A Level-Set Approach for the Metamorphosis of Solid Models

David E. Breen† Ross T. Whitaker‡

†Computer Graphics Laboratory
MS 348-74
California Institute of Technology
Pasadena, CA 91125
Tel (626) 395-2866
FAX (626) 793-9544
david@gg.caltech.edu
corresponding author

‡School of Computing
4540 Merrill Engineering Building
University of Utah
Salt Lake City, UT 84112-9205
Tel (801) 587-9549
FAX (801) 581-5843
whitaker@cs.utah.edu

October 30, 2000

Abstract

This paper presents a new approach to 3-D shape metamorphosis. We express the *interpolation* of two shapes as a process where one shape deforms to maximize its similarity with another shape. The process incrementally optimizes an objective function while deforming an implicit surface model. We represent the deformable surface as a level set (iso-surface) of a densely sampled scalar function of 3 dimensions. Such *level-set* models have been shown to mimic conventional parametric deformable surface models by encoding surface movements as changes in the greyscale values of a volume dataset. Thus, a well-founded mathematical structure leads to a set of procedures that describes how voxel values can be manipulated to create deformations that are represented as a sequence of volumes. The result is a 3-D morphing method that offers several advantages over previous methods including minimal need for user input, no model parameterization, flexible topology, and sub-voxel accuracy.

Index Terms: Level set method, morphing, solid model, distance function, animation, volume graphics, optimization, deformable model

1 Introduction

Shape metamorphosis is a process where an object continuously changes its own shape into the shape of another object. Within the computer graphics community *morphing* (the vernacular for metamorphosis) has been used frequently for special effects in movies, advertising, and entertainment. *Image morphing* takes a 2-D image of an object and transforms the appearance of that object into the appearance of another object, with the goal of producing natural-appearing, or at least sensible, intermediate images. *Three-dimensional morphing* or *shape morphing* involves smoothly changing the model of one object into the model of another object. Shape morphing algorithms have been developed for both surface models and volumetric models. The surface model algorithms transform the surface patches (usually polygons) of the source model into the surface patches of the target model. The volume-based morphing algorithms represent 3-D objects as volumes and manipulate the voxel values of volumes in order to make one object become another. Volume datasets needed for this process may be acquired directly from 3-D scanning devices, such as MRI or CT, or they may be generated via 3-D scan conversion of solid geometric models.

Despite the increased complexity and computation time associated with morphing 3-D shapes rather than 2-D images, shape morphing does have some distinct advantages. First, image morphing is directly tied to the views of the two input images. Any morphing effect must be based only on the information available in the two images. This prevents the animator from making camera, lighting or shading changes during the morph. If the morph is performed with 3-D techniques, the animator is provided with much more flexibility in presenting the results of the morph. Once a sequence of transforming models has been generated, the animator may experiment with a variety of camera angles and motions for viewing the models. The scene's global lighting and shading parameters may also change during the morph. A second advantage is that a 3-D morph ensures that the resulting rendered images represent continuous sequence of actual 3-D shapes. In image morphing, maintaining the 3-D feasibility of the intermediate images is the responsibility of the user, i.e., care must be taken to make sure intermediate images represent pictures of feasible objects rather than a fuzzy mixture of ghost-like objects.

One disadvantage of 3-D morphing is that it requires two 3-D models. It is not always possible to acquire accurate models of the objects that one wishes to morph, e.g., a sequence of familiar faces. At present, digital photographs of real objects tend to capture more visual detail than 3-D models of the same. Despite this, there is a class of morphing problems, for which 3-D models exist or are easily obtainable, which lend themselves to the application of shape morphing.

In order to compare the adequacy of different approaches to 3-D shape morphing, we have identified several desirable aspects of a 3-D morphing technology. These properties are:

1. The transition process should begin with a *source* surface and end with a specified *target* surface.
2. Intermediate surfaces should undergo continuous 3-D transitions (rather than continuity only in the image space).
3. The morphing algorithm should apply to a wide range of shapes and topologies.
4. A 3-D morphing algorithm should incorporate user input easily but should degrade gracefully without it.
5. Transitional shapes should depend only on the surface geometry of the two input shapes and user input.

These requirements are not exhaustive, but they capture many of the practical aspects of 3-D morphing.

In this paper we present a new approach to 3-D model morphing. We employ an active deformable surface which starts at the source shape and smoothly changes into the target shape. This deformation process is described by the optimization of an objective function that measures the similarity between the target and the deforming surface. We represent the deformable surface as a level set (iso-surface) of a densely sampled scalar function of 3 dimensions. The sampling produces a regularly-spaced rectilinear volume dataset. Such *level-set* models [37, 41] have been shown to mimic conventional parametric deformable surface models by encoding surface movements as changes in the greyscale values of the volume. A well-founded mathematical structure leads to a set of procedures that describes how voxel values can be manipulated to create deformations in the level sets. The result is a voxel-based modeling technology that offers several advantages over previous methods, including support for a wide range of user input, no need for parameterization, flexible topology, and sub-voxel accuracy. This paper describes the application of this technology to 3-D surface morphing.

This work, as with several other volumetric morphing techniques [11, 26], is based on the image morphing strategy, and consists of two distinct steps. The first step is a *global, geometric warping*, which utilizes a

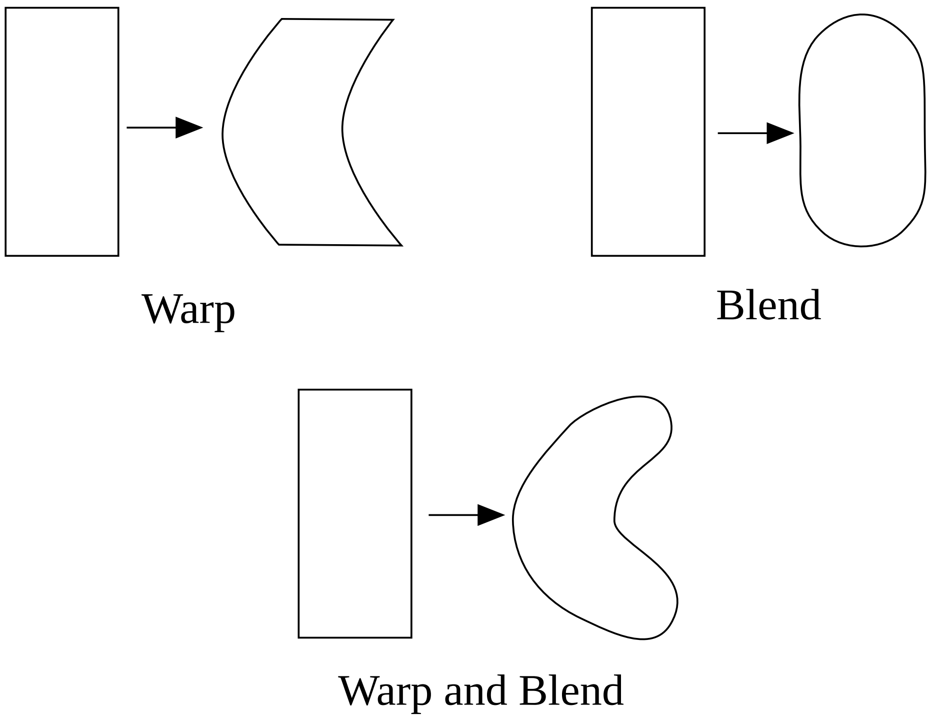

Figure 1: Shape morphing, like image morphing, can be described as a warping combined with a blending.

coordinate transformation that maps the source model into approximately the same shape and orientation as the target model, as shown in Figure 1. This coordinate transformation can take a variety of different forms, but the literature has shown that a rigid transformation combined with a non-rigid, spline-based, coordinate map is quite effective. The non-rigid transformation is usually determined by a set of user-defined correspondences (i.e., fiducials) between the source image/volume and the target. The second step is a *blending* that completes the morph by ensuring that any discrepancies that remain in the images/volumes after the warping are gradually removed over the course of the morphing process. In image morphing and volumetric shape morphing, the blending step is usually achieved by a simple linear interpolation, or cross-dissolve, of the two images/volumes. The warping stage of image/volume morphing has been studied extensively by other researchers, and our work primarily focuses on developing a superior method for the blending stage, one which reduces the amount of user input required to produce an acceptable morphing result. Indeed, the proposed level-set approach to shape metamorphosis will produce a reasonable morph with virtually no user input. Given this technology, user input can be incorporated in an incremental fashion starting with a minimal amount and proceeding with progressively more until a satisfactory result is achieved.

Our morphing approach consists of several stages. First, the source and target objects are defined, with

Constructive Solid Geometry (CSG) models in our examples. The models are scan converted into 3-D distance volumes (a signed distance transform), where the shortest distance to the solid model is stored at each voxel. We use the sign convention that distance is positive inside the object, and negative outside the object. A level-set model is fit to the zero level set of the source model distance volume. A coordinate transformation provides a mapping from every point in the domain of the level-set model into the distance volume of the target model. This is the warping step. The level-set model is then allowed to deform using the signed distance transform of the target model. Each point on the source surface moves in the direction normal to the surface at that point with a velocity proportional to the signed distance transform of the target object at that point in 3-space. This process is the *blending* step of the 3-D surface morph. At user-defined time intervals the level-set model is converted into a polygonal model. This model is expressed in the coordinate system of the source model. Therefore the coordinate transformation is incrementally applied, as a function of time, to the sequence of polygonal models, which are then rendered to produce the frames of the morphing animation.

In general, an animator controls the morph by defining how the models overlap, using a variety of coordinate transformations. This can include rigid transformations, scalings, and non-rigid transformations determined by sets of user-defined fiducials. Our current implementation also provides an automated mechanism for determining rotation, translation, and non-uniform scaling, and thereby offers the *option* of a totally automatic morphing algorithm. The system accomplishes this by overlaying the centroids of the two models, aligning their principal axes, and non-uniformly scaling along these axes. The surface of the level-set model only moves in those areas where the target and the source surfaces are not brought perfectly together by the coordinate transformation. The portion of the level-set model which is outside of the target model will shrink, while the portion inside the target model will expand to become the final shape.

The level-set approach to 3-D model morphing provides several advantages over other 3-D morphing algorithms. Level-set models are active; their underlying motion is defined algorithmically rather than interactively, guaranteeing that the source shape will become the target shape, given that both models initially overlap. The amount of user input required to produce a morph is directly proportional to the amount of control the animator wishes to impose on the process. The animator may allow the system to automatically generate the morph, or he/she can employ a full 3-D warping to describe the morph, with the level sets simply providing a small fine-tuning of the surface. Any intermediate amount of user input will produce a reasonable morph. Because level-set models do not rely on any kind of parameterization, they do not suffer the problems of parametric surfaces, e.g. a limited set of possible shapes and the need for reparameterization after undergoing significant changes in shape. This lack of parameterization, along with no direct representation of topological structure, allow level-set models to easily change topology while morphing. The model can "split" into pieces to form multiple objects. Conversely several disjoint objects may come together to make a single object. Finally, both the scan conversion and deformation stages of our morphing approach produce models with user-defined sub-voxel accuracy. This provides a flexible time-quality trade off and generates superior results at low volume resolutions as compared to previous work.

The remainder of the paper proceeds as follows. The next section describes related morphing work. The section following that describes the novel aspect of our method—shape interpolation based on the optimization of a similarity function using level-set models. It is followed by a discussion of general properties and numerical methods. Section 6 describes all stages of our morphing approach. The paper closes with three morphing examples, conclusions, and some thoughts on future work.

2 Related Work

In recent years numerous 3-D morphing algorithms have been reported in the literature. These algorithms generally fall into two categories, surface-based approaches and volume-based approaches. The former primarily consists of those methods that continuously change one polygonal surface into another. Volumetric methods modify the voxel values of a volume dataset in order to smoothly transform the object defined by the volume from a source shape into a target shape. A third class of algorithms involves morphing implicit models, either by transforming the underlying structure of "blobby" models [14], or by creating higher dimensional interpolating implicit functions [39].

Kent et al. [23] described the fundamentals for morphing 3-D polygonal surfaces. The first step is to create a single topological description of vertices, edges and faces, which contains the combined topological structure of both the source and target surfaces (the correspondence phase). The vertices of the structure are then interpolated between their position on the initial surface to their position on the target surface to create the morphing result. Parent [31] presents an improved technique for establishing vertex and edge correspondences when creating the combined topological structure. Chen et al. [9] applied the same approach to polygonal surfaces defined in cylindrical coordinates. Kanai et al. [21] use harmonic maps to define the correspondences in the combined topological structure. Lazarus and Verroust [24] do not build a common adjacency graph, but instead create a common parameterized mesh during a sampling process. Gregory et al. [17] present a method to assist the user in defining these correspondences. Lee et al. [25] apply many of these surface-based methods to morphing multiresolution meshes. Rossignac and Kaul [22, 34] introduce the notion of an interpolating polyhedron, an application of mathematical morphology to face sets. They also describe a user control methodology based on a Bezier curve paradigm. DeCarlo and Gallier [12] demonstrate that it is possible (with significant effort) to morph polygonal models of differing genus. Surface-based methods are important because of the wealth of polygonal models available to animators, but they are burdened with the difficult task of creating a single topological structure which can represent the source and target surfaces. While it has been shown that changing the genus of a morphing polygonal surface is possible, for example from a sphere to a torus, it can be a difficult and tedious process which requires significant user input. Another troubling feature of surface-based methods is the problem of self-intersection. These methods cannot guarantee that polygonal surfaces will not pass through themselves, creating physically nonsensical intermediate results.

Embedding the morphing surfaces in volumes alleviates these problems. Hughes [19] demonstrates how volumes can be used to create a morph between objects of different genus. The approach involves linearly interpolating the Fourier transforms of the volumes. A schedule is used during interpolation which filters out the high frequencies of the initial volume, while interpolating the low frequencies of both volumes, and gradually adds the high frequencies of the target volume. He et al. [18] propose a similar approach using wavelets to control the high-frequency artifacts. They also developed methods that allow the user some control of the interpolation by establishing explicit correspondences between the volumes. Lerios et al. [26] describe a morphing method which is a 3-D extension of Beier and Neely's [4] 2-D (image) morphing technique. The first step is to apply a user-defined geometric warp to the source object in order to deform it into approximately the same shape as the target object. This step allows the user to specify corresponding features on the two objects. The second step interpolates the values between matching voxels in the (warped) source and target volumes. Chen et al. [10] propose disk fields as a superior instrument for specifying the warps needed for this kind of 3-D morphing. Payne and Toga [32] describe a method for changing one volumetric model into another by interpolating the distance fields generated from the two volumes. A distance field, or distance volume, is a volume dataset where the value stored at each voxel is the shortest distance to the surface of the object being represented by the volume. Cohen-Or et al. [11] improve upon this approach by including a two-part warping step that calculates a rigid transformation and an elastic warping based upon user-supplied anchor points. This extra step approximately aligns the target and final objects, and provides significant user control to the overall morphing process.

