Feature-Based Volume Metamorphosis

Apostolos Lerios, Chase D. Garfinkle, Marc Levoy *

Computer Science Department
Stanford University

Abstract

Image metamorphosis, or image *morphing*, is a popular technique for creating a smooth transition between two images. For synthetic images, transforming and rendering the underlying three-dimensional (3D) models has a number of advantages over morphing between two pre-rendered images. In this paper we consider 3D metamorphosis applied to volume-based representations of objects. We discuss the issues which arise in volume morphing and present a method for creating morphs. Our morphing method has two components: first a warping of the two input volumes, then a blending of the resulting warped volumes. The warping component, an extension of Beier and Neely's image warping technique to 3D, is feature-based and allows fine user control, thus ensuring realistic looking intermediate objects. In addition, our warping method is amenable to an efficient approximation which gives a 50 times speedup and is computable to arbitrary accuracy. Also, our technique corrects the ghosting problem present in Beier and Neely's technique. The second component of the morphing process, blending, is also under user control; this guarantees smooth transitions in the renderings.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism.

Additional Keywords: Volume morphing, warping, rendering; sculpting; shape interpolation, transformation, blending; computer animation.

1 Introduction

1.1 Image Morphing versus 3D Morphing

Image morphing, the construction of an image sequence depicting a gradual transition between two images, has been extensively investigated [21] [2] [6] [16]. For images generated from 3D models, there is an alternative to morphing the images themselves: *3D morphing* generates intermediate 3D models, the morphs, directly from the given models; the morphs are then rendered to produce an image sequence depicting the transformation. 3D morphing overcomes the following shortcomings of 2D morphing as applied to images generated from 3D models:

- In 3D morphing, creating the morphs is independent of the viewing and lighting parameters. Hence, we can create a morph sequence once, and then experiment with various camera angles and lighting conditions during rendering. In 2D morphing, a new morph must be recomputed every time we wish to alter our viewpoint or the illumination of the 3D model.
- 2D techniques, lacking information on the model's spatial configuration, are unable to correctly handle changes in illumination and visibility. Two examples of this type of artifact are: (i) Shadows and highlights fail to match shape changes occurring in the morph. (ii) When a feature of the 3D object is not visible in the original 2D image, this feature cannot be made to appear during the morph; for example, when a singing actor needs to open her mouth during a morph, pulling her lips apart thickens the lips instead of revealing her teeth.

1.2 Geometric versus Volumetric 3D Models

The models subjected to 3D morphing can be described either by geometric primitives or by volumes (volumetric data sets). Each representation requires different morphing algorithms. This dichotomy parallels the separation of 2D morphing techniques into those that operate on raster images [21] [2] [6], and those that assume vector-based image representations [16]. We believe that volume-based descriptions are more appropriate for 3D morphing for the following reasons:

- The quality and applicability of geometric 3D morphing techniques [12] is highly dependent on the models' geometric primitives and their topological properties. Volume morphing is independent of object geometries and topologies, and thus imposes no such restrictions on the objects which can be successfully morphed.
- Volume morphing may be applied to objects represented either by geometric primitives or by volumes. Geometric descriptions can be easily converted to high-quality volume representations, as we will see in section 2. The reverse process produces topologically complex objects, usually inappropriate for geometric morphing.

1.3 Volume Morphing

The 3D volume morphing problem can be stated as follows. Given two volumes $\mathcal{S}$ and $\mathcal{T}$, henceforth called the *source* and *target* volumes, we must produce a sequence of intermediate volumes, the *morphs*, meeting the following two conditions:

Realism: The morphs should be realistic objects which have plausible 3D geometry and which retain the essential features of the source and target.

Smoothness: The renderings of the morphs must depict a smooth transition from $\mathcal{S}$ to $\mathcal{T}$.

From the former condition stems the major challenge in designing a 3D morphing system: as automatic feature recognition and matching have yet to equal human perception, user input is crucial in defining the transformation of the objects. The challenge for the designer

*Center for Integrated Systems, Stanford University, Stanford, CA 94305
{lerios,cgar,levoy}@cs.stanford.edu
http://www-graphics.stanford.edu/

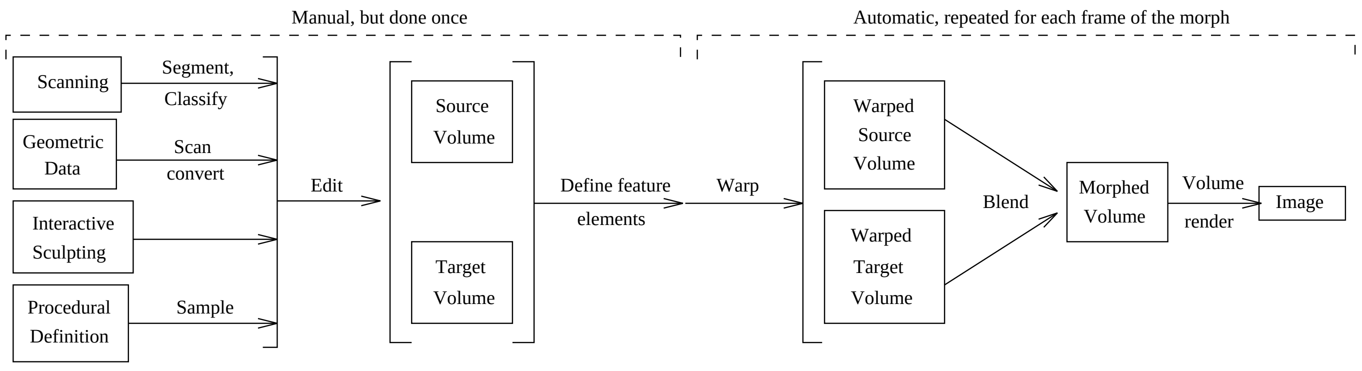


Figure 1: Data flow in a morphing system. Editing comprises retouching and aligning the volumes for cosmetic reasons.

of a 3D morphing technique is two-fold: the morphing algorithm must permit fine user control and the accompanying user interface (UI) should be intuitive.

