformation field, the influence of the warp smoothly decays away from the input curve, and there is no influence at all outside the bounding field (Figure 5b). Mathematically, the scalar field $f_{M'}$ of the deformed implicit model can be defined as follows:

$$f_{M'}(\mathbf{p}) = f_M(\mathbf{p} + f_{bounding}(\mathbf{p}) \cdot \mathbf{D}(\mathbf{p})) \tag{1}$$

where f_M is the scalar field of the original implicit model, $\mathbf{p}$ is a sample 3D point, $f_{bounding} : \mathbb{R}^3 \rightarrow \mathbb{R}$ is the bounding field, and $\mathbf{D} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ is the deformation field. In the implicit domain, deformation is realized by displacing the sample points instead of deforming the scalar field itself. The deformation field $\mathbf{D}$ returns the displacement of a point $\mathbf{p}$, and the displacement is then modulated by the bounding field $f_{bounding}$ which returns a scalar value $[0, 1]$. Note that this deformation field is actually the "reverse" deformation, so the sum takes $\mathbf{p}$ from its deformed to undeformed position. The constructions of the deformation field $\mathbf{D}$ and the bounding field $f_{bounding}$ are described in Section 5 and Section 6 respectively.

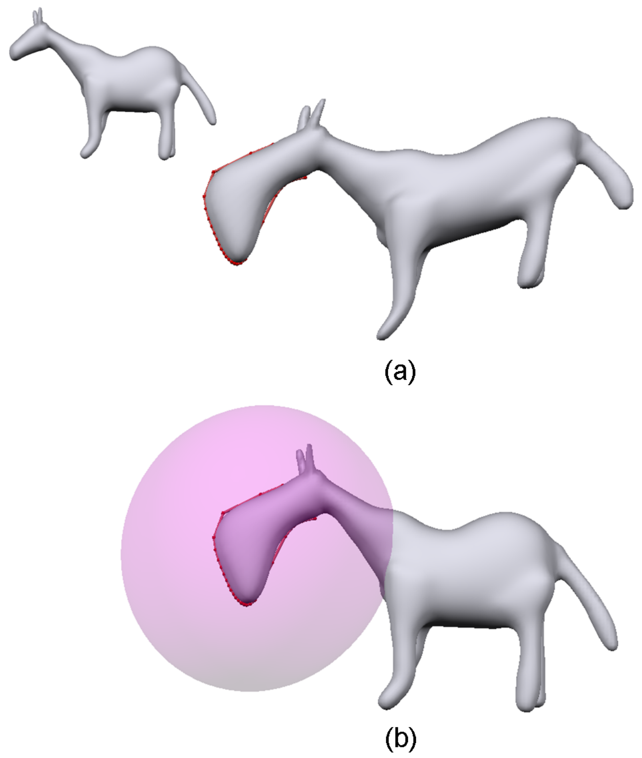


Figure 5: The curve constructs the deformation field. Since the deformation field has global influence, however, undesirable artifacts appear in (a). By applying the bounding field, the influence can be localized inside the bounding field (b). The bounding field is visualized with a transparent surface.

4. Hybrid Warp Node

Our deformation is represented by a composition node in the BlobTree, a unary *Warp* node which stores a *WarpCurve* as the deformation parameter (see Figure 6a). When the tree is traversed for a particular point $\mathbf{p}$ in space, the WarpCurve is used to determine the displacement of $\mathbf{p}$, and then the child of the Warp node is evaluated using this new position, yielding a scalar value. The problem with this structure is that once the warp is applied, the WarpCurve does not have influence if there are subsequent deformations, and as a result the curve handle can become detached from the surface (Figure 6a). While this may be desirable in some cases, in others the artist may find it confusing.

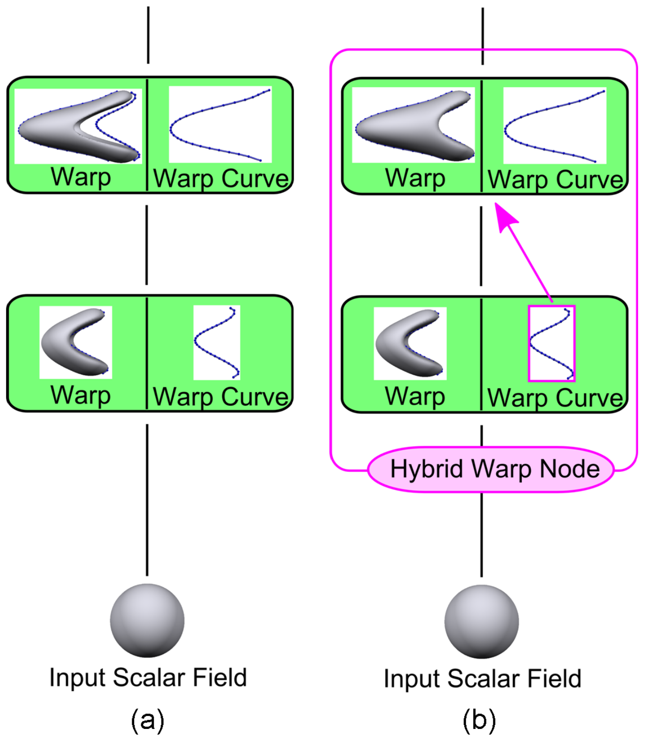


Figure 6: The BlobTree model without (a) and with (b) the HybridWarp node. In a HybridWarp node, the WarpCurves of the Warp nodes in the subtree are considered when applying the final Warp, so all the descendant curves remain on the surface. In this figure, a green box represents a Warp node where a WarpCurve is stored on the right side and the warp result is shown on the left side.