The proposed volume-based morphing technique shares many of the advantages of previous volume-based methods. It can easily morph objects of different genus. It is not burdened with the difficult vertex/edge/face book-keeping of surface methods, and surfaces cannot pass through each other. Additionally, our method provides a novel blending/interpolation mechanism that will not produce the ghosting or spontaneous generation of new objects which is possible when simply interpolating voxel intensity or distance values (for example, see Figures 4, 5 and 6). Our method does not require user input to produce a reasonable morph, but can easily incorporate previously published warping techniques to provide a wide range of animator control. Finally, because we calculate the distance transform to sub-voxel accuracy, and effectively track a deforming surface within voxels our results do not produce the aliasing artifacts commonly found in other distance-based, volumetric approaches.

3 Metamorphosis As A Goal-Driven Process

Morphing strategies may be built upon any number of underlying principles. For instance, surface metamorphosis could consist of a sequence of time slices from a four-dimensional manifold which optimizes some space-time criteria [39]. The proposed strategy for object metamorphosis is based on yet another principle: shape metamorphosis is the process by which one object seeks to *resemble* another. This philosophy raises two questions. First, what is the metric by which we can quantify the similarity of two objects? Second, what is the process by which one object seeks to optimize that metric? This paper shows that even very simple answers to these two questions produce very powerful algorithms for shape metamorphosis, with a great deal of opportunity for future enhancement and refinement of the resulting algorithms.

In order to create a very general algorithm for metamorphosis, we work with a very general notion of a surface. Consider an open set $\Omega_A \subset \mathbb{R}^3$, which is the source object, and a target $\Omega_B \subset \mathbb{R}^3$. The source, Ω_A, is enclosed by a surface $\mathcal{S}_A = \partial\Omega_A$, and likewise $\mathcal{S}_B = \partial\Omega_B$. We require that the objects be compact and lie in some finite domain $U \subset \mathbb{R}^3$. Notice that we do not require any specific connectivity or topology, which means that each object could consist of a set of disconnected pieces (each with any number of holes) all sitting in U.

We propose a very simple metric for comparing two shapes that maximizes the volume shared by the interiors of the two objects. First define an inside-outside function for the target, $\gamma_B : \mathbb{R}^3 \mapsto \mathbb{R}$ for Ω_B, such that

$$\begin{aligned} \gamma_B(\mathbf{x}) &= 0 \quad \forall\, \mathbf{x} \in \mathcal{S}_B \\ \gamma_B(\mathbf{x}) &> 0 \quad \forall\, \mathbf{x} \in \Omega_B \\ \gamma_B(\mathbf{x}) &< 0 \quad \text{otherwise} \ . \end{aligned} \tag{1}$$

The inside-outside function γ_B can be used to quantify the extent to which an intermediate object Ω_t overlaps with the target, Ω_B, with the volume integral

$$\mathcal{M}_{\Omega_B,\Omega_t} = \int_{\Omega_t} \gamma_B(\mathbf{x}) \mathrm{d}\mathbf{x}. \tag{2}$$

Notice, this integral achieves its maximum,

$$\Omega_m = \arg\max_{\Omega_t} \left(\int_{\Omega_t} \gamma_B(\mathbf{x}) \mathrm{d}\mathbf{x} \right) \tag{3}$$

when $\Omega_m = \Omega_B$, because in that case the integral includes all of the positive parts of γ_B and none of the negative parts.

We can compute the first variation of this metric with respect to Ω_t by noting that incremental changes in the object shape can be expressed in terms of the surface that encloses it:

$$\int_{\Omega_t + \mathrm{d}\Omega_t} \gamma_B(\mathbf{x}) \mathrm{d}\mathbf{x} = \int_{\Omega_t} \gamma_B(\mathbf{x}) \mathrm{d}\mathbf{x} + \int_{\mathcal{S}_t} \gamma_B(\mathbf{x}) \epsilon(\mathbf{x}) \cdot \mathbf{N}(\mathbf{x}) \mathrm{d}\mathbf{x}, \tag{4}$$

where $\mathbf{N} : \mathcal{S}_t \mapsto S^3$ is the surface normal, which is a mapping from every point on the surface to the unit sphere. Differentiating with respect to ϵ gives the first variation [13] with respect to surface position:

$$\mathrm{d}\mathcal{M} = \gamma_B(\mathbf{x}) \mathbf{N}(\mathbf{x}). \tag{5}$$

Using the first variation from Equation 5 and a hill climbing strategy, we obtain the surface motion, defined for each surface point, that minimizes our similarity metric:

$$\frac{\mathrm{d}\mathbf{s}}{\mathrm{d}t} = \gamma_B\left(\mathbf{s}(t)\right) \mathbf{N}\left(\mathbf{s}(t)\right) \, \forall\, \mathbf{s}(t) \, \in \, \mathcal{S}_t. \tag{6}$$

This equation states that each point on the surface $\mathcal{S}_t$ moves at each time step $\mathrm{d}t$ in the direction of the surface normal $\mathbf{N}(\mathbf{s}(t))$ with a step size proportional to the value of the value of the inside-outside function $\gamma_B(\mathbf{s}(t))$ at the point location.

4 Properties

This section describes the properties of the solutions of the partial differential equation given in (6). For this paper, we examine the case of simple, closed, surfaces, but the results easily generalize to more complicated surfaces. In this section we systematically show that if the initial and target objects overlap, the final solution of the metamorphosis will be identical to the target.

4.1 Steady-State Solution

We begin by examining properties of the steady-state solution of Equation 6, which is the surface $\mathcal{S}_t$ such that $\partial\mathbf{s}(t)/\partial t = 0$ for all $\mathbf{s} \in \mathcal{S}_t$. Thus, for steady state, which we denote $\mathcal{S}_\infty$,

$$\gamma_B(\mathbf{x})\mathbf{N}(\mathbf{s}) = 0 \Rightarrow \gamma_B(\mathbf{s}) = 0 \ \ \forall \mathbf{s} \in \mathcal{S}_\infty. \tag{7}$$

The inside-outside function for the target, γ_B, has the property that

$$\gamma_B(\mathbf{x}) = 0 \ \ \Leftrightarrow \ \ \mathbf{x} \ \in \ \mathcal{S}_B. \tag{8}$$

Thus, we can conclude $\mathcal{S}_\infty \subseteq \mathcal{S}_B$. From this it follows that if $\mathcal{S}_\infty$ is compact and closed, it must have one of two forms:

1. $\mathcal{S}_\infty = \emptyset$,
2. $\mathcal{S}_\infty = \mathcal{S}_B$.

The first case is trivial, because there are no points on $\mathcal{S}_\infty$ to violate the condition given in (7). The second case follows from the fact that simple, closed, connected surfaces have no proper subsets that are also closed surfaces. Thus

$$\mathcal{S}_\infty \subseteq \mathcal{S}_B \text{ and } \mathcal{S}_\infty \text{ closed } \quad \Rightarrow \ \ \mathcal{S}_\infty = \mathcal{S}_\mathrm{B}. \tag{9}$$

Also, because the first variation of the metric $\mathcal{M}_{\Omega_B,\Omega_t}$ is zero only where $\mathcal{S}_t = \mathcal{S}_B$, it achieves its maximum for that solution with $\Omega_t = \Omega_B$.

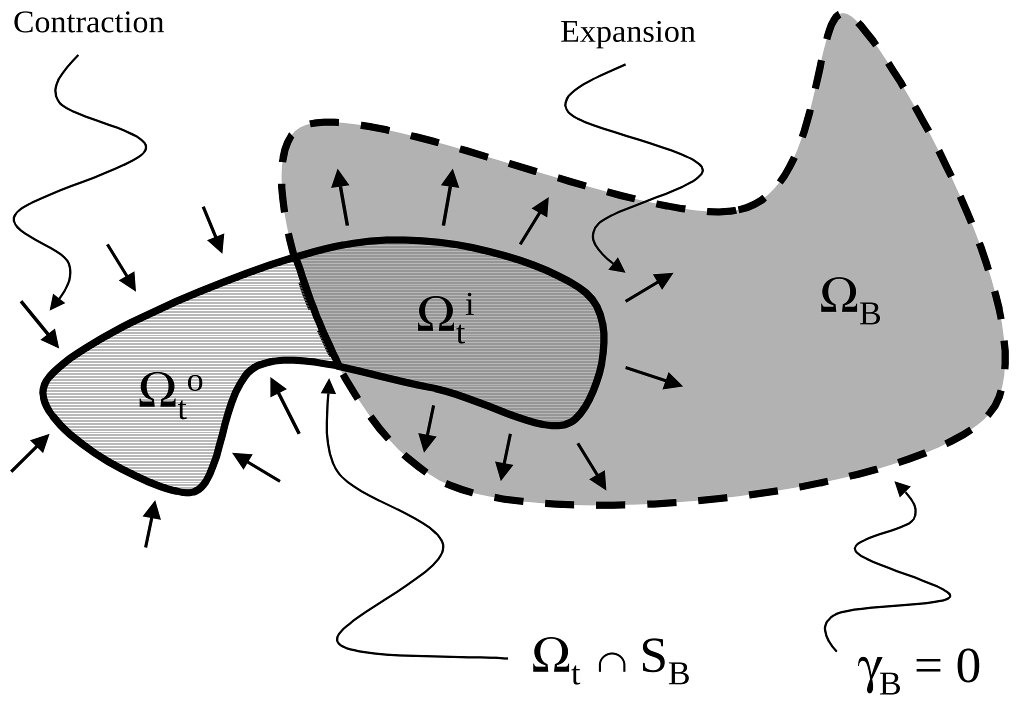

Figure 2: In order to see that overlapping models converge we can divide the model Ω_t into two parts separated by a stationary boundary; one part contracts to annihilation while the other expands until it is equal to the target.

Another important aspect of the algorithm is that a component of the source that overlaps with a component of the target will deform in such a way that it takes on the shape of that target component.[1] This follows from the fact the metric $\mathcal{M}_{\Omega_B,\Omega_t}$ can be decomposed into two parts (as in Figure 2):

$$\mathcal{M}_{\Omega_B,\Omega_t} = \mathcal{M}^i_{\Omega_B,\Omega_t} + \mathcal{M}^o_{\Omega_B,\Omega_t} = \int_{\Omega_t^i} \gamma_B(\mathbf{x}) + \int_{\Omega_t^o} \gamma_B(\mathbf{x}), \tag{10}$$

where $\Omega_t^i = \Omega_t \cap \Omega_B$ and $\Omega_t^o = \Omega_t - \Omega_t^i - \mathcal{S}_B$. In this case the first variation produces a pair of deformable surfaces that share a boundary along $\Omega_t \cap \mathcal{S}_B$. The exterior model remains on the exterior of $\mathcal{S}_B$ and contracts (increasing the value of $\mathcal{M}^o$) until it collapses into itself along $\mathcal{S}_B$ and the value of $\mathcal{M}^o$ reaches zero. The interior model expands, cannot pass into the exterior of $\mathcal{S}_B$, and increases the value of $\mathcal{M}^i$ at a rate

$$\frac{\partial \mathcal{M}^i}{\partial t} = \int_{\mathcal{S}_t^i} \gamma_B(\mathbf{x}) \mathrm{d}\mathbf{x}. \tag{11}$$

[1]A target object may consist of several, disjoint components.

Because $\gamma_B(\mathbf{x})$ is nonnegative on the interior of the target, Ω_B, the metric does not stop increasing (and thus the surface keeps moving) until the surface reaches its steady state, i.e., $\gamma_B(\mathbf{x}) = 0 \ \forall \ \mathbf{x} \in \mathcal{S}_t^i$. Thus, if a source component overlaps with a target component, the resulting morph does not terminate until it reaches the shape of the target.

4.2 Strategy

This formulation suggests a strategy for shape metamorphosis: construct a family of objects Ω_t (with a corresponding $\mathcal{S}_t$), with $\Omega_t|_{t=0} = \Omega_A$ that evolve according to a hill-climbing strategy, and maximize $\mathcal{M}$. If properly initialized (i.e., all of the components of Ω_B have some overlap with Ω_A) then the deformation will seek the global maximum, which is the target Ω_B.

The intermediate shapes contained in Ω_t depend, of course, on the choice of γ_B. In order to avoid numerical difficulties and to avoid discontinuities in the solution, γ_B should be continuous.[2] Furthermore γ_B should reward shapes that are similar to the target but offset by some small distance. That is γ_B should carry information about the shape of the surface into 3-D so that shapes tend to "look like" the target as they get nearer. This suggests that a natural choice of γ_B is the *signed distance transform* [2, 5] of the target surface $\mathcal{S}_B$ or some monotonic function thereof. Thus, if we let $D_B : U \mapsto \mathbb{R}$ be the signed distance transform, $\gamma_B(\mathbf{x}) = f\left(D_B(\mathbf{x})\right)$, where $f(0) = 0$ and $f'(a) > 0$. By tuning f one can control the way the metamorphosis behaves when intermediate surfaces are far from the target. If $f(a) = a$, then $\mathcal{S}_t$ contracts or expands with a magnitude that depends on the signed distance to the target.

As it stands this process is a "low-level" approach to shape metamorphosis, which can form continuous deformations for shapes that are somehow "close" in their initial shapes. As described in the introduction, this low-level process is meant to address the blending stage of 3-D metamorphosis—it is meant to be combined with a higher-level process which accounts for "semantic" aspects of shape correspondence. Incorporating user input is important for any shape morphing technique, because in many cases finding the best set of transition shapes depends on context. Only users can apply semantic considerations to the transformation of one object to another. Our assertion, however, is that this underlying coordinate transformation can achieve only some finite similarity between the "warped" source model and the target, and even this may

[2]In general signed distance transforms are C0 continuous, but not C1 continuous.

require a great deal of user input. In the event that a user is not able or willing to define every important correspondence between two objects, some other method must "fill in" the gaps remaining between the source and target surface. We propose the use of deformable level-set models, to achieve a continuous transition (blending) between the shapes that result from the underlying coordinate transformation.