Our solution to 3D morphing attempts to meet both conditions of the morphing problem, while allowing a simple, yet powerful UI. To this end, we create each morph in two steps (see figure 1):

Warping: $\mathcal{S}$ and $\mathcal{T}$ are warped to obtain volumes $\mathcal{S}'$ and $\mathcal{T}'$. Our warping technique allows the animator to define quickly the exact shape of objects represented in $\mathcal{S}'$ and $\mathcal{T}'$, thus meeting the realism condition.

Blending: $\mathcal{S}'$ and $\mathcal{T}'$ are combined into one volume, the morph. Our blending technique provides the user with sufficient control to create a smooth morph.

1.4 Prior Work

Prior work on feature-based 2D morphing [2] will be discussed in section 3.

Prior work in volume morphing comprises [9], [8], and [5]. These approaches can be summarized in terms of our warping/blending framework.

[5] and [8] have both presented warping techniques. [5] examined the theory of extending selected 2D warping techniques into 3D. A UI was not presented, however, and only morphs of simple objects were shown. [8] presents an algorithm which attempts to automatically identify correspondences between the volumes, without the aid of user input.

[9] and [8] have suggested using a frequency or wavelet representation of the volumes to perform the blending, allowing different interpolation schedules across subbands. In addition, they have observed that isosurfaces of the morphs may move abruptly, or even completely disappear and reappear as the morph progresses, destroying its continuity. This suggests that volume rendering may be superior to isosurface extraction for rendering the morphs.

Our paper is partitioned into the following sections: Section 2 covers volume acquisition methods. Sections 3 and 4 present the warping and blending steps of our morphing algorithm. Section 4.2 describes an efficient implementation of our warping method and section 5 discusses our results. We conclude with suggestions for future work and applications in section 6.

2 Volume Acquisition

Volume data may be acquired in several ways, the most common of which are listed below.

Scanned volumes: Some scanning technologies, such as Computerized Tomography (CT) or Magnetic Resonance Imaging (MRI) generate volume data. Figures 5(a) and 5(c) show CT scans of a human and an orangutan head, respectively.

Scan converted geometric models: A geometric model can be *voxelized* [10], preferably with antialiasing [20], generating a volume-based representation of the model. Figures 6(a), 6(b), 7(a), and 7(b) show examples of scan-converted volumes.

Interactive sculpting: Interactive modeling, or *sculpting* [19] [7], can generate volume data directly.

Procedural definition: Hypertexture volumes [15] can be defined procedurally by functions over 3D space.

3 Warping

The first step in the volume morphing pipeline is warping the source and target volumes $\mathcal{S}$ and $\mathcal{T}$. Volume warping has been the subject of several investigations in computer graphics, computer vision, and medicine. Warping techniques can be coarsely classified into two groups: (i) Techniques that allow only minimal user control, consisting of at most a few scalar parameters. These algorithms automatically determine similarities between two volumes, and then seek the warp which transforms the first volume to the second one [18]. (ii) Techniques in which user control consists of manually specifying the warp for a collection of points in the volume. The rest of the volume is then warped by interpolating the warping function. This group of algorithms includes free-form deformations [17], as well as semi-automatic medical data alignment [18].

As stated in section 1.3, user control over the warps is crucial in designing good morphs. Point-to-point mapping methods [21], in the form of either regular lattices or scattered points [13], have worked in 2D. However, regular grids provide a cumbersome interface in 2D; in 3D they would likely become unmanageable. Also, prohibitively many scattered points are needed to adequately specify a 3D warp.

Our solution is a feature-based approach extending the work of [2] into the 3D domain. The next two sections will introduce our feature-based 3D warping and discuss the UI to feature specification.

3.1 Feature-Based 3D Warping using Fields

The purpose of a *feature element* is to identify a feature of an object. For example, consider the X-29 plane of figure 6(b); an element can be used to delineate the nose of the plane. In feature-based morphing, elements come in *pairs*, one element in the source volume $\mathcal{S}$, and its counterpart in the target volume $\mathcal{T}$. A pair of elements identifies corresponding features in the two volumes, i.e. features that should be transformed to one another during the morph. For instance, when morphing the dart of figure 6(a) to the X-29 plane, the tip of the dart should turn into the nose of the plane. In order to obtain good morphs, we need to specify a *collection* of element pairs which define the overall correspondence of the two objects. These element pairs interact like magnets shaping a pliable volume:

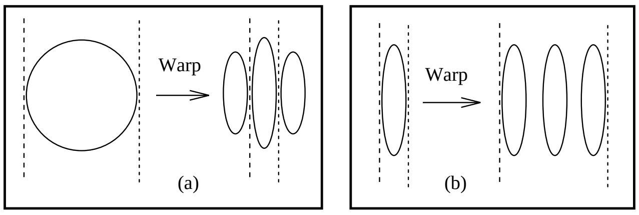


Figure 2: 2D warp artifacts (not to scale). (a) shows the result of squeezing a circle using two feature lines placed on opposite sides of the circle. The warped circle spills outside the corresponding, closely spaced, lines. Similarly, in (b), the narrow ellipsoid with two lines on either side does not expand to a circle when the lines are drawn apart; we get instead three copies of the ellipsoid.

while a single magnet can only move, turn, and stretch the volume, multiple magnets generate interacting fields, termed *influence fields*, which combine to shape the volume in complex ways. Sculpting with multiple magnets becomes easier if we have magnets of various kinds in our toolbox, each magnet generating a differently shaped influence field. The elements in our toolkit are points, line segments, rectangles, and boxes.

In the following presentation, we first describe individual elements, and discuss how they identify features. We then show how a pair of elements guarantees that corresponding features are transformed to one another during the morph. Finally, we discuss how multiple element pairs interact.