To deal with this problem we can optionally define a *Hybrid-Warp* node when multiple curves are applied. A HybridWarp node is a type of "super-node" that encapsulates multiple Warp nodes. Essentially, we do not modify the BlobTree structure, but the WarpCurves of the descendent Warp nodes in the subtree of a Warp node also work as constraints if the descendent Warp nodes are contained in the same HybridWarp node. An example of a HybridWarp node is shown in Figure 6b. In our implementation, we assume that if the artist applies multiple WarpCurve actions in sequence, the behavior of a HybridWarp node is intended, and so one is automatically added (Figure 7a). Any two sequentially-constructed Warp nodes are grouped into the same HybridWarp node, and as additional WarpCurves are added, they are added into this HybridWarp node. If another implicit modeling operator is applied, its node is inserted above the HybridWarp node and the HybridWarp is considered to be complete. Of course, the artist is also free to explicitly modify the tree, grouping or breaking up Warp nodes into HybridWarp nodes.

Evaluation of a HybridWarp node involves a special traversal, in which the descendant Warp nodes are processed sequentially. Conceptually, after each deformation is applied, we want the next deformation to preserve the position of all descendant warp curves. The traversals of two- and three-child Hybrid-Warp nodes are illustrated in Figure 7b, and performed as follows:

1. A Warp node WN is evaluated.
2. The subtree of WN is traversed until a node other than a Warp node is found.
3. The system recognizes that the traversed subtree is in a HybridWarp node and returns the WarpCurves within the HybridWarp node to WN. We call the returned WarpCurves *indirect warp curves* specified with pink arrows in Figure 7 while the stored WarpCurve of WN is a *direct warp curve*.
4. A deformation field $\mathbf{D}$ and bounding field $f_{bounding}$ are constructed using the direct warp curve and the indirect warp curves, and Equation 1 is evaluated in WN.

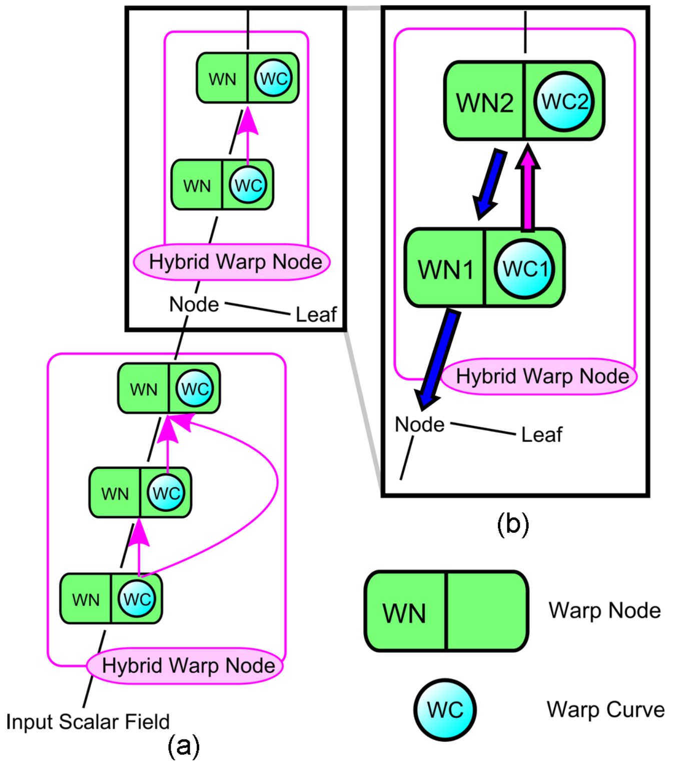


Figure 7: An example BlobTree structure with HybridWarp nodes (a). Two HybridWarp nodes are defined yet they do not have relationship, because another type of node is applied between the two HybridWarp nodes. (b) shows an example of traversing a HybridWarp node. When *WN2* is evaluated, *WC1* is returned to *WN2* as an indirect warp curve and deformation is constructed using *WC1* and *WC2*.

5. Deformation Field

In this section, we describe how to construct a deformation field $\mathbf{D}$ of Equation 1 using the direct warp curve and the indirect warp curves at each Warp node. A deformation field takes a sample point as input and returns the displacement as output. Since this deformation warps sample points in the scalar field of the original implicit model instead of warping the scalar field itself, the deformation field is an inverse warp (Figure 8). To create such a field, we interpolate the "negative" displacement vectors at each vertex of a curve. We denote the constraints for the interpolation as $(\mathbf{v}_i, -\mathbf{d}_i)$, where $\mathbf{d}_i$ is the displacement vector stored at a vertex $\mathbf{v}_i$ of a curve. This interpolation technique is described in Section 5.2. To enhance the deformation behaviors, we also define additional constraints along a curve. We call such constraints *off-curve constraints*. Since the indirect warp curves do not have displacements at the evaluated Warp node, zero displacements are assigned as their constraints which are denoted as $(\mathbf{v}_i, \mathbf{0})$ and do not have off-curve constraints. Zero displacements can prevent deformation of space in the assigned regions.