This higher-level information affects the blending process through a 3-D coordinate transformation, T. T maps a point $\mathbf{x}$ in the coordinate system of the source Ω_A into the coordinate system of the target Ω_B. As described in the introduction, this transformation is meant to be quite general; it can accommodate a wide range of *global* deformations. However, if the blending is sufficiently powerful and can adequately deal with significant shape discrepancies (after the coordinate transformation), only a crude alignment of the source and target is necessary. The coordinate transformation enters into the deformation process through the term γ_B, which quantifies the proximity of one surface to another by means of the distance transform. If we express the distance transform for the target in coordinates of the target, then we have $\gamma_B(\mathbf{x}) = f\left(D_B\left(T(\mathbf{x})\right)\right)$. Alternatively one could resample the distance transform of Ω_B in the coordinate system of Ω_A, i.e., $\gamma'_B(\mathbf{x}) = \gamma_B\left(T(\mathbf{x})\right)$.

5 Level-Set Models

In practice, the strategy of shape metamorphosis by surface deformations described in the previous section must be computed using some specific surface representation. Ideally, we would like the surface representation to be as general as the underlying theory. For this we use the method of *level-set models* [37], which is a technique for modeling surfaces as iso-values of a densely sampled scalar function over the domain U. Surface movements are encoded as changes in the greyscale values of the voxels. Using level-set models, we can compute goal-driven deformations on surfaces without any explicit surface representation.

5.1 Theory

The strategy of level-set models represents a set of surface points $\mathcal{S}$ as an iso-surface of ϕ, which we will call the embedding.

$$\mathcal{S} = \{\mathbf{x} | \phi(\mathbf{x}) = k\}, \tag{12}$$

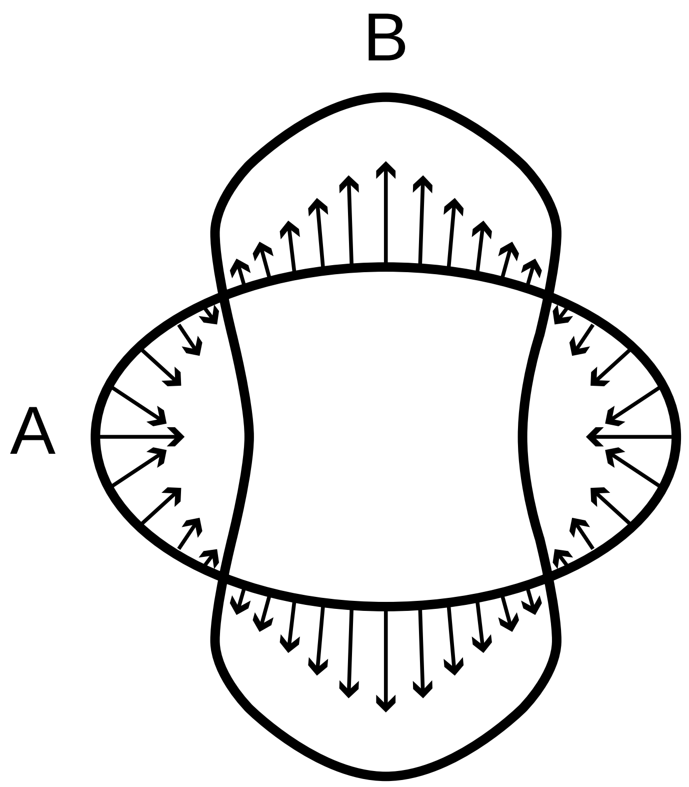

Figure 3: A 2-D morphing example showing scaled surface normals.

where the value of k is arbitrary and will fall out in subsequent calculations.[3] We can also represent a family of surfaces and a corresponding family of embeddings:

$$\mathcal{S}_t = \{\mathbf{x} | \phi(\mathbf{x}, t) = k\}. \tag{13}$$

Consider a point $\mathbf{s}(t)$ on the surface that moves through space as a function of t. Because $\mathbf{s}(t)$ remains the kth level-surface of ϕ over time, the total derivative of ϕ with respect to time must be zero. Thus,

$$\frac{\partial \phi(\mathbf{s}(t), t)}{\partial t} + \nabla \phi(\mathbf{s}(t), t) \cdot \frac{\mathrm{d}\mathbf{s}(t)}{\mathrm{d}t} = 0, \tag{14}$$

which establishes the connection between the way points on $\mathcal{S}_t$ move and the way the greyscale values (at positions on the surface) of the embedding change. We can rewrite Equation 14 as follows:

$$\frac{\partial \phi}{\partial t} = -\nabla \phi \cdot \frac{\mathrm{d}\mathbf{s}(t)}{\mathrm{d}t} = |\nabla \phi| \frac{\mathrm{d}\mathbf{s}(t)}{\mathrm{d}t} \cdot \mathbf{N}(\mathbf{s}), \tag{15}$$

using the fact that $\nabla \phi = |\nabla \phi| \mathbf{N}$.

[3]For the remainder of this paper we use the convention that $k = 0$ is the level set of interest.

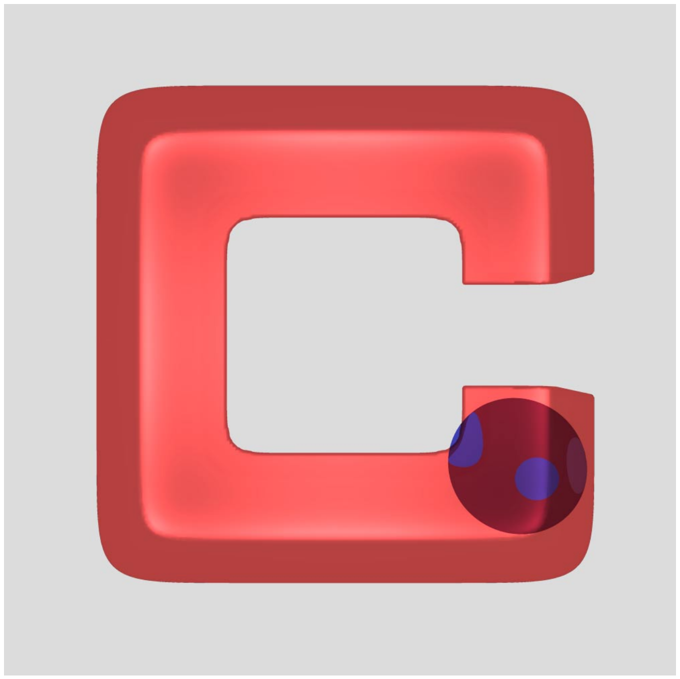

Figure 4: Initial configuration for a morphing sequence.

We can now "plug" into Equation 15 any surface motion we wish to compute. Thus, inserting $\mathrm{d}\mathbf{s}(t)/\mathrm{d}t$ from Equation 6, which describes the motion of the surface model as it becomes more like the target, yields

$$\frac{\partial \phi\left(\mathbf{s}\right)}{\partial t} = |\nabla \phi| \gamma_B \left(\mathbf{s}(t)\right), \tag{16}$$

where we have used the fact that surface normal is unit length, i.e. $N \cdot N = 1$. Notice, we have not chosen a particular k, and therefore this analysis applies to *every* level set of ϕ. Thus we are describing the deformation of an embedded family of surface models each of which evolves according to the same equation:

$$\frac{\partial \phi(\mathbf{x})}{\partial t} = |\nabla \phi(\mathbf{x})| \gamma_B(\mathbf{x}). \tag{17}$$

The particular level set of interest is the one that we choose, by construction of the initial conditions, such that $\phi(\mathbf{x}, 0) = k \ \forall \ \mathbf{x} \in \mathcal{S}_A$.

The deformation process, defined by Equations 6 and 17, is further detailed in Figure 3 with a two-dimensional example. Given the inputs Ω_A and Ω_B, surface $\mathcal{S}_t$ will begin on the surface of $\mathcal{S}_A$. Each point on the surface will move in the direction of the normal at that point with a velocity proportional to the signed distance at that point in 3-space from $\mathcal{S}_B$. Those parts of $\mathcal{S}_t$ that are outside of Ω_B will contract, because D_B is negative in those regions. The parts of $\mathcal{S}_t$ inside of Ω_B will move in the direction of the surface normal, and

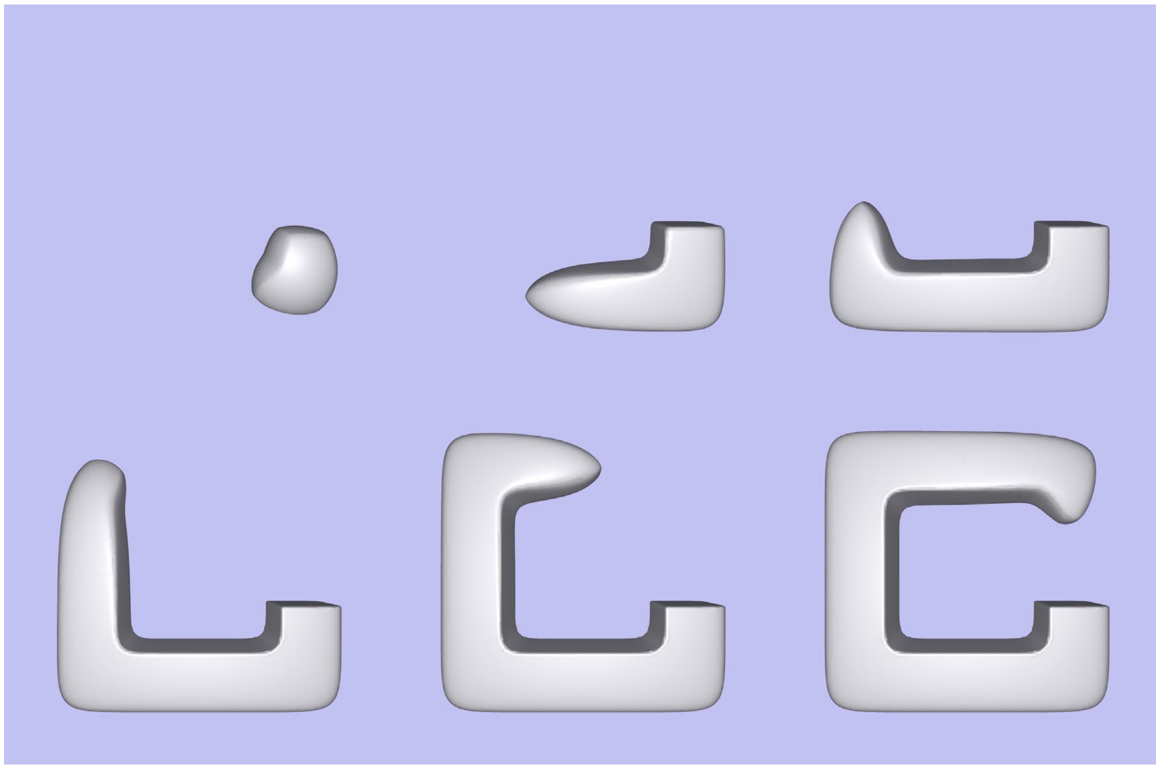

Figure 5: A 3-D level-set morphing example.

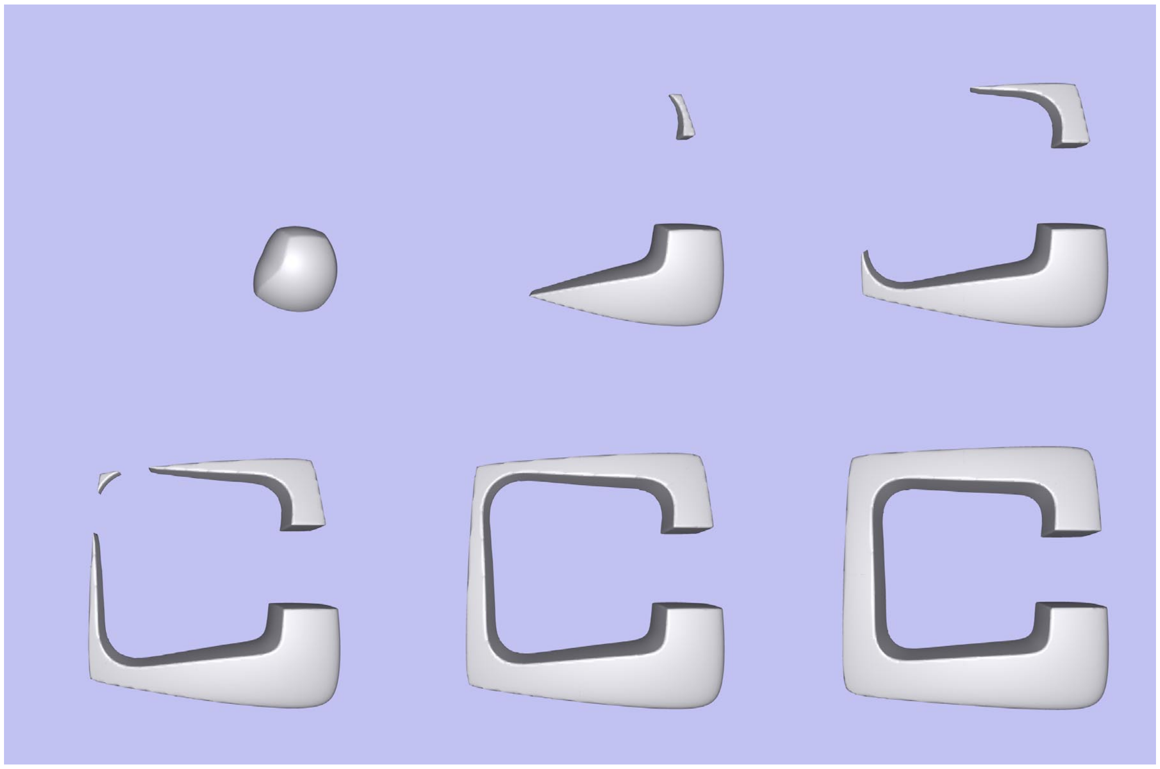

Figure 6: Interpolation of two distance volumes.

will expand. Segments of the surface further away from $\mathcal{S}_B$ will move faster towards $\mathcal{S}_B$ than the segments closer to the surface. As segments of $\mathcal{S}_t$ reach the surface of Ω_B they will no longer move because D_B goes to zero in these regions.