Individual Feature Elements

Individual feature elements should be designed in a manner such that they can delineate any feature an object may possess. However, expressiveness should not sacrifice simplicity, as complex features can still be matched by a group of simple elements. Hence, the defining *attributes* of our elements encode only the essential characteristics of features:

Spatial configuration: The feature's position and orientation are encoded in an element's *local coordinate system*, comprising four vectors. These are the position vector of its origin $\mathbf{c}$, and three mutually perpendicular unit vectors $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$, defining the directions of the coordinate axes. The element's *scaling factors* s_x, s_y, and s_z define a feature's extent along each of the principal axes.

Dimensionality: The dimensionality of a feature depends on the subjective perception of a feature's relative size in each dimension: the tip of the plane's nose is perceived as a point, the edge of the plane's wing as a line, the dart's fin as a surface, and the dart's shaft as a volume. Accordingly, our simplified elements have a *type*, which can be a point, segment, rectangle, or box. In our magnetic sculpting analogy, the element type determines the shape of its influence field. For example, a box magnet defines the path of points within and near the box; points further from the box are influenced less as their distance increases.

The reader familiar with the 2D technique of [2] will notice two differences between our 3D elements and a direct extention of 2D feature lines into 3D; in fact, these are the only differences as far as the warping algorithm is concerned.

First, in the 2D technique, the shape of a feature line's influence field is controlled by two manually specified parameters. Instead, we provide four simple types of influence fields — point, segment, rectangle, and box — thus allowing for a more intuitive, yet equally powerful, UI.

Second, our feature elements encode the 3D extent of a 3D feature via the scaling factors s_x, s_y, and s_z; by contrast, feature lines in [2] capture only the 1D extent of a 2D feature, in the direction of each feature line. These scaling factors introduce additional degrees of freedom for each feature element. In the majority of situations, these extra degrees have a minor effect on the warp and may thus be ignored. However, under extreme warps, they permit the user to solve the *ghosting* problem, documented in [2] and illustrated in figure 2. For instance, in part (b) of this example, the ellipsoid is replicated because each feature line requires that an unscaled ellipsoid appear by its side: the feature lines in [2] cannot specify any stretching in the perpendicular direction. However, in a 2D analogue of our technique, the user would use the lines' scaling factors to stretch the ellipsoid. First, the user would encode the ellipsoid's width in the scaling factors of the original feature lines. Then, in order to stretch the ellipsoid into a circle, the user would not only move the feature lines apart, but will also make the lines' scaling factors encode the desired new width of the ellipsoid. In fact, using our technique, a single feature line suffices to turn the ellipsoid into a circle.

Element Pairs

As in the 2D morphing system of [2], the animator identifies two corresponding features in $\mathcal{S}$ and $\mathcal{T}$, by defining a pair of elements (e_s, e_t). These features should be transformed to one another during the morph. Such a transformation requires that the feature of $\mathcal{S}$ be moved, turned, and stretched to match respectively the position, orientation, and size of the corresponding feature of $\mathcal{T}$. Consequently, for each frame of the morph, our warp should generate a volume $\mathcal{S}'$ from $\mathcal{S}$ with the following property: the feature of $\mathcal{S}$ should possess an intermediate position, orientation and size in $\mathcal{S}'$. This is achieved by computing the warp in two steps:

Interpolation: We interpolate the local coordinate systems[1] and scaling factors of elements e_s and e_t to produce an *interpolated element* e'. This element encodes the spatial configuration of the feature in $\mathcal{S}'$.

Inverse mapping: For every point in $\mathbf{p}'$ of $\mathcal{S}'$, we find the corresponding point $\mathbf{p}$ in $\mathcal{S}$ in two simple steps (see figure 3): (i) We find the coordinates of $\mathbf{p}'$ in the scaled local system of element e' by

$$\begin{aligned} p_x &= (\mathbf{p}' - \mathbf{c}') \cdot \mathbf{x}'/s'_x \\ p_y &= (\mathbf{p}' - \mathbf{c}') \cdot \mathbf{y}'/s'_y \\ p_z &= (\mathbf{p}' - \mathbf{c}') \cdot \mathbf{z}'/s'_z. \end{aligned}$$

(ii) $\mathbf{p}$ is the point with coordinates p_x, p_y and p_z in the scaled local system of element e_s, i.e. the point $\mathbf{c} + p_x s_x \mathbf{x} + p_y s_y \mathbf{y} + p_z s_z \mathbf{z}$.[2]

Collections of Element Pairs

In extending the warping algorithm of the previous paragraph to multiple element pairs, we adhere to the intuitive mental model of magnetic sculpting used in [2]. Each pair of elements defines a field that extends throughout the volume. A collection of element pairs defines a collection of fields, all of which influence each point in the volume. We therefore use a weighted averaging scheme to determine the point $\mathbf{p}$ in $\mathcal{S}$ that corresponds to each point $\mathbf{p}'$ of $\mathcal{S}'$. That is, we first compute to what point $\mathbf{p}_i$ each element pair would map $\mathbf{p}'$ in the absence of all other pairs; then, we average the $\mathbf{p}_i$'s using a weighting function that depends on the distance of $\mathbf{p}'$ to the interpolated elements e'_i.

Our weighting scheme uses an inverse square law: $\mathbf{p}_i$ is weighted by $(d+\epsilon)^{-2}$ where d is the distance of $\mathbf{p}'$ from the element e'_i; ϵ is a

[1] The axes directions $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are interpolated in spherical coordinates to ensure smooth rotations.

[2] $\mathcal{T}$ is warped into $\mathcal{T}'$ in a similar way, the only difference being that e_t is used in this last step in place of e_s.