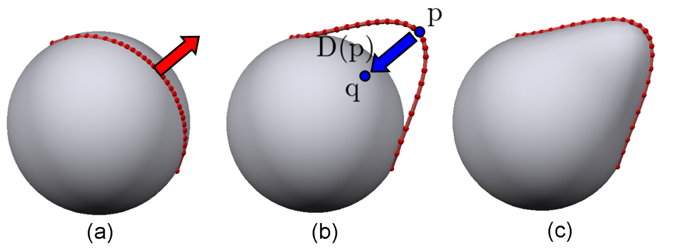


Figure 8: The deformation process of implicit surfaces. When a curve is edited (a), the system constructs a deformation field $\mathbf{D}$ which is an inverse warp. In (b), when a sample point $\mathbf{p}$ is evaluated, the deformation field $\mathbf{D}$ displaces $\mathbf{p}$ to $\mathbf{q} = \mathbf{p} + \mathbf{D}(\mathbf{p})$ and the field value of $\mathbf{q}$ is returned. (c) is the result of this deformation.

5.1. Defining Off-Curve Constraints

We define 4 off-curve constraints $\mathbf{oc}_i$ to each vertex $\mathbf{v}_i$ of a curve and assign them zero displacements $(\mathbf{oc}_i, \mathbf{0})$ to ensure that the amount of displacements decreases from the curve. Two of the off-curve constraints are placed along the normalized displacement vector $\hat{\mathbf{d}}_i$ of $\mathbf{v}_i$. The other off-curve constraints are along the orthogonal vector to $\hat{\mathbf{d}}_i$ and the curve. The 4 off-curve constraints are then defined as follows:

$$\mathbf{oc}_i = \begin{cases} \mathbf{v}_i \pm \Delta l_i \cdot \hat{\mathbf{d}}_i \\ \mathbf{v}_i \pm \Delta l_i \cdot (\hat{\mathbf{d}}_i \times \mathcal{T}(\mathbf{v}_i)) \end{cases} \quad (2)$$

where Δl_i is the distance between $\mathbf{v}_i$ and $\mathbf{oc}_i$, $\mathcal{T}$ is the trajectory of a curve. Since a curve is a 3D polyline composed of several vertices, $\mathcal{T}(\mathbf{v}_i)$ can be found by calculating $(\mathbf{v}_{i+1} - \mathbf{v}_i) + (\mathbf{v}_i - \mathbf{v}_{i-1})$ and normalizing it. Δl_i varies depending on the length of the displacement vector $\mathbf{d}_i$. The longer the length of $\mathbf{d}_i$ is, the longer Δl_i we set in order to increase the ROI. From several experiments we use $2\|\mathbf{d}_i\|$ as Δl_i. The locations of the off-curve constraints are illustrated in Figure 9.

Off-curve constraints sometimes cause a displacement interpolation error, particularly when Δl_i is long or the shape of a curve is sharp. That is because some of the off-curve constraints may intersect each other. To filter out such off-curve constraints, we find the shortest distance from an off-curve constraint $\mathbf{oc}_i$ to each vertex of a curve. If the ratio of the shortest distance and Δl_i (the distance between $\mathbf{oc}_i$ and its parent vertex $\mathbf{v}_i$) is smaller than the tolerance, we filter $\mathbf{oc}_i$ out. We choose as the tolerance ratio 2/3 which is also used in ShapeShop [27] to

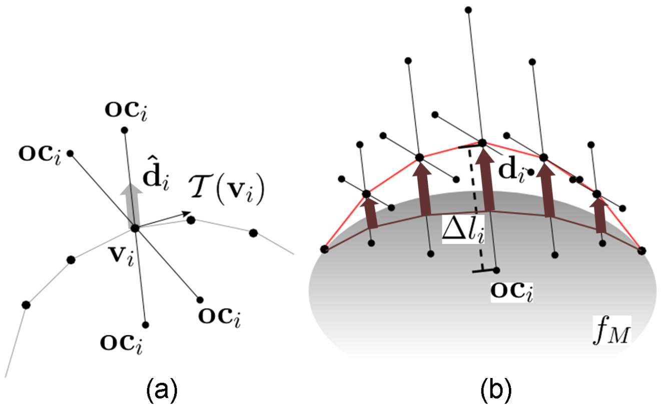


Figure 9: The locations of the off-curve constraints. 4 offset-curve constraints $\mathbf{oc}_i$ are added to each vertex $\mathbf{v}_i$ of a curve along the normalized displacement vector $\hat{\mathbf{d}}_i$ and the trajectory of the curve $\mathcal{T}(\mathbf{v}_i)$ (a). Δl_i is proportional to the length of the displacement $\mathbf{d}_i$ (b).

filter constraints during variational scalar field construction. A deformation field is constructed using the remaining off-curve constraints and the constraints at the curve vertices.

5.2. Variational Warp

We construct a deformation field using the variational warp technique [24, 6], which has the advantage of interpolating an arbitrary set of constraints with C^2 continuity. This interpolant is fit to the constraints placed at vertices of the direct and indirect warp curves, and the off-curve constraints of the direct warp curve. As described above, the off-curve constraints and the constraints of the indirect warp curves store zero vectors as displacements, preventing deformation of space in these regions. A deformation field $\mathbf{D}$ is then defined in terms of constraints $(\mathbf{v}_i, -\mathbf{d}_i)$:

$$\mathbf{D}(\mathbf{p}) = \sum_{i=1}^{m} \mathbf{w}_i \varphi(\|\mathbf{p} - \mathbf{v}_i\|) + \mathcal{P}(\mathbf{p}) \quad (3)$$

where $\mathbf{p}$ is a 3D sample point, m is the number of constraints, $\varphi(r) = r^3$, and $\mathcal{P}(\mathbf{p}) = \mathbf{p}_x\mathbf{c}_1 + \mathbf{p}_y\mathbf{c}_2 + \mathbf{p}_z\mathbf{c}_3 + \mathbf{c}_4$. The weights $\mathbf{w}_i \in \mathbb{R}^3$ and coefficients $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{c}_3$, and $\mathbf{c}_4 \in \mathbb{R}^3$ can be calculated by solving a dense linear system defined by the evaluation of Equation 3 at each known solution $\mathbf{D}(\mathbf{v}_i) = -\mathbf{d}_i$. A reasonable deformation field with a sufficient number of constraints can be maintained at interactive rates despite the $O(N^3)$ computational cost of the linear solve (see Section 7.1 for the performance details).