A three-dimensional morphing example is presented in Figures 4 and 5. Figure 4 presents the initial configuration for the morphing sequence. A small sphere will morph into the larger "C" shape. The two objects are first rendered as translucent objects to give some indication of their initial overlap. The ball is mostly contained inside the "C" shape, with the light blue regions of the sphere protruding outside the red "C". The morphing sequence in Figure 5 demonstrates that the regions outside of the "C" contract slightly to fit to the surface of the "C". The surface regions of the sphere inside of the "C" expand to fill the remainder of the object. This also demonstrates that having a surface expand in the direction of its local normal will allow it to deform to fit concave objects, a fact also shown by Miller et al. [29]. Figure 6 contrasts our method of blending with the method used in several other volume-based morphing techniques. Here, the voxel values of the two initial distance volumes are simply interpolated. It can be seen that voxel interpolation produces undesirable artifacts, namely pieces of the "C" shape "pop out of thin air." This is a typical problem when using voxel interpolation to blend volumetric objects. In order to overcome this problem in this example, the sphere would have to be geometrically warped into approximately the same shape as the "C". Our method requires no warping. The user specifies the initial overlap of the two objects and the level-set deformation completes the morph.

The complete deformation strategy is as follows. First, initialize a volume so that the kth level set is, approximately, aligned with $\mathcal{S}_B$. For all of our work we will use the zero-set as the level-set model. This initialization can be done by using the discrete distance transform of $\mathcal{S}_B$, which we compute from CSG models using the method of Breen et al. [7, 8]. We use this initialization to solve the initial value problem given by Equation 17, using the distance transform of the target as the γ_B. We solve this equation using finite forward differences, as described in the next section. When the model is sufficiently close to the target (a threshold on the RMS distance to the target), the process stops and the metamorphosis is complete.

5.2 Implementation

5.2.1 Numerical Issues

The solutions to the partial differential equations described in Section 5.1 are computed using finite differences on a discrete grid[4] The use of a grid and discrete time steps raises a number of numerical and computational issues that are important to the implementation.

The first issue is the discrete approximation of the derivatives in Equation 17. Let u^n be a discrete approximation to $\phi(\mathbf{x}, t)$ at the nth discrete time step. The equation can be solved using finite forward differences if one uses the up-wind scheme, proposed by Osher and Sethian [30], to compute the spatial derivatives. The update equation is

$$u^{n+1}_{i,j,k} = u^n_{i,j,k} + \Delta t \Delta u^n_{i,j,k}, \tag{18}$$

where Δt is a constant that is chosen to ensure stability, and $\Delta u^n_{i,j,k}$ is the discrete approximation to $\partial\phi/\partial t$. We assume, without a loss in generality, that the grid spacing is unity. The initial conditions u^0 are established by the algorithm and the boundary conditions are such that the derivatives toward the outside of the grid are zero (Neumann type).

The up-wind scheme relies on *one-side* derivatives:

$$\begin{aligned}
\delta^+_x u^n_{i,j,k} &= u^n_{i+1,j,k} - u^n_{i,j,k}, &(19)\\
\delta^-_x u^n_{i,j,k} &= u^n_{i,j,k} - u^n_{i-1,j,k}, &(20)\\
\delta^+_y u^n_{i,j,k} &= u^n_{i,j+1,k} - u^n_{i,j,k}, &(21)\\
\delta^-_y u^n_{i,j,k} &= u^n_{i,j,k} - u^n_{i,j-1,k}, &(22)
\end{aligned}$$

and so forth. The partials in Equation 17 are computed using only those derivatives that are up-wind relative to the movement of the level set. Thus, the update becomes

$$\begin{aligned}
\Delta u_{i,j,k} \quad = \quad & \gamma_B\,(i,j,k) \cdot \\
& \begin{cases} \left(\sum_{w\in x,y,z} \min(\delta^+_w u^n_{i,j,k}, 0)^2 + \sum_{w\in x,y,z} \max(\delta^-_w u^n_{i,j,k}, 0)^2\right)^{\frac{1}{2}} & \text{for } \gamma_{\mathrm{B}}(\mathrm{i,j,k}) \geq 0 \\ \left(\sum_{w\in x,y,z} \max(\delta^+_w u^n_{i,j,k}, 0)^2 + \sum_{w\in x,y,z} \min(\delta^-_w u^n_{i,j,k}, 0)^2\right)^{\frac{1}{2}} & \text{for } \gamma_{\mathrm{B}}(\mathrm{i,j,k}) < 0. \end{cases}
\end{aligned} \tag{23}$$

[4]The level-set software used to produce the morphing results in this paper is available for public use. Contact Ross Whitaker at whitaker@cs.utah.edu for more information.

The time steps, Δt, are limited by the speed of the fastest moving wavefront, which can move only one grid unit per iteration, i.e.,

$$\Delta t \leq \frac{1}{3 \max_{i,j,k} |\gamma_B(i,j,k)|}. \tag{24}$$

In practice, for the purposes of computer animation, one might further limit the time steps to obtain sequences with a sufficient number of in-between frames.

Thus, the level-set method for computing the 3D shape metamorphosis is as follows:

1. Initialize model volume u^0 by sampling the inside-outside function of the source.
2. Construct the volume v, by sampling the inside-outside function of the target.
3. $v_{\max} = 0$
4. For each voxel (i,j,k):
   (a) Find $v_{i,j,k}$.
   (b) $v_{\max} = MAX(|v_{i,j,k}|, v_{\max})$
   (c) Calculate derivatives and the total change at (i,j,k) using nearest neighbors according to the up-wind scheme given in Equation 23.
   (d) Save $\Delta u^n_{i,j,k}$ in a separate volume.
5. Compute Δt according to Equation 24.
6. For each voxel (i,j,k):
   (a) Update $u^{n+1}_{i,j,k}$ according to Equation 18
   (b) Compute the stopping criterion, either RMS change or RMS distance to target.
   (c) If stopping criterion is met, finish, otherwise go to step 4

5.2.2 Sparse-Field Solutions

The up-wind solutions to the equations described in the previous section produces the motion of level-set models over the entire range of the embedding, i.e., for all values of k in Equation 13. However, this method

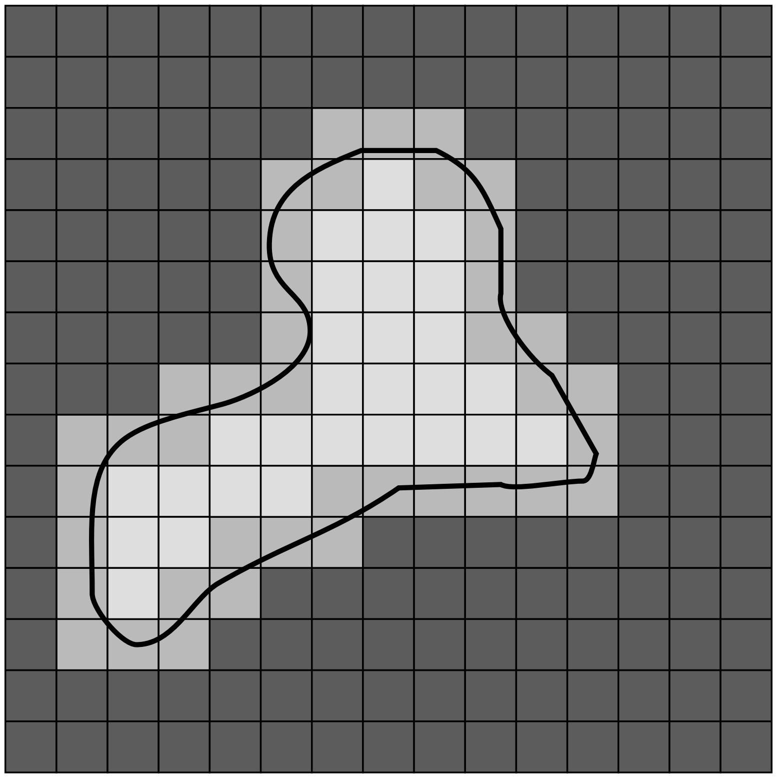

Figure 7: A level curve of a 2-D scalar field passes through a finite set of grid points. Only those grid points and their nearest neighbors are relevant to the evolution of that curve.

requires updating *every voxel in the volume for each iteration.*, which means that the computation time increases as a function of the volume, rather than the surface area, of the model. Because the application of this paper, surface metamorphosis, requires only a single model, the calculation of solutions over the entire range of iso-values is an unnecessary computational burden.

The literature has shown this situation can be improved by the use of *narrow-band* methods, which compute solutions only in a narrow band of voxels that surround the level set of interest [1, 28]. In previous work [40, 42] we described an alternative numerical algorithm, called the sparse-field method, that computes the geometry of only a small subset of points in the range and requires a fraction of the computation time required by previous algorithms. We have shown two advantages to this method. The first is a significant improvement in computation times. The second is increased accuracy when fitting models to forcing functions that are defined to sub-voxel accuracy.

The sparse-field algorithm takes advantage of the fact that a k-level surface, S, of a discrete image u (of any dimension) has a set of cells through which it passes, as shown in Figure 7. The set of grid points adjacent to the level set is called the *active set*, and the individual elements of this set are called *active points*. As a first-order approximation, the distance of the level set from the center of any active point is proportional to the value of u divided by the gradient magnitude at that point. We compute the evolution given by Equation 17 on the active set and then update the neighborhood around the active set using a fast approximation to the distance transform, which simply adds the "city-block" distance to values of the active set. Because active points must be adjacent to the level-set model, their positions lie within a fixed distance to the model. Therefore the values of u for elements in the active set must lie within a certain range of greyscale values. When active-point values move out of this *active range* they are no longer adjacent to the model. They must be removed from the set and other grid points, those whose values are moving into the active range, must be added to take their place. The precise ordering and execution of these operations is important to the operation of the algorithm.

The values of the points in the active set can be updated using the up-wind scheme described in the previous section. In order to maintain stability, one must update the neighborhoods of active grid points in a way that allows grid points to enter and leave the active set without those changes in status affecting their values. Grid points should be removed from the active set when they are no longer the nearest grid point to the zero crossing. If we assume that the embedding u is a discrete approximation to the distance transform of the model, then the distance of a particular grid point, (i, j, k), to the level set is given by the value of u at that grid point. If the distance between grid points is defined to be unity, then we should remove a point from the active set when the value of u at that point no longer lies in the interval $[-\frac{1}{2}, \frac{1}{2}]$. If the neighbors of that point maintain their distance of 1, then those neighbors will move into the active range just as (i, j, k) is ready to be removed.

There are two operations that are significant to the evolution of the active set. First, the values of u at active points change from one iteration to the next. Second, as the values of active points pass out of the active range they are removed from the active set and other neighboring grid points are added to the active set to take their place. Formal definitions of active sets and the operations that affect them are detailed in [42], and it is shown that active sets will always form a boundary between positive and negative regions in the image, even as control of the level set passes from one set of active points to another.

Because grid points that are near the active set are kept at a fixed value difference from the active points,

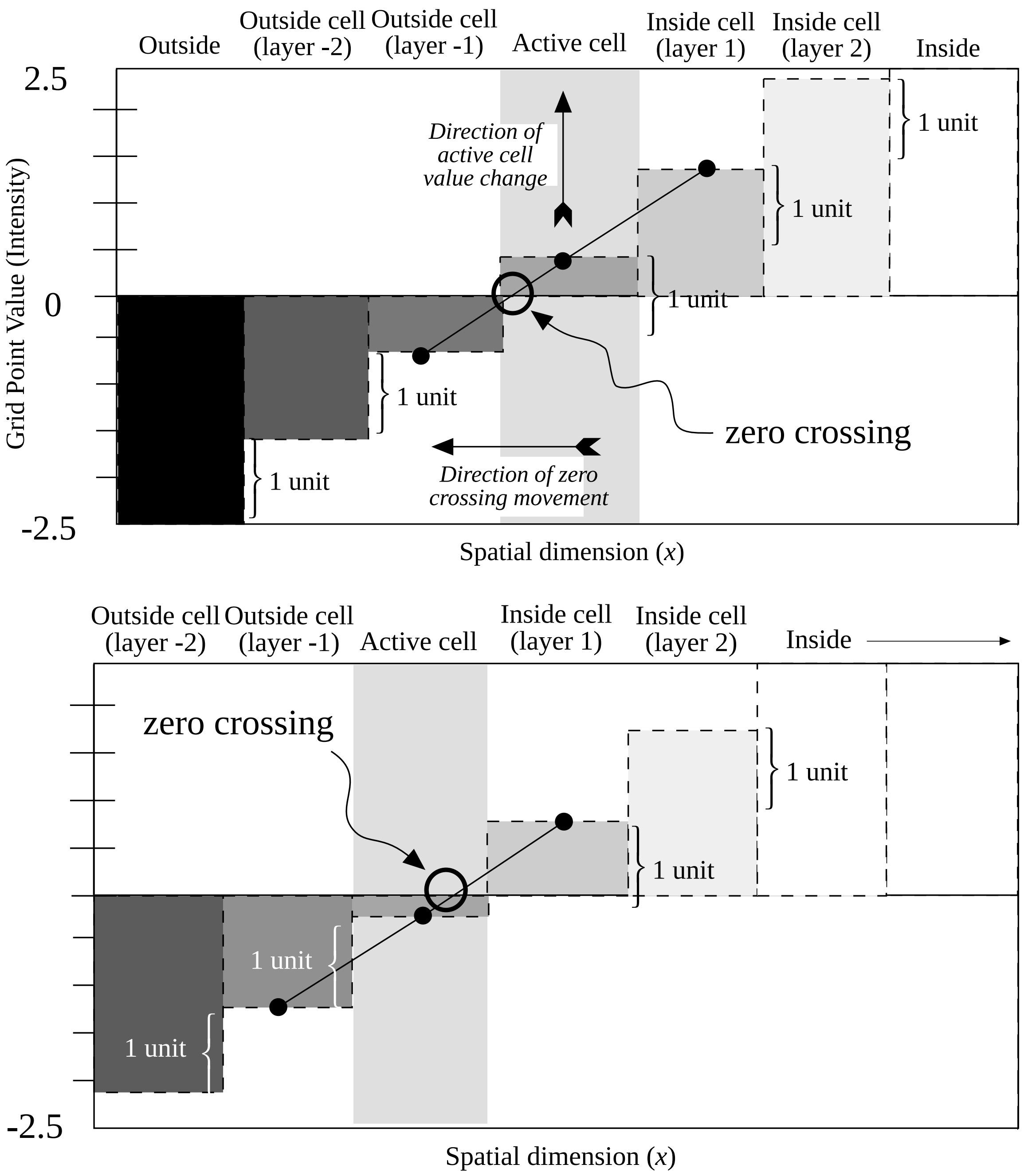

Figure 8: The status of grid points and their values at two different points in time show that as the zero crossing moves, *activity* is passed one grid point to another.

active points serve to control the behavior of adjacent nonactive grid points. The neighborhoods of the active set are defined in *layers*, $L_{+1}, \ldots, L_{\ell}, \ldots, L_{+N}$ and $L_{-1}, \ldots, L_{-\ell}, \ldots, L_{-N}$, where the ℓ indicates the distance (city block distance) from the nearest active grid point, and negative numbers are used for the outside layers. For notational convenience the active set is denoted L_0. The number of layers should coincide with the size of the footprint or neighborhood used to calculate derivatives. In this way, the inside and outside grid points undergo no changes in their values that affect or distort the evolution of the zero set. The work in this paper uses only first-order derivatives of ϕ, which are calculated using nearest neighbors (6 connected). Therefore only 3 layers are necessary (1 inside layer, 1 outside layer, and the active set). These layers are denoted L_1, L_{-1}, and L_0. The active set has grid point values in the range $[-\frac{1}{2}, \frac{1}{2}]$. The values of the grid points in each neighborhood layer are kept 1 unit from the next layer closest to the active set as shown in Figure 8. Thus the values of layer L_ℓ fall in the interval $[\ell - \frac{1}{2}, \ell + \frac{1}{2}]$. For $2N + 1$ layers, the values of the grid points that are totally inside and outside are $N + \frac{1}{2}$ and $-N - \frac{1}{2}$, respectively.