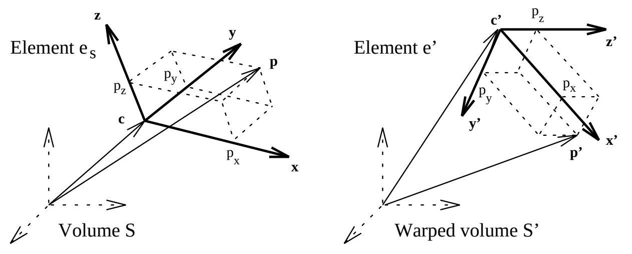


Figure 3: Single element warp. In order to find the point $\mathbf{p}$ in volume $\mathcal{S}$ that corresponds to $\mathbf{p}'$ in $\mathcal{S}'$, we first find the coordinates (p_x, p_y, p_z) of $\mathbf{p}'$ in the scaled local system of element e'; $\mathbf{p}$ is then the point with coordinates (p_x, p_y, p_z) in the scaled local system of element e_s. To simplify the figure, we have assumed unity scaling factors for all elements.

small constant used to avoid division by zero.[3] The type of element e'_i determines how d is calculated:

Points: d is the distance between $\mathbf{p}'$ and the origin $\mathbf{c}$ of the local coordinate system of element e'_i. This definition is identical to [21].

Segments: The element is treated as a line segment centered at the origin $\mathbf{c}$, aligned with the local x-axis and having length s_x; d is the distance of $\mathbf{p}'$ from this line segment. This definition is identical to [2].

Rectangles: Rectangles have the same center and x extent as segments, but also extend into a second dimension, having width s_y along the local y-axis. d is zero if $\mathbf{p}'$ is on the rectangle, otherwise it is the distance of $\mathbf{p}'$ from the rectangle. This definition extends segments to area elements.

Boxes: Boxes add depth to rectangles, thus extending for s_z units along the local z-axis. d is zero if $\mathbf{p}'$ is within the box, otherwise it is the distance of $\mathbf{p}'$ from the box's surface.

The reader will notice that the point, segment, and rectangle element types are redundant, as far as the mathematical formulation of our warp is concerned. However, a variety of element types maintains best the intuitive conceptual analogy to magnetic sculpting.

3.2 User Interface

The UI to the warping algorithm has to depict the source and target volumes, in conjunction with the feature elements. Hardware-assisted volume rendering [4] makes possible a UI solely based on direct visualization of the volumes, with the embedded elements interactively scan-converted. Using a low-end rendering pipeline, however, the UI has to resort to geometric representations of the models embedded in the volumes. These geometric representations can be obtained in either of two ways:

- Pre-existing volumes are visualized by isosurface extraction via marching cubes [14]. Several different isosurfaces can be extracted to visualize all prominent features of the volume, a volume rendering guiding the extraction process.
- For volumes that were obtained by scan converting geometric models, the original model can be used.

Once geometric representations of the models are available, the animator can use the commercial modeler of his/her choice to specify the elements. Our system, shown in figure 6(d), is based on Inventor, the Silicon Graphics (SGI) 3D programming environment. Models are drawn in user-defined materials, usually translucent, in order to distinguish them from the feature elements. These, in turn, are drawn in such a way that their attributes — local coordinate system, scaling factors, and dimensionality — are graphically depicted and altered using a minimal set of widgets.

4 Blending

The warping step has produced two warped volumes $\mathcal{S}'$ and $\mathcal{T}'$ from the source and target volumes $\mathcal{S}$ and $\mathcal{T}$. Any practical warp is likely to misalign some features of $\mathcal{S}$ and $\mathcal{T}$, possibly because these were not specifically delineated by feature elements. Even if perfectly aligned, matching features may have different opacities. These areas of the morph, collectively called *mismatches*, will have to be smoothly faded in/out in the rendered sequence, in order to maintain the illusion of a smooth transformation. This is the goal of blending.

We have two alternatives for performing this blending step. It may either be done by cross-dissolving images rendered from $\mathcal{S}'$ and $\mathcal{T}'$, which we call 2.5D morphing, or by cross-dissolving the volumes themselves, and rendering the result, i.e. a full 3D morph. The 2.5D approach produces smooth image sequences and provides the view and lighting independence of 3D morphing discussed in section 1.1; however, some disadvantages of 2D morphing are reintroduced, such as incorrect lighting and occlusions. Consequently, 2.5D morphs do not look as realistic as 3D morphs. For example, the "missing link" of figure 5(f) lacks distinct teeth, and the base of the skull appears unrealistically transparent.

For this reason, we decided to investigate full 3D morphing, whereby we blend the warped volumes by interpolating their voxel values. The interpolation weight $w(t)$ is a function that varies over time, where "time" is the normalized frame number[4]. We have the option of using either a linear or non-linear $w(t)$.

4.1 Linear Cross-Dissolving

The pixel cross-dissolving of 2D morphing suggests a linear $w(t)$. Indeed, it works well for blending the color information of $\mathcal{S}'$ and $\mathcal{T}'$. However, it fails to interpolate opacities in a manner such that the rendered morph sequence appears smooth. This is due to the exponential dependence of the color of a ray cast through the volume on the opacities of the voxels it encounters. This phenomenon is illustrated in the morph of figure 5. In particular, the morph abruptly snaps into the source and target volumes if a linear $w(t)$ is used: figure 5(g) shows that at time 0.06, very early in the morph, the empty space towards the front of the human head has already been filled in by the warped orangutan volume.

4.2 Non-Linear Cross-Dissolving

In order to obtain smoothly progressing renderings, we would like to compensate for the exponential dependence of rendered color on opacity as we blend $\mathcal{S}'$ and $\mathcal{T}'$. This can be done by devising an appropriate $w(t)$.

In principle, there cannot exist an ideal compensating $w(t)$. The exact relationship between rendered color and opacity depends on the distance the ray travels through voxels with this opacity. Hence a globally applied $w(t)$ cannot compensate at once for all mismatches since they have different thickness. Even a locally chosen $w(t)$ cannot work, as different viewpoints cast different rays through the morph.

