Texture Transfer During Shape Transformation

HUONG QUYNH DINH
Stevens Institute of Technology
and
ANTHONY YEZZI and GREG TURK
Georgia Institute of Technology

Mappings between surfaces have a variety of uses, including texture transfer, multi-way morphing, and surface analysis. Given a 4D implicit function that defines a morph between two implicit surfaces, this article presents a method of calculating a mapping between the two surfaces. We create such a mapping by solving two PDEs over a tetrahedralized hypersurface that connects the two surfaces in 4D. Solving the first PDE yields a vector field that indicates how points on one surface flow to the other. Solving the second PDE propagates position labels along this vector field so that the second surface is tagged with a unique position on the first surface. One strength of this method is that it produces correspondences between surfaces even when they have different topologies. Even if the surfaces split apart or holes appear, the method still produces a mapping entirely automatically. We demonstrate the use of this approach to transfer texture between two surfaces that may have differing topologies.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Geometric algorithms, languages, and systems*

General Terms: Algorithms

Additional Key Words and Phrases: Morphing, texture mapping, surface correspondence, implicit surfaces

1. INTRODUCTION

There are many tasks in computer graphics that require a mapping between two surfaces A and B. Perhaps the most well-known example is texture mapping, where one of the surfaces is a rectangular patch in 2D and the other surface is an object in 3D that requires a texture. Morphing between surfaces (sometimes called shape transformation) is a second common use of mappings between surfaces. Yet another application of mappings is to analyze a collection of surfaces to look for patterns of common features. To date, graphics researchers have concentrated on producing mappings for *parametric* surfaces. *Implicit* surfaces have received less attention. This article presents a method of creating a mapping between two implicit surfaces for which an implicit morph has already been defined.

Implicit representations are especially popular for shape morphing because they gracefully handle changes in topology. Many algorithms for generating implicit morphs have been published [Hughes

This work was funded in part by NSF CARGO grant DMS-0138420.
Authors' addresses: H. Q. Dinh, Multimedia, Vision, and Visualization, Department of Computer Science, Stevens Institute of Technology, Castle Point on Hudson, Hoboken, NJ 07030; email: quynh@cs.stevens-tech.edu; A. Yezzi, Van Leer Electrical Engineering Building, Georgia Institute of Technology, 777 Atlantic Drive, Atlanta, GA 30332-0250; email: ayezzi@ece.gatech.edu; G. Turk, College of Computing, Georgia Institute of Technology, 801 Atlantic Drive, Atlanta, GA 30332-0280; email: turk@cc.gatech.edu.

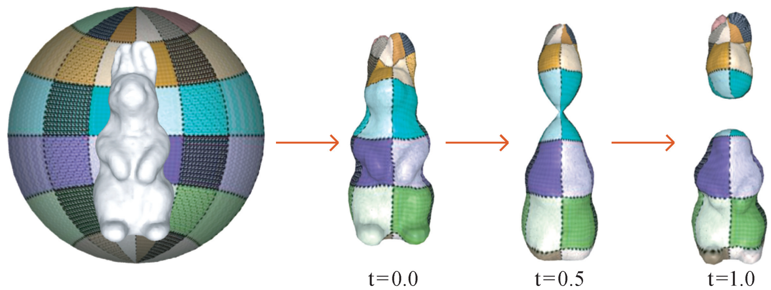

Fig. 1. Transferring a texture during shape transformation—pulling apart a patchwork bunny (model from Cyberware). We solve PDEs over a 4D mesh to obtain a mapping from the sphere to the bunny. Once the source bunny shape has texture coordinates, these coordinates are transfered directly to consecutive shapes in the morph using the same PDEs.

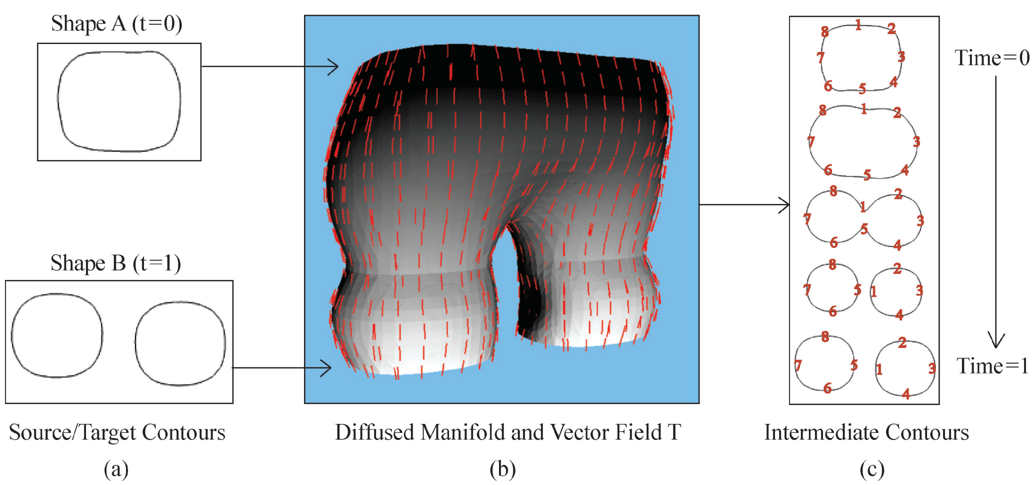

Fig. 2. (a) 2D example of mapping from one closed contour (shape A) to two disjoint contours (shape B). (b) 2D Manifold extracted from the implicit morph spanning 2D space and time. The surface is colored according to the value of the scalar field s. Red lines depict the vector field $\mathbf{T}$ defined by the gradient of s. (c) Propagation of labels along the vector field generates an explicit mapping shown as labeled points on the contours.