Since the variational technique has global influence, undesirable deformations occur in the regions where constraints are not placed, as can been seen in Figure 5a. To localize the deformation effects, therefore, we construct a bounding field (Section 6) which modulates the deformation field.

6. Bounding Field

In this section we describe how to construct the bounding field $f_{bounding}$ of Equation 1. This field is a scalar field which falls from a maximum value of one to zero at a finite distance. The function that generates the field takes a sample point as input and returns a scalar value $[0, 1]$ as output. We generate such a scalar field with convolution using the direct warp curve, which is a 3D polyline, as a skeleton. Note that unlike the deformation field the indirect warp curves are not used for the bounding field construction.

6.1. Convolution Field

The problem of using a polyline as a skeleton is that the resulting field will have bulges at joints [5]. Convolution can avoid the bulges even if several scalar fields are combined. A convolution field f is created by convolving a skeleton $\mathcal{S}$ with a kernel function h:

$$f(\mathbf{p}) = \int_{\mathcal{S}} h(\mathbf{p}, \mathcal{S}) d\mathcal{S} \tag{4}$$

We choose as our kernel function the *Cauchy* kernel [17, 30] which is often used to generate convolution surfaces having bounded scalar fields from polylines, such as [34, 4]:

$$h(\mathbf{p}, \mathcal{S}) = \frac{1}{(1 + s^2 r^2(\mathbf{p}, \mathcal{S}))^2} \tag{5}$$

where s is a tangent parameter controlling the kernel width and r is the distance between a sample point $\mathbf{p}$ and a skeleton $\mathcal{S}$. A convolution field of a polyline is generated by summing the convolution field of each line segment. Thus, we calculate Equation 4 by replacing the skeleton $\mathcal{S}$ with a line segment $L(t) = \mathbf{b} + t\hat{\mathbf{a}}, 0 \le t \le l$ ($\mathbf{b}$ is the base vector and $\hat{\mathbf{a}}$ is the normalized direction vector):

$$\begin{aligned} f_{line}(\mathbf{p}) = &\frac{1}{2p^2}\left[\frac{h}{s^2h^2 + p^2} + \frac{l-h}{s^2(l-h)^2 + p^2}\right] \\ &+ \frac{1}{2sp^3}\left[\tan^{-1}\left(\frac{sh}{p}\right) + \tan^{-1}\left(\frac{s(l-h)}{p}\right)\right] \end{aligned} \tag{6}$$

where $h = (\mathbf{p} - \mathbf{b}) \cdot \hat{\mathbf{a}}$ and $p^2 = 1 + s^2(||\mathbf{p} - \mathbf{b}||^2 - h^2)$. More details of this calculation can be found in [17, 34]. By summing the resulting scalar field f_{line} of each line segment, we can get a scalar field $f_{polyline}$ generated from a curve.

To control the size of a bounding field, we adjust a tangent parameter s which controls the kernel width. The size of a bounding field is proportional to how far the artist pulls a curve (Figure 10). The more the artist pulls the curve, the bigger bounding field is required to increase the ROI. Thus, we search for the longest displacement vector $d_{longest}$ from the vertices of the curve and determine the value of s according to the length of $d_{longest}$. Based on several experiments, we define $s = 1/||d_{longest}||$ which can create a bounding field with a reasonable size. Note that if $||d_{longest}||$ is equal to zero, a bounding field is not created.

Convolution can generate a scalar "falloff" field, which decreases to zero at a finite distance from the curve polyline, but the field values inside the scalar field cannot be controlled. We need to scale the range of field values to $[0, 1]$ in order to ensure that the deformation behaves in a predictable manner. We describe this field value adjustment in the next section.

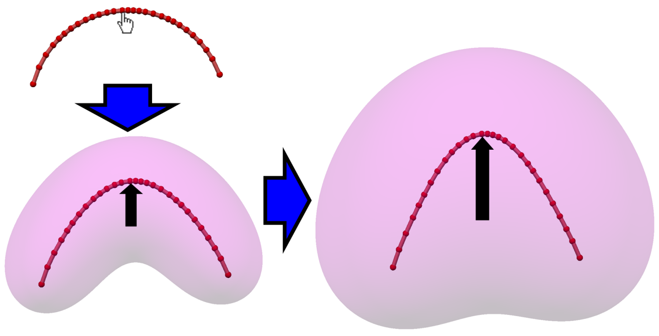


Figure 10: The convolution field generated from the direct warp curve. The more the artist pulls the curve, the larger the spatial extent of the constructed convolution field.

6.2. Field Value Adjustment

To be used for our deformation, the scalar bounding field must have a value equal to one at the curve and decrease to zero at some distance. This is necessary because the full vector displacement must be applied to vertices on the curve if the surface is to appear to track it. If the field value at curve is greater or less than one, the deformation is scaled and the deformed surface will become detached from the curve. Convolution does not have parameters to control the field values and the resulting scalar field of Figure 10 becomes like Figure 11a.