In practice, the mismatches between $\mathcal{S}'$ and $\mathcal{T}'$ are small in number and extent. Hence, the above theoretical objections do not prevent us from empirically deriving a successful $w(t)$. Our design goal is to compensate for the exponential relation of rendered color

[3] Distance measurements postulate cubical volumes of unit side length. Also, we always set ϵ to 0.001.

[4] In other words, "time" is a real number linearly increasing from 0 to 1 as the morph unfolds.

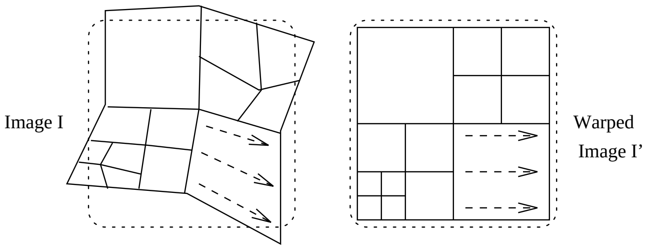


Figure 4: 2D analogue of piecewise linear warping. A warped image $\mathcal{I}'$ is first subdivided by an adaptive grid of squares, here marked by solid lines. Then, each square vertex is warped into $\mathcal{I}$. Finally, pixels in the interior of each grid cell are warped by bilinearly interpolating the warped positions of the vertices. The dashed arrows demonstrate how the interior of the bottom right square is warped. The dotted rectangles mark image buffer borders.

to opacity by interpolating opacities at the rate of an inverse exponential. The sigmoid curve given by

$$\frac{\tan^{-1}(2s(t-0.5))}{2\tan^{-1}s}+\frac{1}{2}$$

satisfies this requirement. It suppresses the contribution of $\mathcal{T}'$'s opacity in the early part of the morph, the degree of suppression controlled by the *blending parameter* s. Similarly, the contribution of $\mathcal{T}'$'s opacity is enhanced in the latter part of the morph. Figure 5(h), illustrates the application of compensated interpolation to the morph of figure 5: in contrast to figure 5(g), figure 5(h) looks very much like the human head, as an early frame in the morph sequence should.
sectionPerformance and Optimization
A performance drawback of our feature-based warping technique is that each point in the warped volume is influenced by all elements, since the influence fields never decay to zero. It follows that the time to warp a volume is proportional to the number of element pairs. An efficient C++ implementation, using incremental calculations, needs 160 minutes to warp a single 300^3 volume with 30 element pairs on an SGI Indigo 2.

We have implemented two optimizations which greatly accelerate the computation of the warped volume $\mathcal{V}'$, where we henceforth use $\mathcal{V}$ to denote either $\mathcal{S}$ or $\mathcal{T}$. First, we approximate the spatially non-linear warping function with a *piecewise linear warp* [13]. Second, we introduce an *octree subdivision* over $\mathcal{V}$.

4.3 Piecewise Linear Approximation

The 2D form of this optimization, shown in figure 4, illustrates its key steps within the familiar framework of image warping. In 3D, piecewise linear warping begins by subdividing $\mathcal{V}'$ into a coarse, 3D, regular grid, and warping the grid vertices into $\mathcal{V}$, using the algorithm of section 3.1. The voxels in the interior of each cubic grid cell are then warped by trilinearly interpolating the warped positions of the cube's vertices. Using this method, $\mathcal{V}'$ can be computed by scan-converting each cube in turn. Essentially, we treat $\mathcal{V}$ as a solid texture, with the warped grid specifying the mapping into texture space. The expensive computation of section 3.1 is now performed only for a small fraction of the voxels, and scan-conversion dominates the warping time.

This piecewise linear approximation will not accurately capture the warp in highly non-linear regions, unless we use a very fine grid. However, computing a uniformly fine sampling of the warp defeats the efficiency gain of this approach. Hence, we use an adaptive grid which is subdivided more finely in regions where the warp is highly non-linear. To determine whether a grid cell requires subdivision, we compare the exact and approximated warped positions of several points within the cell. If the error is above a user-specified threshold, the cell is subdivided further. In order to reduce computation, we use the vertices of the next-higher resolution grid as the points at which to measure the error. Using this technique, the non-linear warp can be approximated to arbitrary accuracy.[5]

Since we are subsampling the warping function, it is possible that this algorithm will fail to subdivide non-linear regions. Analytically bounding the variance of the warping function would guarantee conservative subdivision. However, this is unnecessary in practice, as the warps used in generating morphs generally do not possess large high-frequency components.

This optimization has been applied to 2D morphing systems, as well; by using common texture-mapping hardware to warp the images, 2D morphs can be generated at interactive rates [1].

4.4 Octree Subdivision

$\mathcal{V}$ usually contains large "empty" regions, that is, regions which are completely transparent. The warp will map these parts of $\mathcal{V}$ into empty regions of $\mathcal{V}'$. Scan conversion, as described above, need not take place when a warped grid cell is wholly contained within such a region. By constructing an octree over $\mathcal{V}$, we can identify many such cells, and thus avoid scan converting them.

4.5 Implementation

Our optimized warping method warps a 300^3 volume in approximately 3 minutes per frame on an SGI Indigo 2. This represents a speedup of 50 over the unoptimized algorithm, without noticeable loss of quality. The running time is still dominated by scan-conversion and resampling, both of which can be accelerated by the use of 3D texture-mapping hardware.

5 Results and Conclusions

Our color figures show the source volumes, target volumes, and halfway morphs for three morph sequences we have created.

The human and orangutan volumes shown in figures 5(a) and 5(c) were warped using 26 element pairs to produce the volumes of figures 5(b) and 5(d) at the midpoint of the morph. The blended middle morph appears in figure 5(e).

Figures 6 and 7 show two examples of color morphs, requiring 37 and 29 element pairs, respectively. The UI, displaying the elements used to control the morph of figure 6, is shown in 6(d).