This algorithm can be implemented efficiently using linked-list data structures combined with arrays to store the values of the grid points and their states as shown in Figure 9. This requires only those grid points whose values are changing, the active points and their neighbors, to be visited at each time step. The computation time grows as m^2, where m is the number of grid points along one dimension of U (sometimes called the resolution of the discrete sampling). The m^2 growth in computation time for the sparse-field models is consistent with conventional (parameterized) models, for which computation times increase with surface area rather than volume.

Another advantage of the sparse-field approach is resolution. Equation 17 describes a process whereby all of the level sets of ϕ are pushed toward the zero-set of γ_B. The result is a *shock*, a discontinuity in ϕ. In discrete volumes these shocks take the form of high-contrast areas, which cause aliasing in the resulting models. This results in surface models that are unacceptable for many computer graphics applications, and which do not resemble the target in the final stages of the morph (violating criteria 3 in Section 1).

When using the sparse-field method, the active points serve as a set of control points on the level set. Changing the values of these voxels changes the position of the level set. The forcing function is sampled not at the grid point, but at the location of the nearest level set, which generally lies between grid points. Using a first-order approximation to ϕ produces results that avoid the aliasing problem associated with the shocks

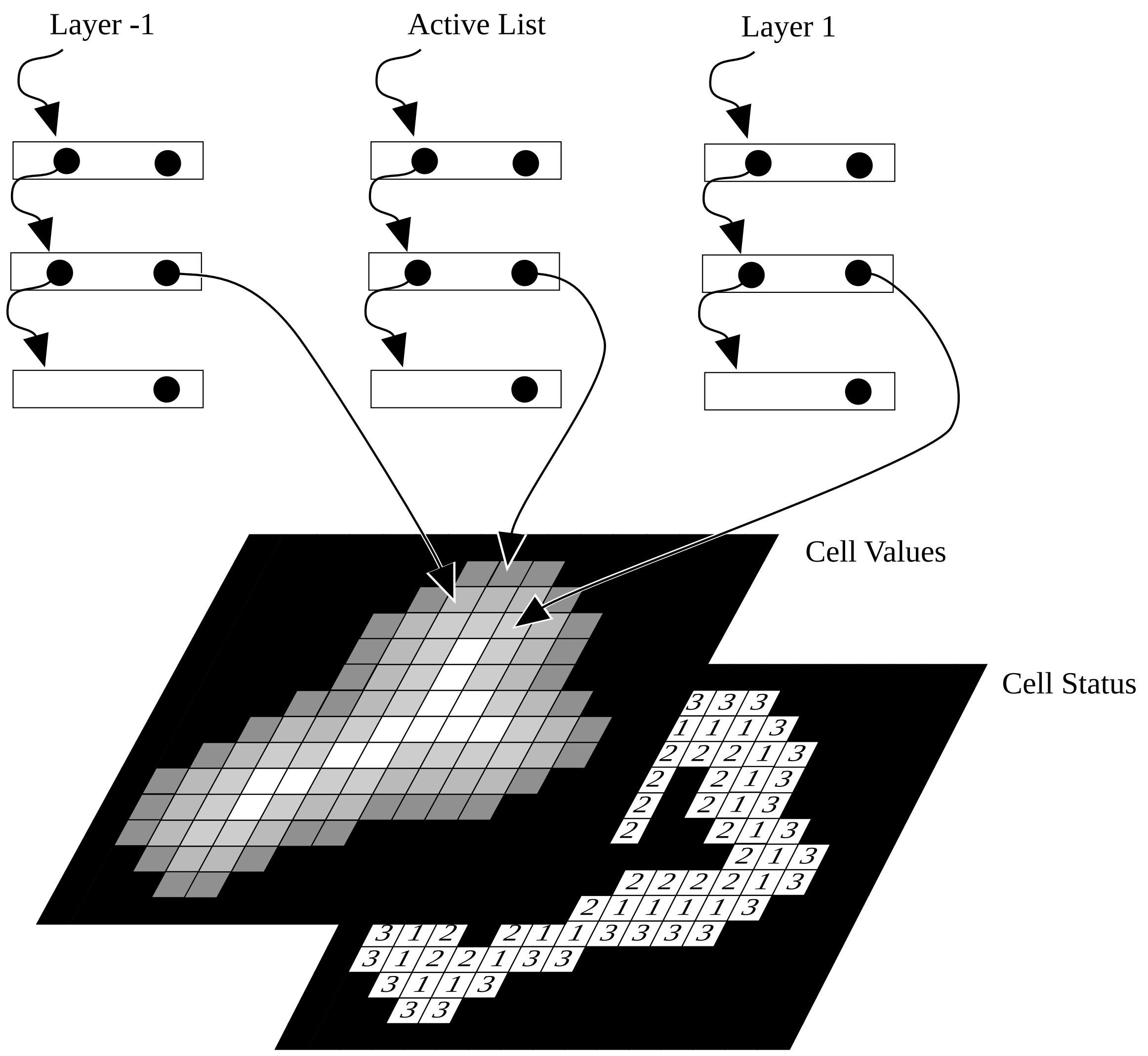

Figure 9: Linked-list data structures provide efficient access to those grid points with values and status that must be updated.

that typically occur with level-set models. Previous work has shown significant increases in the accuracy of fitting level-set models using the first-order modification to the sparse-field method [42], which is essential to the shape metamorphosis application in this paper.

With the first-order modification, the procedure for updating the image and the active set based on surface movements is as follows:

1. For each active grid point (i, j, k):
   (a) Use first-order derivatives and Newton's method to calculate the position (i', j', k') of the nearest zero-crossing to (i, j, k).
   (b) Calculate the $\gamma_B(i', j', k')$ using trilinear interpolation, if necessary.
   (c) Compute the net change of $u^n_{i,j,k}$, based on $\gamma_B(i', j', k')$ and the values of its derivatives using the up-wind scheme (Equation 23).
2. For each active grid point (i, j, k) add the change to the grid point value and determine if the new value $u^{n+1}_{i,j,k}$ falls outside the $[-\frac{1}{2}, \frac{1}{2}]$ interval. If so, put (i, j, k) on lists of grid points that are changing status, called the ***status list***; S_1 or S_{-1}, for $u^{n+1}_{i,j,k} > \frac{1}{2}$ or $u^{n+1}_{i,j,k} < -\frac{1}{2}$, respectively.
3. Visit the grid points in the $2N$ layers L_ℓ in the order $\ell = \pm 1, \ldots \pm N$, and update the grid point values based on the values (by adding or subtracting one unit) of the next inner layer, $L_{\ell \mp 1}$. If more than one $L_{\ell \mp 1}$ neighbor exists then use the neighbor that indicates a level set closest to that grid point, i.e., use the maximum for the outside layers and minimum for the inside layers. If a grid point in layer L_ℓ has no $L_{\ell \mp 1}$ neighbors, then it is demoted to $L_{\ell \pm 1}$, the next level away from the active set.
4. For each status list $S_{\pm 1}, S_{\pm 2}, \ldots, S_{\pm N}$:
   (a) For each element (i, j, k) on the status list S_ℓ, remove (i, j, k) from the list $L_{\ell \mp 1}$, and add it to the L_ℓ list, or, in the case of $\ell = \pm(N + 1)$, remove it from all lists.
   (b) Add all $L_{\ell \mp 1}$ neighbors to the $S_{\ell \pm 1}$ list.

More details on sparse-field method and its properties can be found in [40, 42].

6 Summary of the Level-Set Metamorphosis Approach

This section describes the complete approach, based on level-set models, to 3-D shape metamorphosis which meets the criteria listed in the introduction. The specific steps of our level-set morphing approach are 1) 3-D scan conversion of source and target objects, 2) application of coordinate transformations, 3) level-set deformation, and 4) polygonization and rendering.

3-D Scan Conversion. The essential input to the deformation stage of our morphing approach is two 3-D models represented as distance volumes. A distance volume (or distance transform) is a volume dataset where the value stored at each voxel is the shortest distance to the surface of the object being represented by the volume. In our examples, distance volumes are generated by scan converting CSG models, but any technique that converts a solid model into a distance volume may be used. Several methods have been developed for converting polygonal, swept and volumetric models into distance volumes[11, 16, 20, 32, 35, 37].

We have developed a 3-D scan conversion technique that produces a distance volume from a CSG model consisting of superellipsoids [3] and calculates distance to subvoxel accuracy [7, 8]. The distance volume is generated in a two step process. The first step calculates the shortest distance to the CSG model at a set of points within a narrow band around the evaluated surface. Additionally, a second set of points, labeled the zero set, which lies on the CSG model's surface is computed. A point in the zero set is associated with each point in the narrow band. Once the narrow band and zero set are calculated a fast marching method [36, 38] is employed to propagate the shortest distance and closest point information out to the remaining voxels in the volume.

Controlling the Morph with Coordinate Transformations. In order for our active level-set model to deform from one surface into another the source and target objects must overlap. The objects may be automatically or interactively positioned, as well as, interactively warped in order to produce a particular model alignment. The user may choose any of these methods depending upon the level of control and final output desired. The source object will shrink in those areas where it is outside the target object, and will expand in those areas inside the target model. Thus, the user controls the morph by defining the regions of overlap between the source and the target. This is accomplished by applying a coordinate transformation which maps the voxel locations of the source object into new locations in the target object's distance volume. The transformation is given by

$$\mathbf{x}' = T(\mathbf{x}, \alpha), \tag{25}$$

where $0 \leq \alpha \leq 1$ parameterizes a continuous family of transformations that begins with identity, i.e. $\mathbf{x} = T(\mathbf{x}, 0)$, and smoothly becomes the user-defined transformation at $T(\mathbf{x}, 1)$. The parameterization is utilized during the polygonization stage and is explained further on in this section.

For this work, we have developed a software tool that allows a user to interactively position, rotate and scale the source and target objects in order to produce the transformation T. The coordinate systems of the two objects are aligned, and the user is able to manipulate the objects until they are properly overlapped. We have also developed a technique for automatically positioning, orienting and scaling objects, using 3-D moments, in order to achieve a significant correspondence between two objects without any user input. This method is detailed in the Appendix.

A generalized warping, as defined in [11, 26], may also be employed to provide even more detailed control of the process. Here, the user specifies the numerous geometric features which correspond in the source and target objects. The morph may be predominantly controlled by the geometric warp which has been interactively defined by the user and which has a number of degrees of freedom that are proportional to the number of fiducials. In this case the level-set model would simply fine-tune the surface model as the source object is incrementally transformed by the warping algorithm into the target object. Because the goal of our work is to demonstrate a more powerful blending mechanism that performs well without extensive user input, we do not utilize such generalized warpings for the morphing results presented in this paper.

Level-Set Deformation. Once the overlap of the source and target objects has been defined and any generalized warping has been applied to the source, the level-set deformation process, as described in Sections 3, 4 and 5, is initiated. The process produces a sequence of volume datasets that represent the morphing object. The user defines how often the level-set volume is written to disk during the deformation process.

Polygonization and Rendering. In order to view the morph, we extract a polygonal iso-surface (with the Marching Cubes algorithm [27]) from each volume produced by the level-set deformation process. The polygons are rendered to produce a series of images, which are then combined to produce an animation. Once the level-set models have been converted into polygons, any number of conventional rendering and animation techniques may be used to shade and view the morphing object. We have developed a color shading method, based on scan-converted closest-point and color information, in order to define the colors on the resulting unparameterized polygonal models. Using this method, the color at any point in space is defined as the color at the closest point on the associated CSG model. We interpolate the color values computed from the source and target models to produce the surface colors for the intermediate shapes. The color shading method is beyond the scope of this paper and is described in [6, 8].

If a shape-changing transformation $T(\mathbf{x}, \alpha)$ (e.g., a scaling, or a generalized warp) has been utilized during the deformation process, the transformation must be interpolated and incrementally applied to the resulting polygonal models generated at each time step. Applying such a transformation implies that the morph is a combination of the user-defined transformation and the level-set deformation. The total time of the morph is scaled down to a range of $[0, 1]$, which matches the parameterization of $T(\mathbf{x}, \alpha)$. For example, a particular morph may produce N time steps, and therefore N individual volume datasets. Before rendering the polygonal model produced from step n, the polygons should be transformed by $T(\mathbf{x}, n/(N-1))$ before being rendered. The number $(N - 1)$ results from starting the count at zero. At the final frame, the transformation is $T(\mathbf{x}, 1) = T$, the same transformation that is used to generate γ_B. Thus, the level-set deformation ensures that the source evolves into the target, to within a coordinate transformation, T, which is accounted for by transforming all of the points in the polygonal model.