We adjust the field values of a bounding field using the *Wyvill* function [5]. A useful feature of the Wyvill function is that it has a zero first derivative at the point where a value reaches zero or one. That means we can smoothly clamp field values greater than one to one, or less than zero to zero. The Wyvill function is a falloff function which can generate a bounded scalar field by taking a distance field as input. We convert a convolution field $f_{polyline}$ generated in Section 6.1 into a distance field, and then apply the Wyvill function to the distance field. The resulting field is our bounding field $f_{bounding}$ which is defined as follows:

$$f_{bounding}(\mathbf{p}) = g_{wyvill}\left(1 - \frac{f_{polyline}(\mathbf{p})}{v_{one}}\right) \tag{7}$$

$$g_{wyvill}(x) = \begin{cases} 1 & x \le 0 \\ (1 - x^2)^3 & 0 < x < 1 \\ 0 & x \ge 1 \end{cases} \tag{8}$$

where g_{wyvill} is the Wyvill function and v_{one} is a field value of $f_{polyline}$ which we want to adjust to one. Since a bounding field should store one as a field value at a curve, we sample the field values of $f_{polyline}$ at the vertices of the curve and set v_{one} to the average of the sampled values. With this field value adjustment, we can get a bounding field whose range is $[0, 1]$ (Figure 11b). The resulting bounding field $f_{bounding}$ is replaced in Equation 1 and smoothly modulates a deformation field as shown in Figure 5b.

7. Results

The WarpCurves tool lets the artist to efficiently create and modify features and details on implicit models. In particular,

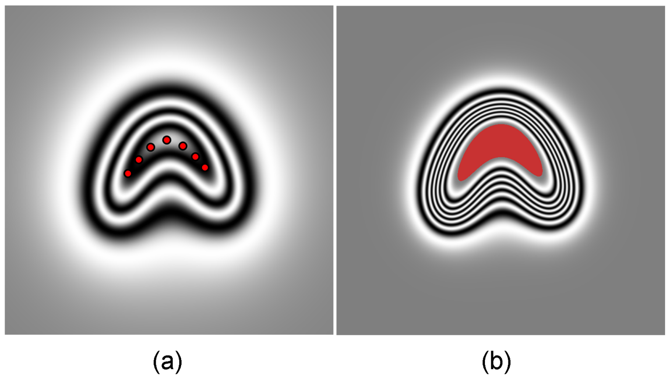


Figure 11: The scalar field generated with convolution (a). By sampling the field values at the vertices of the curve, the value v_{one} is computed. (b) is the scalar field which (a) is converted into using the Wyvill function. The part where the field value is equal to one is highlighted in red. As can been seen, the field values near the curve are one after conversion.

cavities and projections can be easily added using WarpCurves. The ear model shown in Figure 12 was created from a simple flat implicit shape using 3 curves. The features of the ear model were easily added just by manipulating the curves. Adding such features to an implicit model in a controllable and predictable way is virtually impossible with the existing compositional modeling tools available in ShapeShop.

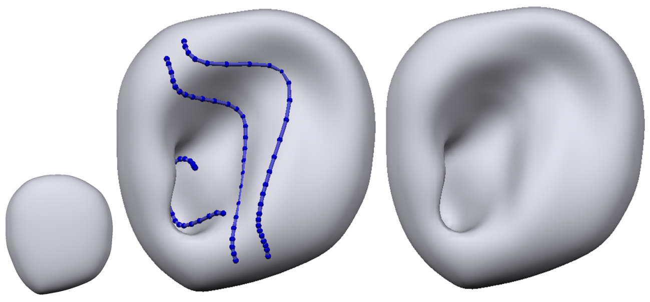


Figure 12: The ear model. The ear model was created by adding details on the flat implicit shape using 3 curves.

The face model shown in Figure 1 demonstrates how WarpCurves can be used to efficiently create large-scale features which would take multiple steps with compositional operators. The advantage of an implicit approach is that composition can be easily integrated with deformation. Since our technique is implemented as an operator in the BlobTree, deformations are hierarchically defined and other operators such as blending and CSG can be applied before or after deformation. In Figure 1, the ears, the mouth, and the eyes were added to the face model using blending and CSG operators after creating the general head shape with WarpCurves.

One significant drawback of compositional modeling is that it can be very difficult to edit existing models. Deformation is often more efficient for editing tasks. The WarpCurves tool allows existing BlobTree models to be quickly deformed. For example, the dog model shown in Figure 13a is an implicit model created with ShapeShop [27]. The face and the body of the dog model were deformed using WarpCurves (Figure 13b). In this example, a BlobTree model is used as an input implicit model, but WarpCurves are not limited to BlobTree models. Our technique is applicable to any implicit model as input, and effectively constructs a BlobTree structure using the input implicit model as a leaf node.

We also asked an artist to experiment with WarpCurves during a 3D modeling session, the results are shown in Figure 14. The models were created from scratch using ShapeShop's sketch-based composition operators and WarpCurves. The main contribution of WarpCurves is to support explicit manipulation of implicit surfaces. When the artist wants to make modifications to a model, he or she usually does not think to blend some volumes, but rather to "push or pull" the surface. Our WarpCurves tool allows the artist to interact with the surface in a more natural way. For example, the mouth of Figure 14a and the feelers of Figure 14b could be created by blending a set of simpler volumes. Using WarpCurves, however, they can be created by directly manipulating the surface.