The total time it takes to compose a 50-frame morph sequence for 300^3 volumes comprises all the steps shown on figure 1. Our experience is that about 24 hours are necessary to turn design into reality on an SGI Indigo 2:

Hours	Task
10	CT scan segmentation, classification, retouching
1	Scan conversion of geometric model
8	Feature element definition (novice)
3	Feature element definition (expert)
5	Warping
3	Blending: 1 hour for each s: 2, 4, 6; retain best
4	Hi-res volume rendering (monochrome)
12	Hi-res volume rendering (color)

We have presented a two step feature-based technique for realistic and smooth metamorphosis between two 3D models represented by volumes. In the first step, our feature-based warping algorithm allows fine user control, and thus ensures realistic morphs. In addition, our warping method is amenable to an efficient, adaptive approximation which gives a 50 times speedup. Also, our technique

[5] We always use an error tolerance of a single voxel width and an initial subdivision of 15^3 cells.

corrects the ghosting problem of [2]. In the second step, our user-controlled blending ensures that the rendered morph sequence appears smooth.

6 Future Work and Applications

We see the potential for improving 3D morphing in three primary aspects:

Warping Techniques: Improved warping methods could allow for finer user control, as well as smoother, possibly spline-based, interpolation of the warping function across the volume. More complex, but more expressive feature elements [11] may also be designed.

User Interface: We envision improving our UI by adding computer-assisted feature identification: the computer suggesting features by landmark data extraction [18], 3D edge identification, or, as in 2D morphing, by motion estimation [6]. Also, we are considering giving the user more flexible control over the movement of feature elements during the morph, i.e. the rule by which interpolated elements are constructed, perhaps by key-framed or spline-path motion.

Blending: Blending can be improved by allowing local definition of the blending rate, associating an interpolation schedule with each feature element.

Morphing's primary application has been in the entertainment industry. However, it can also be used as a general visualization tool for illustration and teaching purposes [3]; for example, our orangutan to human morph could be used as a means of visualizing Darwinian evolution. Finally, our feature-based warping technique can be used in modeling and sculpting.

Acknowledgments

Philippe Lacroute helped render our morphs, and designed part of the dart to X-29 fly-by movie shown on our video. We used the horse mesh courtesy of Rhythm & Hues, the color added by Greg Turk. John W. Rick provided the plastic cast of the orangutan head and Paul F. Hemler arranged the CT scan. Jonathan J. Chew and David Ofelt helped keep our computer resources in operation.

References

[1] T. Beier and S. Neely. Pacific Data Images. Personal communication.

[2] T. Beier and S. Neely. Feature-based image metamorphosis. In *Computer Graphics*, vol 26(2), pp 35–42, New York, NY, July 1992. Proceedings of SIGGRAPH '92.

[3] B. P. Bergeron. Morphing as a means of generating variation in visual medical teaching materials. *Computers in Biology and Medicine*, 24(1):11–18, Jan. 1994.

[4] B. Cabral, N. Cam, and J. Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In A. Kaufman and W. Krueger, editors, *Proceedings of the 1994 Symposium on Volume Visualization*, pp 91–98, New York, NY, Oct. 1994. ACM SIGGRAPH and IEEE Computer Society.

[5] M. Chen, M. W. Jones, and P. Townsend. Methods for volume metamorphosis. To appear in *Image Processing for Broadcast and Video Production*, Y. Paker and S. Wilbur editors, Springer-Verlag, London, 1995.

[6] M. Covell and M. Withgott. Spanning the gap between motion estimation and morphing. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, vol 5, pp 213–216, New York, NY, 1994. IEEE.

[7] T. A. Galyean and J. F. Hughes. Sculpting: An interactive volumetric modeling technique. In *Computer Graphics*, vol 25(4), pp 267–274, New York, NY, July 1991. Proceedings of SIGGRAPH '91.

[8] T. He, S. Wang, and A. Kaufman. Wavelet-based volume morphing. In D. Bergeron and A. Kaufman, editors, *Proceedings of Visualization '94*, pp 85–91, Los Alamitos, CA, Oct. 1994. IEEE Computer Society and ACM SIGGRAPH.

[9] J. F. Hughes. Scheduled Fourier volume morphing. In *Computer Graphics*, vol 26(2), pp 43–46, New York, NY, July 1992. Proceedings of SIGGRAPH '92.

[10] A. Kaufman, D. Cohen, and R. Yagel. Volume graphics. *Computer*, 26(7):51–64, July 1993.

[11] A. Kaul and J. Rossignac. Solid-interpolating deformations: Construction and animation of PIPs. In F. H. Post and W. Barth, editors, *Eurographics '91*, pp 493–505, Amsterdam, The Netherlands, Sept. 1991. Eurographics Association, North-Holland.

[12] J. R. Kent, W. E. Carlson, and R. E. Parent. Shape transformation for polyhedral objects. In *Computer Graphics*, vol 26(2), pp 47–54, New York, NY, July 1992. Proceedings of SIGGRAPH '92.

[13] P. Litwinowicz. Efficient techniques for interactive texture placement. In *Computer Graphics Proceedings*, Annual Conference Series, pp 119–122, New York, NY, July 1994. Conference Proceedings of SIGGRAPH '94.

[14] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3-D surface construction algorithm. In *Computer Graphics*, vol 21(4), pp 163–169, New York, NY, July 1987. Proceedings of SIGGRAPH '87.

[15] K. Perlin and E. M. Hoffert. Hypertexture. In *Computer Graphics*, vol 23(3), pp 253–262, New York, NY, July 1989. Proceedings of SIGGRAPH '89.

[16] T. W. Sedeberg, P. Gao, G. Wang, and H. Mu. 2-D shape blending: An intrinsic solution to the vertex path problem. In *Computer Graphics Proceedings*, Annual Conference Series, pp 15–18, New York, NY, Aug. 1993. Conference Proceedings of SIGGRAPH '93.