7 Results

Figure 12 presents a morphing sequence of a dart becoming an X-29 jet. The X-29 and dart models were constructed and scan converted into distance volumes of resolution $96 \times 192 \times 240$ with The Clockworks [15], a CSG modeling system. Lerios et al. [26] demonstrate a similar transition with a jet and a dart, which required the specification of 37 different user-defined correspondence elements on both models, roughly 200 user-defined parameters. Our morph required only a few minutes of user time to interactively overlay the source and target models. The jet and dart have been rendered semi-transparently in their initial configurations and presented in Figure 10 in order to demonstrate how they were overlaid before initiating

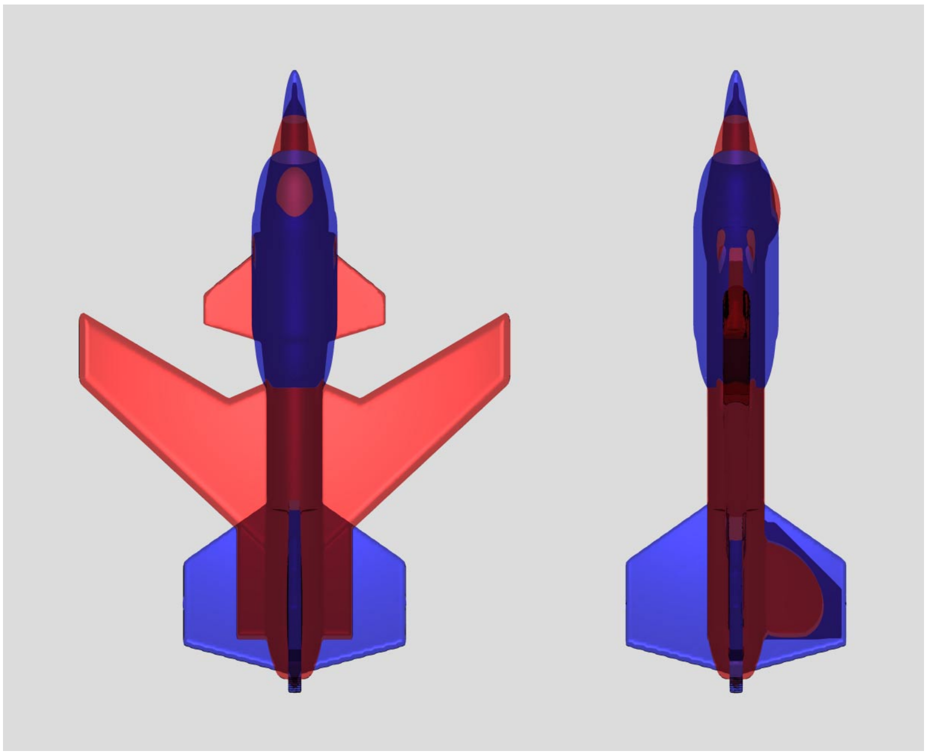
Figure 10: Initial model configuration for morph in Figures 12 and 13.

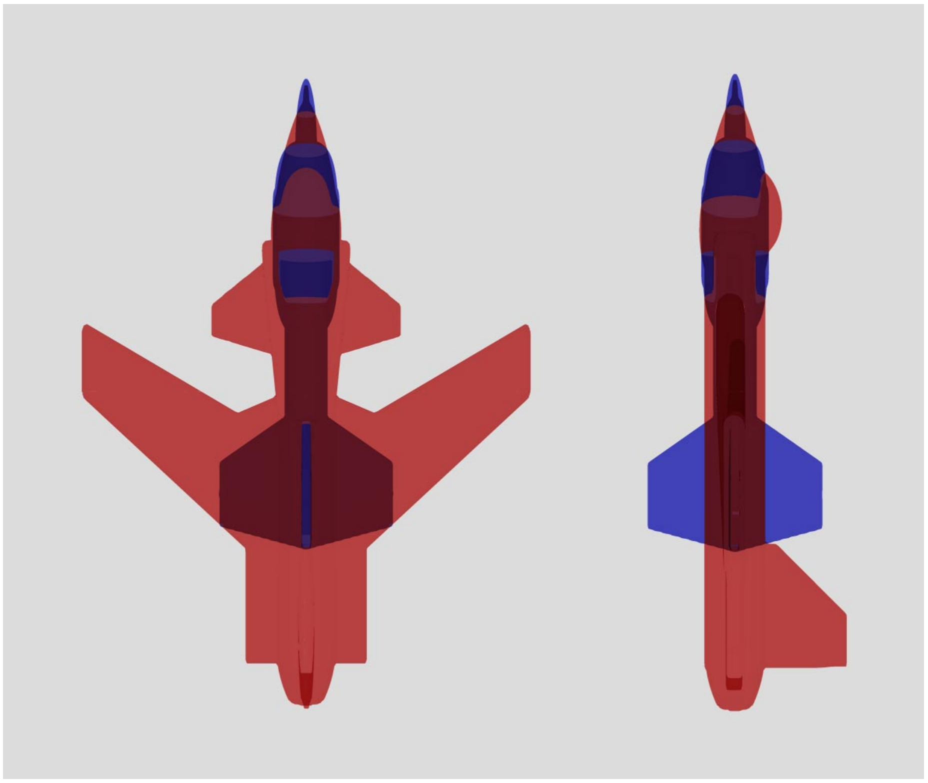
Figure 11: Initial model configuration for morph in Figure 14.

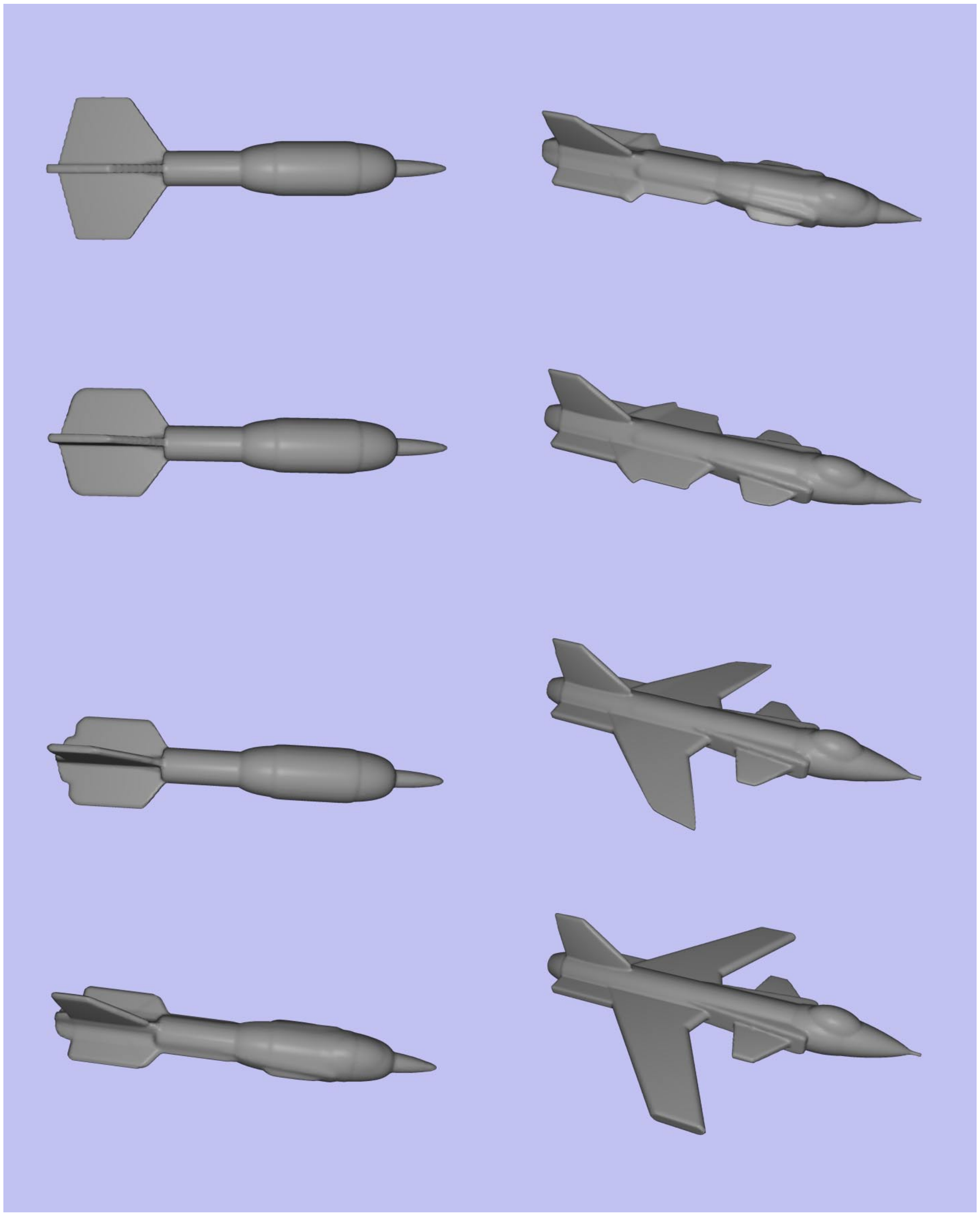

Figure 12: A dart morphing into an X-29 jet.

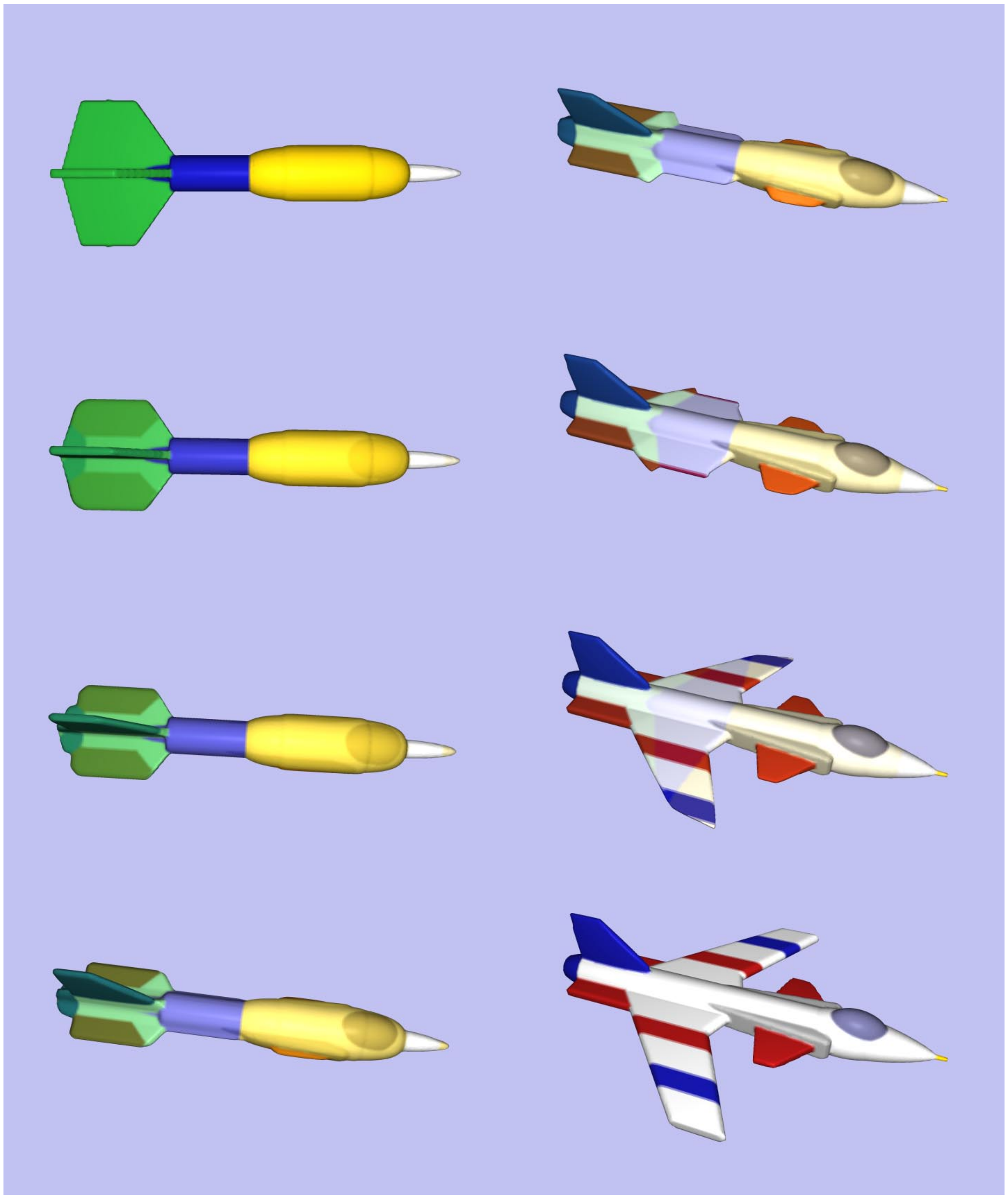

Figure 13: A dart morphing into an X-29 jet with interpolating surface colors.