1992; Payne and Toga 1992; He et al. 1994; Wyvill 1997; Cohen-Or et al. 1998; Turk and O'Brien 1999; Breen and Whitaker 2001]. Given a shape transformation using any implicit approach, our technique creates a mapping between shapes A and B. This mapping can be used in many ways, including texture transfer, multi-way morphing, and surface analysis. In this article, we show how the mapping can be used for texture transfer. For multi-way morphing, mappings are generated between all of the input shapes and used in affine combinations of the shapes. With a mapping between two surfaces in hand, we can also perform surface analysis by comparing how the surfaces stretch or compress relative to each other. Later in Section 6.3, we perform such an analysis to measure how texture is stretched in our texture transfer application. Examples of many of these applications can be found in Praun et al. [2001] using mesh parameterizations.

Figures 1 and 3 show the strengths of our algorithm. In Figure 1, we transfer texture during a morph directly from one intermediate shape to the next without particle tracing. In previous particle tracing approaches, particles had to be propagated from a parameterized domain (e.g. a sphere) to each intermediate shape in the morph separately for texturing [Zonenschein et al. 1997, 1998a; Tigges

Fig. 3. Texturing a shape with complex topology. Texture was transferred from a sphere (top left) to the Happy Buddha model (middle). Bottom left: Surface colored by texture coordinates. Blue indicates topological seams. Right: Close-up of topological seams.

and Wyvil 1999]. In Figure 3, we show that our approach can transfer texture to surfaces of complex topology.

Our map creation technique relies on the solution of two partial differential equations (PDEs) on an n-manifold in $n + 1$ dimensions, where $n = 3$ for morphing between 3D surfaces. Because the lower-dimensional case is easier to follow, we will describe the method for producing a mapping between two 2D contours (see Figure 2). The first step is to create a mesh of triangles that forms a surface (2D manifold) connecting the two contours A and B in 2D space plus time ($n + 1$ dimensions, where $n = 2$ for contours). Such a triangle mesh is created from the implicit morph using standard iso-surface extraction techniques such as Marching Cubes [Lorenson and Cline 1987]. Next, we solve Laplace's equation (related to heat diffusion) that assigns a scalar value s to every vertex of the mesh. Vertices at contour A have value $s = 0$, vertices of B have $s = 1$, and vertices between A and B take on intermediate values. The gradient of this scalar field gives us a vector field that flows from A to B. Finally, we solve a second PDE (a transport equation) that propagates position labels from A to B along the vector field. These position labels (which we call $\mathbf{L}$) define our mapping $F : A \to B$. For creating a mapping between surfaces, all of these same operations are performed on a collection of tetrahedra in 4D rather than on triangles in 3D. The steps in our approach are detailed in Section 4.

After covering related work, we describe the steps needed to produce a mapping: tetrahedralization, solving the first PDE to get a vector field, and then propagating the labels with the second PDE. We then demonstrate the transfer of texture using our mappings, and in the process describe how we treat topological seams and texture seams.

2. RELATED WORK

Related work in the graphics literature includes surface correspondence methods for triangle meshes, shape morphing methods using implicit functions, and texture mapping. In addition to reviewing these areas, in Section 4 we review a PDE method for measuring annular tissue thickness, which motivates the mathematical approach in this article.

Algorithms for shape morphing can be grouped into two categories. The first are methods that generate an explicit parametric correspondence between meshes. The shape is transformed from one mesh to the other by moving points along the parameterized paths of correspondence. The primary disadvantages of these methods is that they cannot automatically handle a change of topology, and they may require significant user input. The second category of morphing algorithms includes methods that generate an implicit function that spans space and time. Intermediate shapes in the transformation are obtained by extracting all points where the implicit function evaluates to zero at the given time slice. Unlike the mesh correspondence methods, changes in topology are given for free by these methods in that the changes need not be explicitly defined. Unfortunately, a mapping between points of the different shapes in the transformation is not given. Lazarus and Verroust [1998] is a good survey of morphing algorithms.