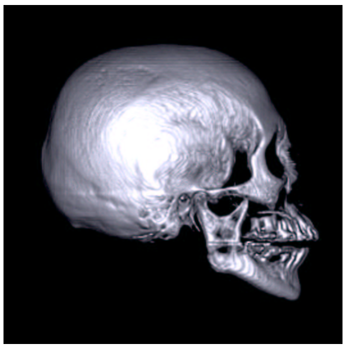
[17] T. W. Sederberg and S. R. Parry. Free-form deformations of solid geometric models. In *Computer Graphics*, vol 20(4), pp 151–160, New York, NY, Aug. 1986. Proceedings of SIGGRAPH '86.

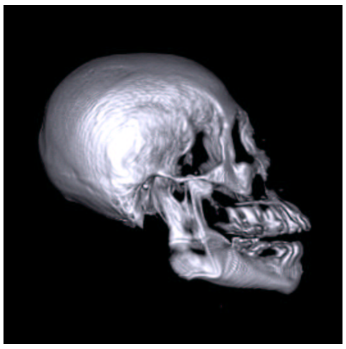
[18] P. A. van den Elsen, E.-J. D. Pol, and M. A. Viergever. Medical image matching — a review with classification. *IEEE Engineering in Medicine and Biology Magazine*, 12(1):26–39, Mar. 1993.

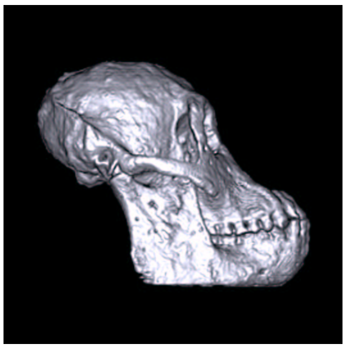
[19] S. W. Wang and A. Kaufman. Volume sculpting. In *Proceedings of 1995 Symposium on Interactive 3D Graphics*, pp 151–156, 214, New York, NY, Apr. 1995. ACM SIGGRAPH.

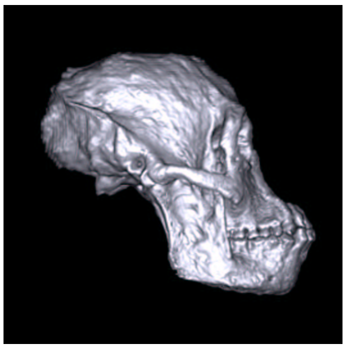
[20] S. W. Wang and A. E. Kaufman. Volume sampled voxelization of geometric primitives. In G. M. Nielson and D. Bergeron, editors, *Proceedings of Visualization '93*, pp 78–84, Los Alamitos, CA, Oct. 1993. IEEE Computer Society and ACM SIGGRAPH.

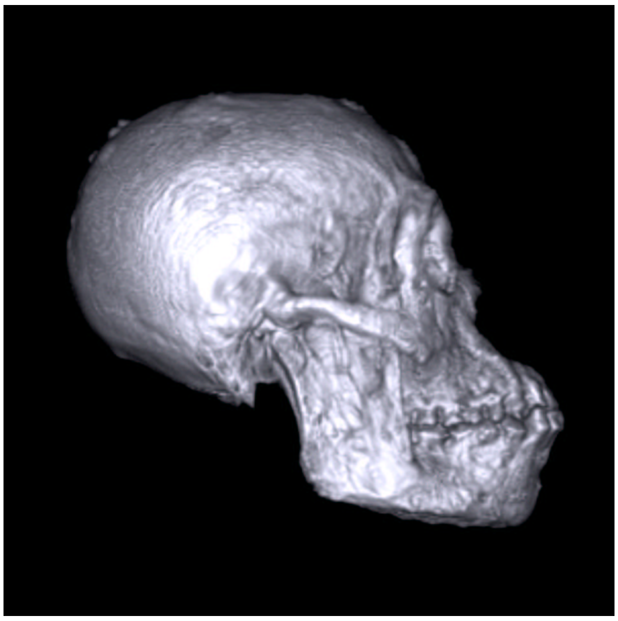
[21] G. Wolberg. *Digital Image Warping*. IEEE Computer Society P., Los Alamitos, CA, 1990.

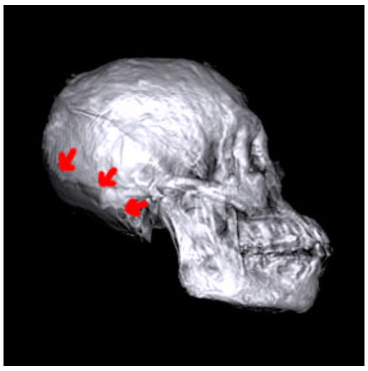

(a) Original CT human head.

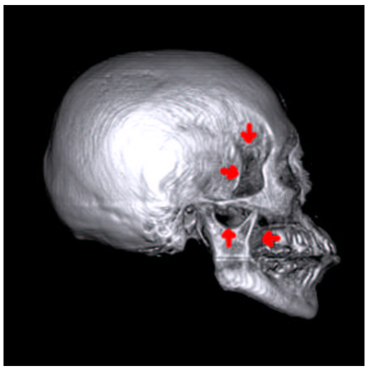

(b) Human head warped to midpoint of morph.


(c) Original CT orangutan head.


(d) Orangutan head warped to midpoint of morph.


(e) 3D volume morph halfway between human head and orangutan head.


(f) Cross−dissolve of figures 5(b) and 5(d) illustrating a drawback of 2.5 D morphing. The base of the skull (indicated by red arrows) appears unrealistically transparent, and the teeth are indistinct, compared to the full 3D morph shown in figure 5(e).


(g) Volume morph at time 0.06 using linear interpolation of warped volumes. Due to the exponential dependence of rendered color on opacity, the empty space towards the front of the human head has already been filled in by the warped orangutan volume (red arrows).