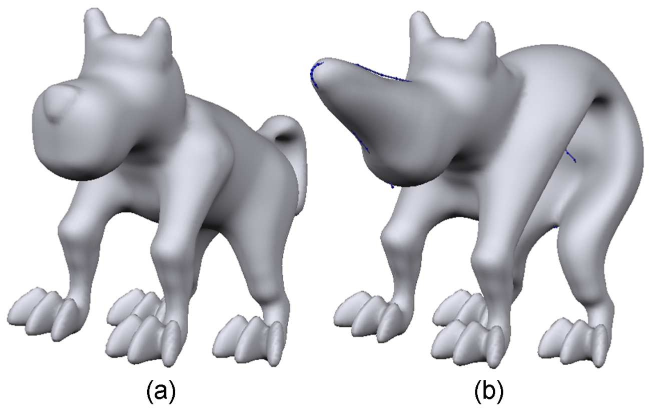
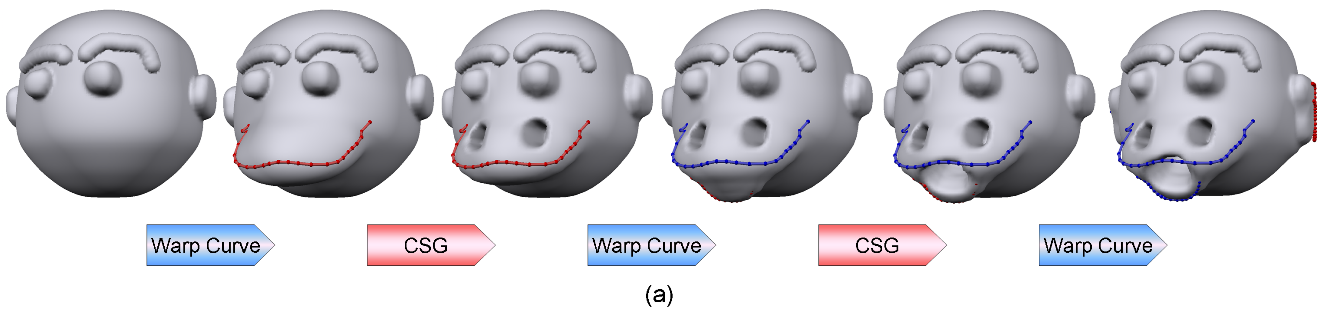


Figure 13: The dog model. The dog model (a) is an input implicit model which was created with ShapeShop. The artist can edit an existing implicit model using curves (b).

7.1. Implementation Details

Compared to parametric or mesh modeling, interactive implicit modeling is computationally intensive because we must literally search through space at each frame to visualize the surface. Due to this, many trade-offs must be made to maintain minimal levels of visual fidelity.

In our implementation, an individual curve handle contains approximately 20 vertices. Higher sampling rates can be used to increase control and improve how closely the surface tracks the curve, at the cost of interactivity. We provide the artist with a slider to manipulate this trade-off. Since computing a variational warp (Section 5.2) is so expensive, we improve interactivity by only using every third set of off-curve constraints along the curve, so one curve contains roughly 45 constraints in total as the default set. The number of off-curve constraints varies during deformation because some may be filtered out to avoid a displacement interpolation error. The poly-line convolution used to construct our bounding field (Section 6.1) is also



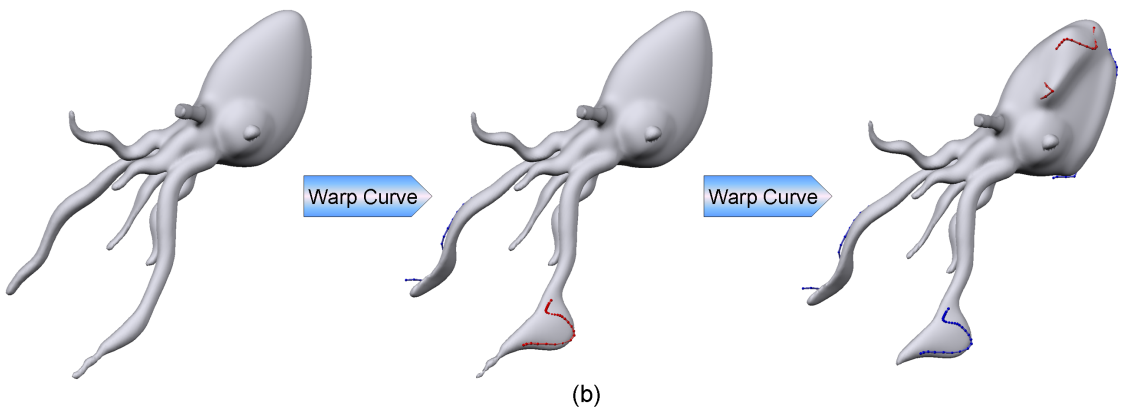


Figure 14: These models were created by an artist. The mouth of the model (a) was created by pulling out the surface using WarpCurves. The feelers of the model (b) were also created using WarpCurves. Because WarpCurves are integrated into a procedural implicit modeling framework, blending and CSG operators were intermixed with deformation during modeling.

a computationally expensive technique. We automatically reduce the number of line segments via curve simplification to reduce this cost. Empirically, we found that reducing to 4-6 line segments was sufficient for most deformations.

The recursive evaluation of the BlobTree structure becomes increasingly expensive as multiple WarpCurves are applied. After a deformation is completed, we insert a cache node [26] above the Warp or HybridWarp node to reduce the evaluation time. Hence, during interactive curve manipulation, once the cache is populated only the top Warp node is actually evaluated. This is critical to maintain interactivity.