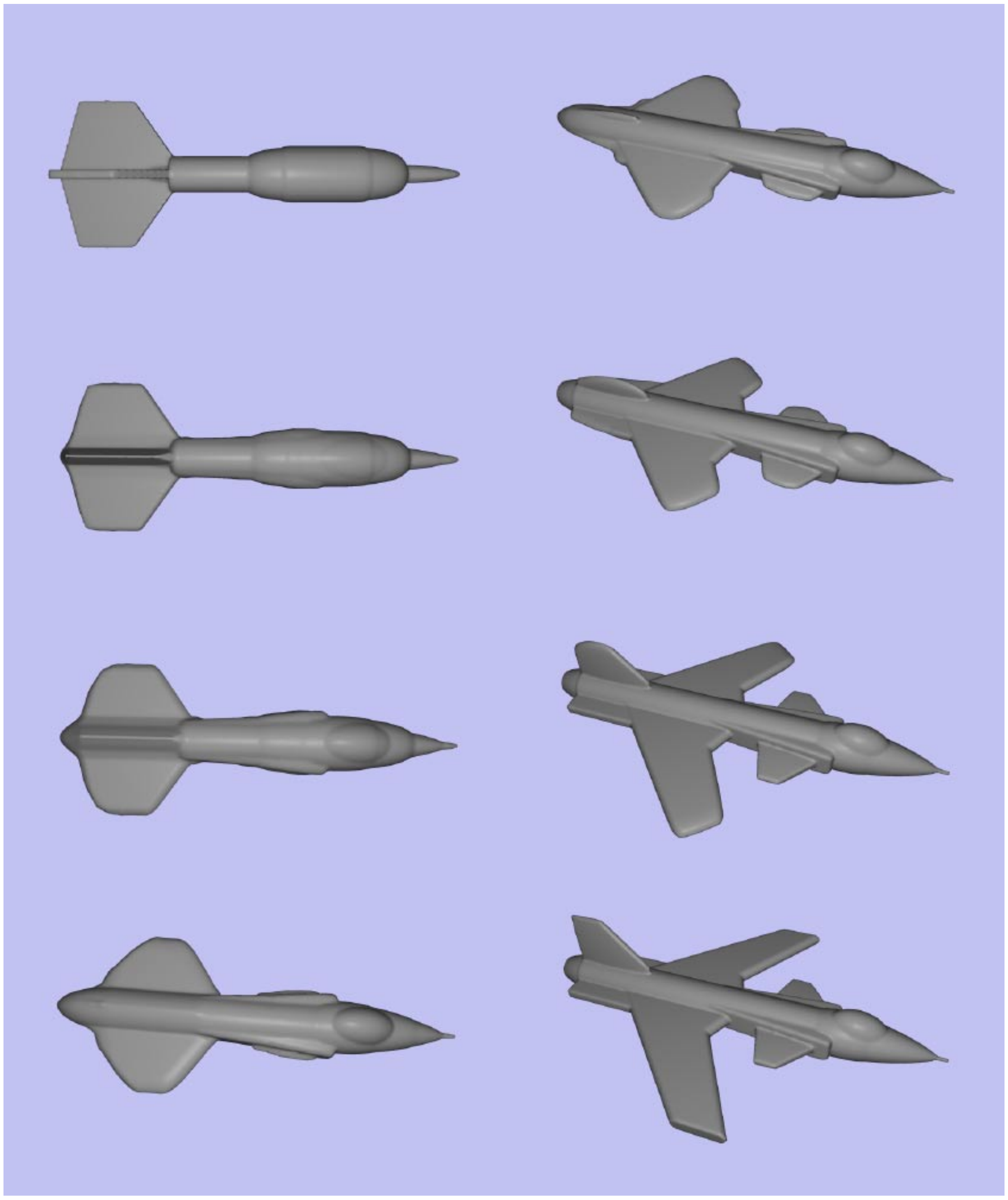

Figure 14: A dart morphing into an X-29 jet using different initial conditions.

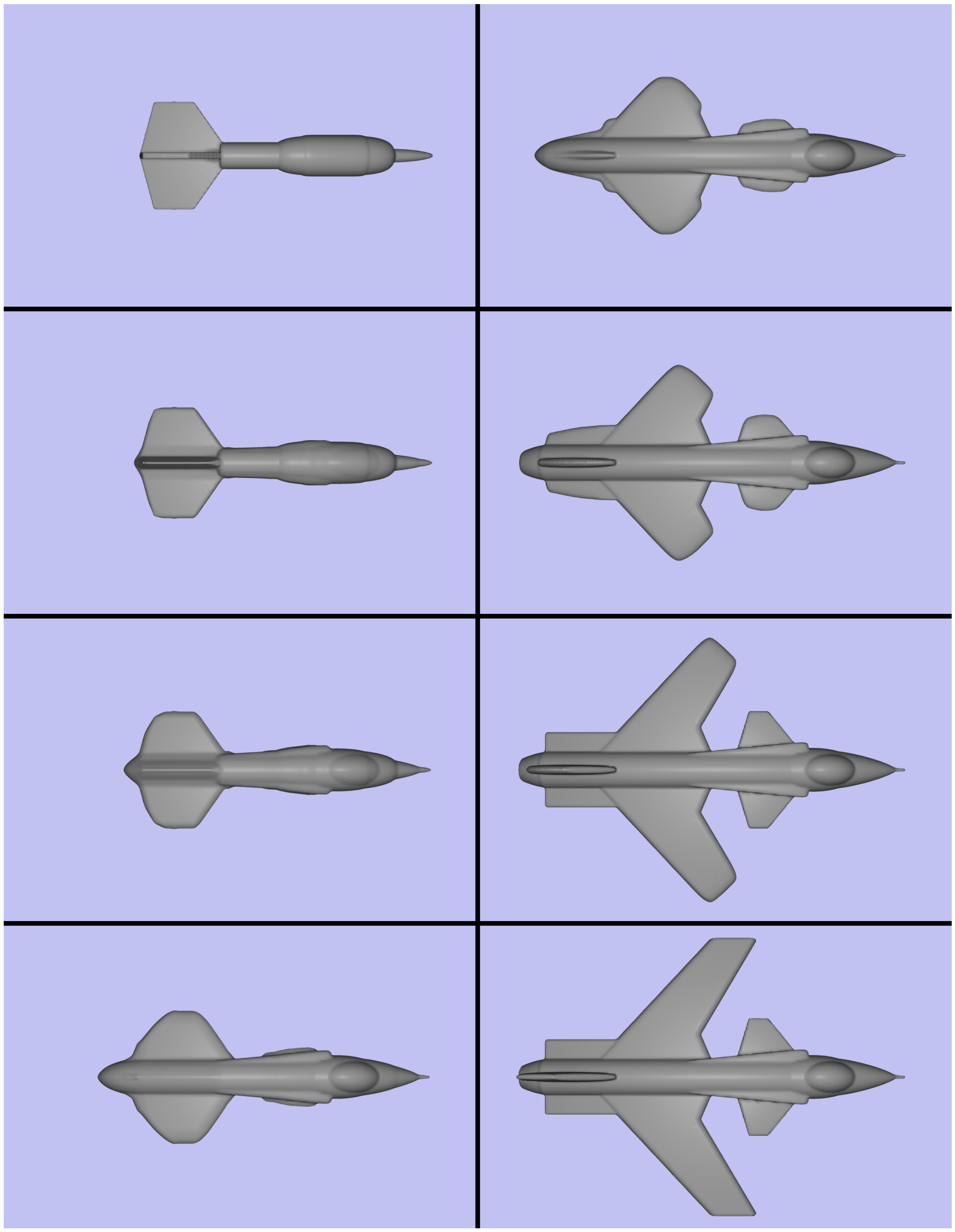

Figure 15: The morph from Figure 14 without applying the incremental transformation.

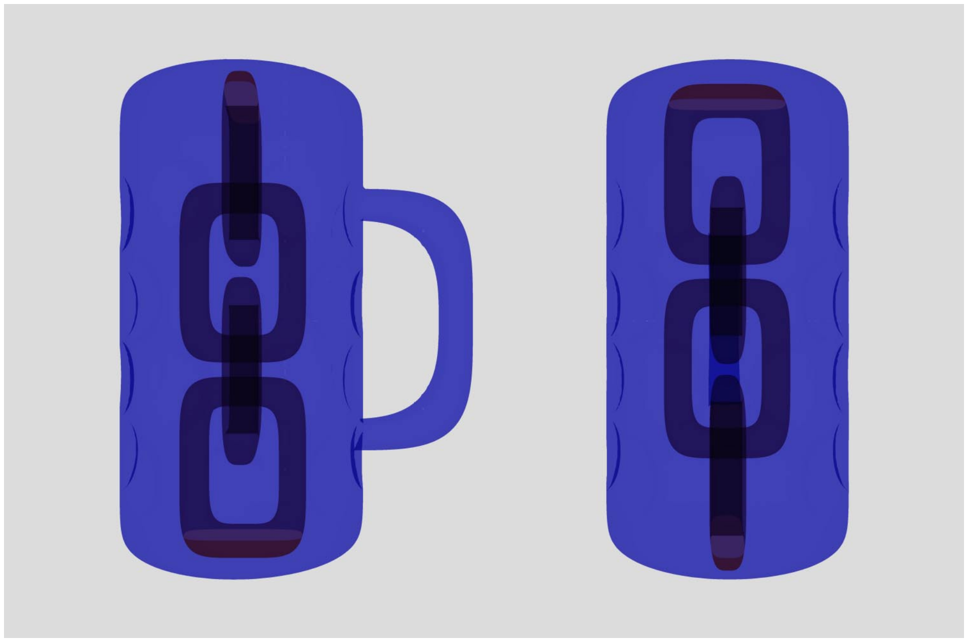

Figure 16: Initial model configuration for morph in Figure 17.

the deformation process. The jet was rendered in light transparent red, and the dart was rendered in light transparent blue. The areas of dark red or dark blue indicate regions where the models overlap. Figure 13 presents the morphing sequence of Figure 12 with interpolating surface colors generated with our color-shading algorithm [6, 8].

Figure 14 uses the same input models with slightly different initial conditions to produce a different morph of the dart turning into the X-29. In this morph the animator wanted the back fins of the dart to morph into the wings of the jet. This was achieved with only a few minutes of user input, the time needed to specify a scale and translation applied to the dart. The level-set deformation stage for this (and the previous) morphing sequence required approximately 9 CPU-minutes on an SGI R10000 Onyx2. Figure 11 presents the initial conditions of the jet and dart model for the second morph. The dart has been scaled by .75 and translated by $[11.9, 24.0, -7.2]$ so that its fins will overlap with the wings of the jet. Before rendering each frame of the morphing sequence, each vertex ($\mathbf{P}$) of the polygonal model produced by the Marching Cubes algorithm is transformed by

$$\mathbf{P}' = (1.0 - ((n/(N-1))(1.0 - 0.75)))\mathbf{P} + (n/(N-1))[11.9, 24.0, -7.2], \tag{26}$$

where n is the frame number and N is the total number of frames. An additional global rotation has been applied to the models in Figures 12, 13, 14 and 17 to highlight the three-dimensional structure of the

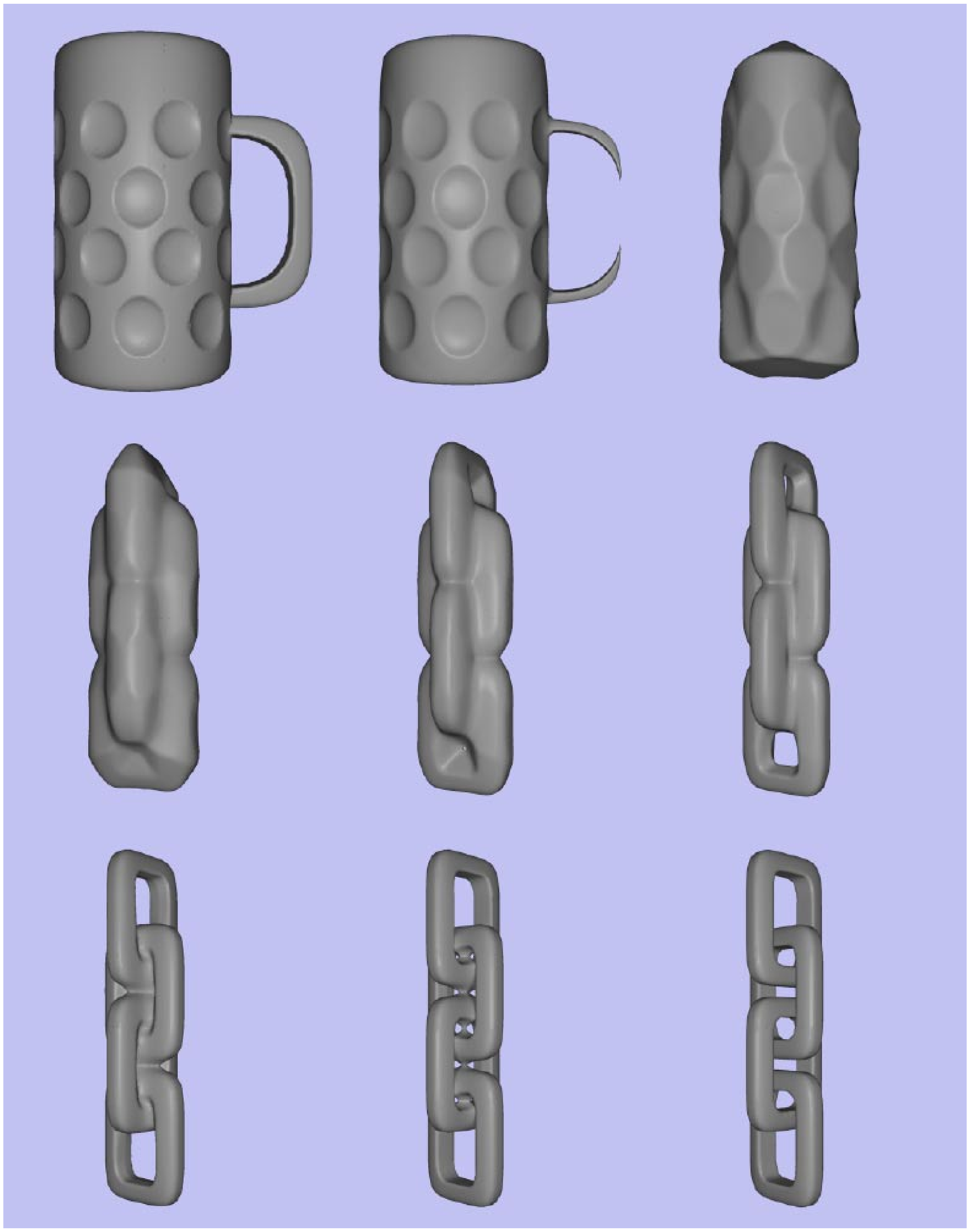

Figure 17: A mug morphing into a chain. This demonstrates that level set models easily cope with changes in topology.

morphing model. Applying the transformation in Equation 26 ensures that the size and location of the morphing model remains approximately constant while it is changing shape. Figure 15 presents the morph from Figure 14 without applying the transformation. It can be seen that without the transformation the morphing model changes shape *and* size, with the tail end of the dart growing into the rear section of the X-29. As seen in Figure 10 the dart and X-29 initially are approximately the same size.