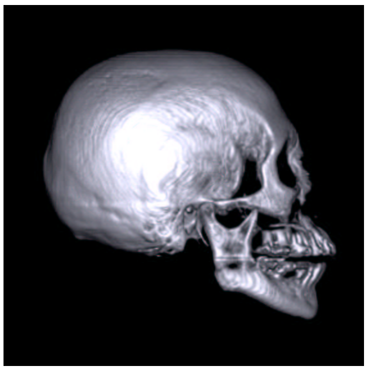
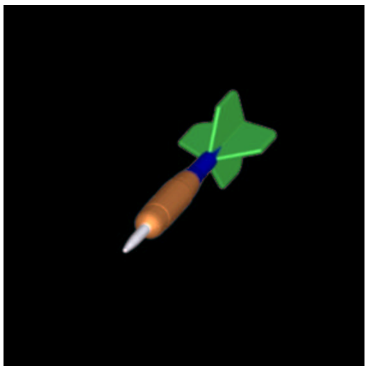
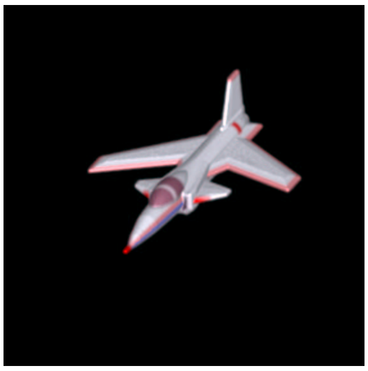
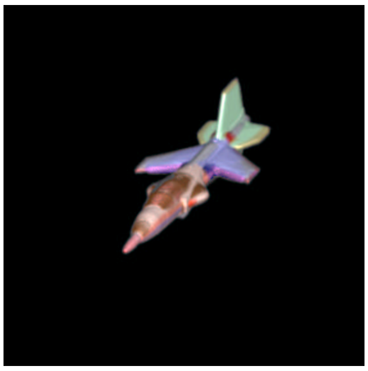
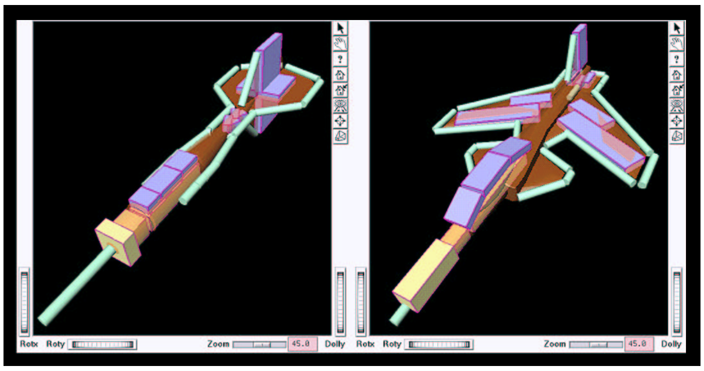

(h) Volume morph at time 0.06 using non−linear interpolation of warped volumes to correct for the exponential dependence of color on opacity. The result is now nearly identical to the human (see section 4.2).

Figure 5: Human to orangutan morph.


(a) Dart volume from scan−converted polygon mesh.

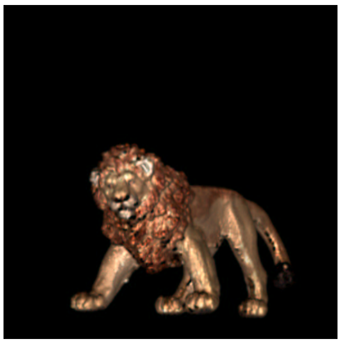

(b) X−29 volume from scan−converted polygon mesh.

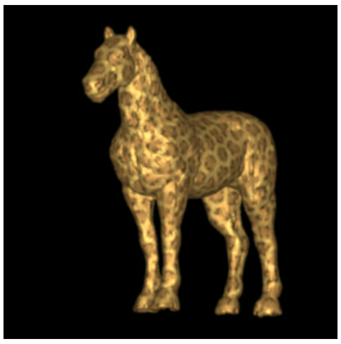

(c) Volume morph halfway between dart and X−29.



(d) User interface showing elements used to establish correspondences between models. Points (not shown), segments, rectangles, and boxes are respectively drawn as pink spheres, green cylinders, narrow blue slabs, and yellow boxes. The x, y, and z axes of each element are shown only when the user clicks on an element in order to change its attributes; otherwise, they remain invisible to prevent cluttering the work area (see section 3.2).

Figure 6: Dart to X-29 morph.


(a) Lion volume from scan−converted polygon mesh.


(b) Leopard−horse volume from scan−converted polygon mesh.

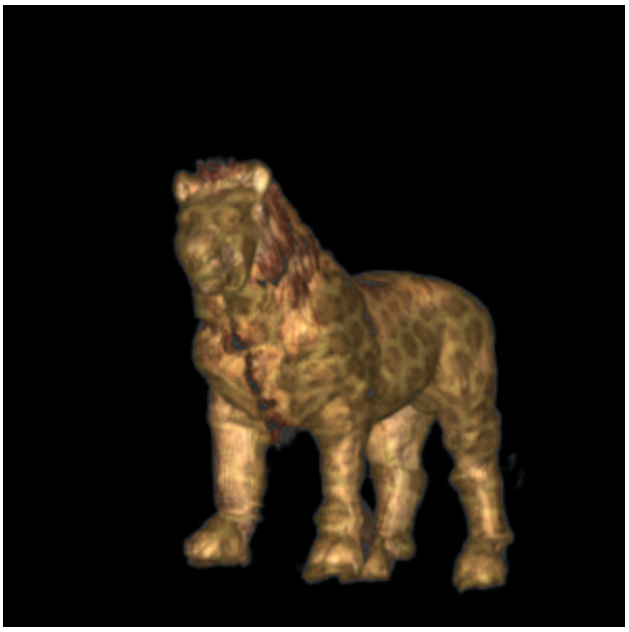

(c) Volume morph halfway between lion and leopard−horse.

Figure 7: Lion to leopard-horse morph.