Table 1 shows the performance of WarpCurves. All timings in Table 1 were measured on a workstation with Intel 2.66GHz Core 2 Quad CPU and 2.75GB of RAM. ShapeShop is not multi-threaded, and we note that a re-polygonization of the entire model is computed at each frame. The polygonization resolution was fixed at a cube size of 0.075. The performance was evaluated using the models shown in this paper: the point primitive (Figure 2), the face model 1 (Figure 1), the ear model (Figure 12), the horse model (Figure 5), the squid model (Figure 14b), the face model 2 (Figure 14a), and the dog model (Figure 13). The table shows the number of constraints for variational warping, the voxel dimensions of polygonization, the number of vertices after polygonization, and the frame rates (fps). We show the approximate data in the table, because they vary during deformation. Note that the models shown in this paper were rendered using high-resolution polygonizations generated after interactive modeling was completed.

	Constraints	Voxel Dimensions	Vertices	fps
Point	55	22x22x22	360	30
Face1	104	22x25x27	444	21
Ear	89	34x34x25	650	16
Horse	44	51x37x24	946	15
Squid	127	88x85x57	1350	7
Face2	85	38x36x99	1476	6
Dog	75	110x109x85	7380	2

Table 1: The timings of deformations.

8. Limitations

The WarpCurves tool demonstrates direct manipulation for implicit surfaces but there are also some limitations. A major limitation of our tool is that deformation cannot be controlled geodesically. Since our deformation is ultimately a spatial deformation, the amount of deformation influence is dependent on the Euclidean distance from the curve. The curve may strongly influence portions of the implicit volume which are geodesically far from the curve yet nearby in Euclidean distance. In this case unexpected deformations will occur (Figure 15). This could be avoided by creating a more complex bounding field, or by ensuring that the deformation constraints do not penetrate geodesically distant portions of the surface. Note, however, that in the implicit domain the deformation needs to influence a support region around the local surface, so that blends applied after deformation are smooth and predictable. This generally means that a larger bounding field is required than what one might intuitively expect.

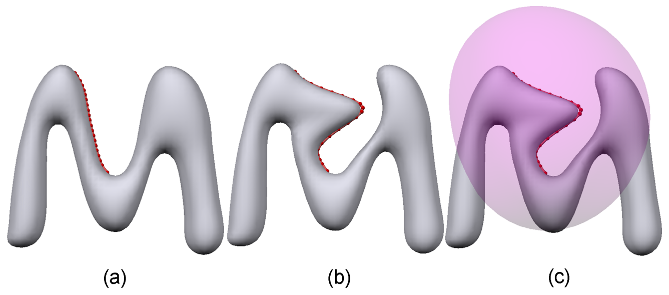


Figure 15: Deformation is constructed according to the Euclidean distance, so the part which is geodesically far from the warp curve yet close in the Euclidean distance also gets strong deformation influence (b). A bounding field inflates depending on the Euclidean distance as well, so it cannot localize such deformation influence (c).

Another limitation is the curve manipulation. During curve editing, the edited part of the curve is not allowed to collide with other curves, or to self-collide. If such collisions occur, the variational warp will diligently interpolate the constraints, resulting in bizarre and unexpected deformations. Although two curves collide with each other in Figure 1, the collision point is not edited. Also, a HybridWarp node allows the WarpCurves of the descendent Warp nodes to have influence at the ascendent Warp nodes but the descendent Warp node is not aware of any WarpCurves of parent Warp nodes. This means that if the artist draws two curves and manipulates the first drawn curve, the second drawn curve is detached from the surface. One way to avoid this problem would be to apply the forward-deformation of the child node to any parent curves within a HybridWarp node, perhaps followed by a gradient walk to make sure the curve lies on the surface. However, this would require solving for two variational warps instead of one, and in practice we did not find that it was necessary.

9. Conclusion and Future Work

We have presented WarpCurves, a curve-based deformation technique for implicit surfaces. A key benefit of our tool is that the artist can explicitly manipulate implicit models using curves. Direct manipulation is often a more efficient way to add and edit features and details of 3D models. Functional implicit surfaces are a representation suitable for shape composition via blending and CSG, but to date it has not been possible to directly manipulate the iso-surface. Our technique provides an approximate solution to this problem.

Our method is based on spatial deformation, but by constructing appropriate inverse warps according to curve manipulation, the WarpCurves tool lets the artist interact with implicit models as if manipulating the surfaces directly. Because of curve-based deformation, features and details can be easily added to implicit models. Our curve-based deformation technique has been integrated into the BlobTree procedural implicit model framework as an operator, which allows the artist to apply blending and CSG operators before or after deformation. These benefits have been shown in our results.

Our future work is to edit textured implicit models using WarpCurves. Texturing for implicit models is often defined using local RGB fields [35]. Since our deformation technique is spatial, the texture fields could also be deformed along the curves. If the modified curve is constrained to the surface, then the deformation could also be restricted to the texture field, providing a useful texture editing tool. Our deformation technique can also be applied to point-based surfaces by changing inverse warps into forward warps. This would essentially be a variant of Wires [31], however by using variational warps and smooth bounding fields we avoid some of the continuity limitations of Wires.

Acknowledgements

The authors would like to thank Aya Walraven for model creation.