Figure 16 presents the initial configuration of a mug-to-chain morph. The mug and chain were originally defined with CSG models and then scan converted into distance volumes of resolution $120 \times 148 \times 184$. Figure 17 presents the morphing result. The level-set deformation stage of this sequence required approximately 20 CPU-minutes on an SGI R10000 Onyx2. This sequence demonstrates that the level-sets approach easily copes with changes of topology during morphing. The results from all three morphing sequences also highlight the advantage of calculating to sub-voxel accuracy. The volume resolutions are somewhat low, but the deforming surfaces extracted from them are mostly free of aliasing artifacts.

8 Conclusions and Future Work

This paper has presented a volume-based method for achieving 3-D shape metamorphosis. The method relies on a novel approach to the blending stage of the morphing process. This stage is formulated as the optimization, via a hill-climbing strategy, of a similarity measure between the deforming surface and the target, utilizing level-set models for the incremental shape changes. These models take advantage of a volume-based representation to calculate surface deformations to sub-voxel accuracy. The movements of these deformable models are driven by the signed distance transform of the target, which is also computed to sub-voxel accuracy. The result is a 3-D morphing technique which demonstrates outstanding fidelity, level of detail, flexibility, and degree of automation, comparing favorably with other methods in the literature.

As with all volumetric morphing methods, our method has its limitations. The basic volumetric representation of the objects can produce aliasing artifacts on objects that have regions of high curvature. Since a distance volume is a sampled representation, the accuracy of individual object features is restricted by the sampling resolution of volume. Additionally, our method is only useful for solid, (i.e. closed) objects. Our method cannot be used to morph open shell-like surfaces.

Even given our current results, more work remains. We will continue to work on our volume-based technique for including texture maps or surface coloring in the 3-D morph. We have also developed methods for creating distance transforms from 3-D polygonal meshes and MR/CAT-generated volume datasets. We will use these capabilities to generate morphing sequences between different types of models. Future work will also focus on better computational schemes, including parallel processing, in order to achieve 3-D morphs at interactive rates.

9 Acknowledgements

We would like to thank Dr. Alan Barr and the other members of the Caltech Computer Graphics Group for their support and assistance. Sean Mauch developed the Fast Marching algorithm and software that was used for scan converting the models in the morphing examples. Timothy Doyle created the dart model. This work was financially supported by the National Science Foundation (ASC-89-20219), as part of the STC for Computer Graphics and Scientific Visualization; the National Institute on Drug Abuse, the National Institute of Mental Health and the NSF, as part of the Human Brain Project; and the Volume Visualization Program of the Office of Naval Research (N00014-97-0227). Additional equipment grants were provided by SGI, Hewlett-Packard, IBM, and Digital Equipment Corporation. This work was initially funded by the former shareholders of the European Computer-Industry Research Centre: Bull SA, ICL PLC, and Siemens AG.

References

[1] David Adalstein and James A. Sethian. A fast level set method for propagating interfaces. *Journal of Computational Physics*, pages 269–277, 1995.

[2] T.F. Banchoff and P.J. Giblin. Global theorems for symmetry sets of smooth curves and polygons in the plane. *Proc. Royal Soc. Edinburgh*, 106A:221–231, 1987.

[3] A. Barr. Superquadrics and angle-preserving transformations. *IEEE Computer Graphics and Applications*, 1(1):11–23, 1981.

[4] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 35–42, July 1992.

[5] Harry Blum and Roger N. Nagel. Shape description using weighted symmetric axis features. *Pattern Recognition*, 10:167–180, 1978.

[6] D. Breen and S. Mauch. Generating shaded offset surfaces with distance, closest-point and color volumes. In *Proceedings of the International Workshop on Volume Graphics*, pages 307–320, March 1999.

[7] D. Breen, S. Mauch, and R. Whitaker. 3D scan conversion of CSG models into distance volumes. In *Proceedings of the 1998 Symposium on Volume Visualization*, pages 7–14. ACM SIGGRAPH, October 1998.

[8] David Breen, Sean Mauch, and Ross Whitaker. 3D scan conversion of CSG models into distance, closest-point and color volumes. In M. Chen, A. Kaufman, and R. Yagel, editors, *Volume Graphics*, chapter 8. Springer, London, 2000.

[9] David T. Chen, Andrei State, and David Banks. Interactive shape metamorphosis. In P. Hanrahan and J. Winget, editors, *1995 Symposium on Interactive 3D Graphics*, pages 43–44. ACM SIGGRAPH, April 1995.

[10] M. Chen, M.W. Jones, and P. Townshend. Volume distortion and morphing using disk fields. *Computers & Graphics*, 20(4):567–575, 1996.

[11] D. Cohen-Or, D. Levin, and A. Solomivici. Three-dimensional distance field metamorphosis. *ACM Transactions on Graphics*, 17(2):116–141, 1998.

[12] D. DeCarlo and J. Gallier. Topological evolution of surfaces. In *Proceedings of Graphics Interface '96*, pages 194–203, May 1996.

[13] C. Fox. *Introduction to Calculus of Variations.* Dover Publications, New York, 1987.

[14] E. Galin and S. Akkouche. Blob metamorphosis based on minkowski sums. *Computer Graphics Forum*, 15(3):143–153, 1996. Eurographics '96 Conference issue.

[15] P. Getto and D. Breen. An object-oriented architecture for a computer animation system. *The Visual Computer*, 6(2):79–92, March 1990.

[16] S. Gibson. Using distance maps for accurate surface representation in sampled volumes. In *Proceedings of the 1998 Symposium on Volume Visualization*, pages 23–30. ACM SIGGRAPH, October 1998.

[17] A. Gregory, A. State, M. Lin, D. Manocha, and M. Livingston. Feature-based surface decomposition for correspondence and morphing between polyhedra. In *Proceedings of Computer Animation*, pages 64–71, June 1998.

[18] T. He, S. Wang, and A. Kaufman. Wavelet-based volume morphing. In *Proceedings of IEEE Visualization '94*, pages 85–92. IEEE Press, October 1994.

[19] John F. Hughes. Scheduled Fourier volume morphing. In E.E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 43–46, July 1992.

[20] M.W. Jones. The production of volume data from triangular meshes using voxelisation. *Computer Graphics Forum*, 15(5):311–318, 1996.

[21] Takashi Kanai, Hiromasa Suzuki, and Fumihiko Kimura. Three-dimensional geometric metamorphosis based on harmonic maps. *The Visual Computer*, 14(4):166–176, 1998.

[22] Anil Kaul and Jarek Rossignac. Solid-interpolating deformations: Construction and animation of PIPs. *Computers & Graphics*, 16(1):107–116, 1992.

[23] James R. Kent, Wayne E. Carlson, and Richard E. Parent. Shape transformation for polyhedral objects. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 47–54, July 1992.

[24] Francis Lazarus and Anne Verroust. Metamorphosis of cylinder-like objects. *Journal of Visualization and Computer Animation*, 8:131–146, 1997.

[25] Aaron W.F. Lee, David Dobkin, Wim Sweldens, and Peter Schröder. Multiresolution mesh morphing. In *SIGGRAPH '99 Conference Proceedings*, Annual Conference Series, pages 343–350. Addison Wesley, August 1999.

[26] Apostolos Lerios, Chase D. Garfinkle, and Marc Levoy. Feature-Based volume metamorphosis. In Robert Cook, editor, *SIGGRAPH '95 Conference Proceedings*, Annual Conference Series, pages 449–456. Addison Wesley, August 1995.

[27] W.E. Lorensen and H.E. Cline. Marching Cubes: A high resolution 3D surface construction algorithm. In M.C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 163–169, July 1987.

[28] Ravikanth Malladi, James A. Sethian, and Baba C. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):158–175, 1995.

[29] James V. Miller, David E. Breen, William E. Lorensen, Robert M. O'Bara, and Michael J. Wozny. Geometrically deformed models: A method for extracting closed geometric models from volume data. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 217–226, July 1991.

[30] S. Osher and J. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[31] Richard Parent. Shape transformation by boundary representation interpolation: A recursive approach to establishing face correspondences. *Journal of Visualization and Computer Animation*, 3:219–239, 1992.

[32] B. Payne and A. Toga. Distance field manipulation of surface models. *IEEE Computer Graphics and Applications*, 12(1):65–71, 1992.

[33] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C (2nd edition)*. Cambridge University Press, New York, NY, 1992.

[34] J. R. Rossignac and A. Kaul. AGRELs and BIPs: Metamorphosis as a bezier curve in the space of polyhedra. *Computer Graphics Forum*, 13(3):179–184, 1994. Eurographics '94 Conference issue.

[35] W. Schroeder, W. Lorensen, and S. Linthicum. Implicit modeling of swept surfaces and volumes. In *Proceedings of IEEE Visualization '94*, pages 40–45. IEEE Press, October 1994.

[36] J.A. Sethian. A fast marching level set method for monotonically advancing fronts. In *Proceedings of the National Academy of Science*, volume 93 of 4, pages 1591–1595, 1996.

[37] J.A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, Cambridge, UK, 1999.

[38] J.N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, 1995.

[39] Greg Turk and James F. O'Brien. Shape transformation using variational implicit functions. In *SIGGRAPH '99 Conference Proceedings*, Annual Conference Series, pages 335–342. Addison Wesley, August 1999.

[40] R. Whitaker. Algorithms for implicit deformable models. In *Proceedings of the Fifth International Conference on Computer Vision*, pages 822–827. IEEE, June 1995.

[41] R. Whitaker and D. Breen. Level-set models for the deformation of solid objects. In *Proceedings of the Third International Workshop on Implicit Surfaces*, pages 19–35. Eurographics Association, June 1998.

[42] Ross T. Whitaker. A level-set approach to 3D reconstruction from range data. *International Journal of Computer Vision*, 29(3):203–231, October 1998.

Appendix Automatic Object Alignment

The user may allow the system to automatically calculate the transformation needed to align the source and target objects. While this won't guarantee that the objects will overlap and produce a reasonable morph, it does provide a way for a user to easily create an initial configuration for the morphing sequence. The alignment is accomplished by calculating two affine transformations (consisting of rotation, translation and scaling) from gross geometric measures, which map a point in the global coordinate system of each of the objects into each of their intrinsic coordinate systems. We use the moments of the objects, calculated on point samples, to construct the transformation for each object. The centroid and principal axes of the objects define local coordinate systems for those objects, which we assume are aligned with each other.

The centroid, $\bar{\mathbf{p}}$, of an object is the average position of its internal points. The object is sampled on a regular grid within its bounding box. Each grid point is tested to determine if it is inside or outside the object. If $\mathcal{V}$ is the set of coordinates of internal points and n is the number of points in that set, then

$$\bar{\mathbf{p}} = \frac{1}{n} \sum_{\mathbf{p} \in \mathcal{V}} \mathbf{p}. \tag{27}$$

The principal axes of the objects are the eigenvectors of the covariance matrix associated with each object. The covariance matrix is defined as

$$\mathbf{C} = \begin{pmatrix} c_{xx} & c_{xy} & c_{xz} \\ c_{yx} & c_{yy} & c_{yz} \\ c_{zx} & c_{zy} & c_{zz} \end{pmatrix}, \tag{28}$$

where

$$c_{ij} = \frac{1}{n} \sum_{\mathbf{p}_i, \mathbf{p}_j \in \mathcal{V}} (\mathbf{p}_i - \bar{\mathbf{p}}_i) * (\mathbf{p}_j - \bar{\mathbf{p}}_j), \tag{29}$$

and $i, j \in \{x, y, z\}$. The eigenvectors $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$, which are orthogonal, and eigenvalues $(\lambda_1, \lambda_2, \lambda_3)$, which are positive and real, of the covariance matrix $\mathbf{C}$ are calculated with a QL algorithm utilizing implicit shifts [33].

The eigenvalues provide some information about the relative size of the objects along each local axis. The square root of the eigenvalues gives a rough measure of the object's dimensions (exact dimensions if the object is an ellipsoid). The normalized eigenvectors from the two objects are ordered and matched up according to the value of their corresponding eigenvalues. We assume that the principal axis with the greatest eigenvalue is the Z-axis, the second greatest is the Y-axis, and the principal axis with the smallest eigenvalue is the X-axis. The ordered eigenvectors may be "lined up" to construct a rotation matrix which maps a vector in the global coordinate system of the object into the local coordinate system defined by the object's principal axes. Assuming that the eigenvalues $\lambda_1, \lambda_2, \lambda_3$ are ordered in increasing magnitude, the associated rotation matrix is defined as

$$\mathbf{R} = \begin{pmatrix} \mathbf{e}_1{}^T & \mathbf{e}_2{}^T & \mathbf{e}_3{}^T \end{pmatrix}. \tag{30}$$

The inverse rotation, which maps a vector in an object's local intrinsic coordinate system back into the global coordinate system is simply the transpose of Equation 30,

$$\mathbf{R}^{-1} = \begin{pmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \\ \mathbf{e}_3 \end{pmatrix}. \tag{31}$$

The scaling matrix $\mathbf{S}$ is used to scale the source object Ω_A into the approximate size of the target object Ω_B. It is defined as

$$\begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{pmatrix}, \tag{32}$$

where

$$\begin{aligned} S_x &= \sqrt{\lambda_1^B/\lambda_1^A}, \\ S_y &= \sqrt{\lambda_2^B/\lambda_2^A}, \\ S_z &= \sqrt{\lambda_3^B/\lambda_3^A}. \end{aligned} \tag{33}$$

λ_1^A, λ_2^A, λ_3^A, λ_1^B, λ_2^B, λ_3^B are the eigenvalues associated with the eigenvectors $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$ of object Ω_A and object Ω_B.

A point $\mathbf{p}$ in the global coordinate system of object Ω_A may be mapped into the local intrinsic coordinate system of object Ω_B ($\mathbf{p}'$) imbedded in B's global coordinate system by first subtracting the centroid of A, $\bar{\mathbf{p}}^A$, then applying the rotation matrix $\mathbf{R^A}$. This places point $\mathbf{p}$ into the intrinsic local coordinate system of A, which is assumed to be aligned with B's. The scaling matrix $\mathbf{S}$ is then applied so that the general dimensions of object Ω_A are approximately the same as object Ω_B's. The rotation matrix $\mathbf{R^B}^{-1}$ is applied, and the point is shifted by $\bar{\mathbf{p}}^B$, the centroid of B, which maps the point into the global coordinate system of B. The transformation steps may be summarized as

$$\mathbf{p}' = (\mathbf{p} - \bar{\mathbf{p}}^A)\mathbf{R^A}\mathbf{S}(\mathbf{R^B})^{-1} + \bar{\mathbf{p}}^B. \tag{34}$$