References

[1] Angelidis, A., Cani, M.-P., Wyvill, G., King, S., 2004. Swirling-sweepers: Constant-volume modeling. In: Proc. PG '04. pp. 10–15.
[2] Barr, A. H., 1984. Global and local deformations of solid primitives. In: Proc. SIGGRAPH '84. pp. 21–30.
[3] Ben-Chen, M., Weber, O., Gotsman, C., 2009. Variational harmonic maps for space deformation. ACM Trans. Graph. 28 (3).
[4] Bernhardt, A., Pihuit, A., Cani, M.-P., Barthe, L., 2008. Matisse: Painting 2D regions for modeling free-form shapes. In: Proc. SBIM '08. pp. 57–64.
[5] Bloomenthal, J., 1997. Introduction to Implicit Surfaces. Morgan Kaufmann, ISBN 1-55860-233-X.
[6] Botsch, M., Kobbelt, L., 2005. Real-time shape editing using radial basis functions. Computer Graphics Forum 24 (3), 611–621.
[7] Coquillart, S., 1990. Extended free-form deformation: a sculpturing tool for 3d geometric modeling. In: Proc. SIGGRAPH '90. pp. 187–196.
[8] Crespin, B., 1999. Implicit free-form deformations. In: Proc. Implicit Surfaces '99. pp. 17–23.
[9] Eyiyurekli, M., Grimm, C., Breen, D., 2009. Editing level-set models with sketched curves. In: Proc. SBIM '09. pp. 45–52.
[10] Gain, J., Bechmann, D., 2008. A survey of spatial deformation from a user-centered perspective. ACM Trans. Graph. 27 (4), 1–21.
[11] Gal, R., Sorkine, O., Mitra, N. J., Cohen-Or, D., 2009. iWires: an analyze-and-edit approach to shape manipulation. ACM Trans. Graph. 28 (3).
[12] Igarashi, T., Moscovich, T., Hughes, J. F., 2005. As-rigid-as-possible shape manipulation. ACM Trans. Graph. 24 (3), 1134–1141.
[13] Joshi, P., Meyer, M., DeRose, T., Green, B., Sanocki, T., 2007. Harmonic coordinates for character articulation. ACM Trans. Graph. 26 (3), 71.
[14] Ju, T., Schaefer, S., Warren, J., 2005. Mean value coordinates for closed triangular meshes. ACM Trans. Graph. 24 (3), 561–566.
[15] Lipman, Y., Levin, D., Cohen-Or, D., 2008. Green coordinates. ACM Trans. Graph. 27 (3).
[16] MacCracken, R., Joy, K. I., 1996. Free-form deformations with lattices of arbitrary topology. In: Proc. SIGGRAPH '96. pp. 181–188.
[17] McCormack, J., Sherstyuk, A., 1998. Creating and rendering convolution surfaces. Computer Graphics Forum 17 (2), 113–120.
[18] Milliron, T., Jensen, R. J., Barzel, R., Finkelstein, A., 2002. A framework for geometric warps and deformations. ACM Trans. Graph. 21 (1), 20–51.
[19] Nealen, A., Igarashi, T., Sorkine, O., Alexa, M., 2007. Fibermesh: designing freeform surfaces with 3d curves. ACM Trans. Graph. 26 (3).
[20] Ono, Y., Chen, B.-Y., Nishita, T., Feng, J., 2002. Free-form deformation with automatically generated multiresolution lattices. In: Proc. CW '02. pp. 472–479.

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

[22] Pasko, G., Pasko, A., Kunii, T., 2005. Bounded blending for function-based shape modeling. IEEE Computer Graphics and Applications 25 (2), 36–45.

[23] Savchenko, V., Pasko, A., 1998. Transformation of functionally defined shapes by extended space mappings. The Visual Computer 14 (5/6), 257–270.

[24] Savchenko, V., Pasko, A., Kunii, T., Savchenko, A., 1995. Feature based sculpting of functionally defined 3d geometric objects. Multimedia Modeling: Towards Information Superhighway, World Scientific, Singapore, 341–348.

[25] Schmidt, R., Singh, K., 2008. Sketch-based procedural surface modeling and compositing using Surface Trees. Computer Graphics Forum 27 (2), 321–330.

[26] Schmidt, R., Wyvill, B., Galin, E., 2005. Interactive implicit modeling with hierarchical spatial caching. In: Proc. SMI '05. pp. 104–113.

[27] Schmidt, R., Wyvill, B., Sousa, M., Jorge, J., 2005. Shapeshop: Sketch-based solid modeling with blobtrees. In: Proc. SBIM '05. pp. 53–62.

[28] Schmitt, B., Pasko, A., Schlick, C., 2003. Shape-driven deformations of functionally defined heterogeneous volumetric objects. In: Proc. GRAPHITE '03. pp. 127–134.

[29] Sederberg, T. W., Parry, S. R., 1986. Free-form deformation of solid geometric models. In: Proc. SIGGRAPH '86. pp. 151–160.

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

[31] Singh, K., Fiume, E., 1998. Wires: a geometric deformation technique. In: Proc. SIGGRAPH '98. pp. 405–414.

[32] Sorkine, O., Botsch, M., 2009. Tutorial: Interactive shape modeling and deformation. In: Eurographics.

[33] Sugihara, M., de Groot, E., Wyvill, B., Schmidt, R., 2008. A sketch-based method to control deformation in a skeletal implicit surface modeler. In: Proc. SBIM '08. pp. 65–72.

[34] Tai, C.-L., Zhang, H., Fong, J. C.-K., 2004. Prototype modeling from sketched silhouettes based on convolution surfaces. Computer Graphics Forum 23 (1), 71–83.

[35] Tigges, M., Wyvil, B., 1999. A field interpolated texture mapping algorithm for skeletal implicit surfaces. In: Proc. CGI '99. pp. 25–33.

[36] von Funck, W., Theisel, H., Seidel, H.-P., 2006. Vector field based shape deformations. ACM Trans. Graph. 25 (3), 1118–1125.

[37] Wyvill, B., Guy, A., Galin, E., 1999. Extending the CSG tree. warping, blending and boolean operations in an implicit surface modeling system. Computer Graphics Forum 18 (2), 149–158.

[38] Wyvill, B., van Overveld, K., 1997. Warping as a modelling tool for csg/implicit models. In: Proc. SMA '97. pp. 205–214.