2.1 Correspondences for Meshes

There are several published methods for creating mappings between triangle meshes, mostly with the goal of creating shape transformations. An early article on this topic is Kent et al. [1992] in which the authors map convex and star-shaped objects onto a unit sphere. Correspondence between two shapes is then defined through the intermediate shape of the sphere. Kaul and Rossignac construct a parameterized interpolating polyhedron (PIP) representing the evolving shape such that faces of the evolving shape have consistent orientation, and vertices move on a straight line from source to target [Kaul and Rossignac 1991]. Kanai et al. [1997] use harmonic maps to generate a transformation between two shapes that are homeomorphic to a sphere. A harmonic mapping embeds a 3D object onto a 2D disk by positioning the single loop boundary of the object onto the boundary of a disk [Eells and Sampson 1988]. Additional user control and transformations between non-spherical homeomorphic shapes are handled in Kanai et al. [2000]. Gregory et al. [1998] produce morphs by having the user select corresponding vertices on two polyhedra of the same genus. Their system automatically decomposes the polyhedra into patches based on these feature points. Lee et al. [1998] developed MAPS (Multiresolution Adaptive Parameterization of Surfaces) to establish a correspondence between a detailed mesh and a simplified version of the same mesh. Lee et al. [1999], establish a correspondence between the base domains of source and target shapes for shape morphing. Praun et al. [2001] use a restricted brush fire algorithm to construct correspondences between multiple surfaces based on feature points identified by the user.

The parametric mesh correspondence methods described above are constrained to transformations between meshes of a common genus. Takahashi et al. [2001] overcome this restriction by allowing the specification of a *key-frame* mesh in space and time where the topological transition takes place. The key-frame mesh binds two surfaces of differing topologies through a pair of faces, each of which is homeomorphic to one of the two surfaces. A correspondence is established between each source mesh and the key-frame mesh using the method of Ohbuchi et al. [2001].

Most of the above mesh-based methods cannot handle changes in topology in a shape transformation, and user input is required in order to help find a mapping between surfaces. In contrast to this, none of the implicit morphing methods require user input, although some allow user intervention to improve the results. The method we present can transfer texture for any morphing method including parametric methods, but is especially useful for implicit morphs that lack a parameterization.

2.2 Implicit Shape Transformation

A wide variety of approaches to performing shape transformation using implicit functions have been published in the graphics literature. These methods require little user-input and are able to handle changes in topology without user intervention.

Brian Wyvill [1997] describes shape morphing as interpolation between corresponding pairs of two blobby implicit models. Hughes transforms discrete implicit representations of two shapes into the frequency domain and performs a “scheduled” interpolation between the shapes, fading out the high frequencies first [Hughes 1992]. Payne and Toga [1992] use linear interpolation between signed distance transforms of the two shapes to produce smooth morphs. At any point in time, the intermediate shape is the zero level-set of the interpolated distance function. Cohen-Or et al. [1998] add user control to this process by allowing spatial deformations that better align the shapes, and the results are superior to those created using non-deformed shapes.

In He et al. [1994], a wavelet transform is used to perform shape morphing. Volumetric data sets are decomposed into a set of frequency bands that are interpolated separately and recombined to form the intermediate models. Before the wavelet transform is performed, the authors first establish a correspondence between voxels of the discretized implicit surface representations by distributing the components of the first object as evenly as possible onto the second object. Although their approach generates an explicit mapping between the source and target shapes, the correspondences depend only on one dimensional length, not area or curvature, even for surfaces, and it often assigns multiple voxels of one model to a single voxel of the other model.

Breen and Whitaker [2001] present a level-set approach for producing shape transformations. They move their level-set according to the value of the signed distance transform of the target shape. Regions of the source shape that are outside of the target shape contract, while regions inside expand. Surface colors are interpolated for intermediate shapes using color volumes [Breen and Mauch 1999; Breen et al. 2000]. Although these color volumes do map a surface point of an intermediate shape with surface points of the source and target shapes, the associations are based purely on closest points—a surface point on the target shape is mapped to the closest point on the source shape. Such a mapping is not unique and may not be bijective.

Turk and O'Brien [1999] use a 4D generalization of thin-plate splines to perform shape transformation. They represent the implicit function for the morph as a weighted sum of radial basis functions, where the weights are found by solving a matrix equation. Thin-plate deformation is provided as a control mechanism for users.

All of the above implicit morphing methods allow topological changes to occur during the morphs, but do not construct a correspondence between the surfaces of the source and target shapes (He et al. [1994] create voxel, not surface, correspondences). In our work, we use either of two different shape transformation algorithms—distance field interpolation [Payne and Toga 1992] and variational implicit transformations [Turk and O'Brien 1999]—to generate an implicit morph that may include topological changes. We then construct an explicit mapping between the shapes in the implicit morph.

2.3 Texture Mapping

Because we will demonstrate our technique on the transfer of texture between surfaces, texture mapping is a closely related topic. In particular, we review work that has been published in the area of texture mapping implicit surfaces. We also review different criteria and metrics used in texture mapping.

2.3.1 *Texture Mapping Implicit Surfaces.* Pedersen [1995] presents an interactive method to texture implicit surfaces. He divides an implicit surface into patches using smooth geodesic curves constructed between user specified points on the surface. Iso-parametric curves are computed automatically for each patch in a two step process with similarities to our own approach. The first step is the creation of a vector field across the patch representing the orthogonal directions of a 2D texture. The second step aligns the iso-parametric curves to the vector field computed in the first step. Though his motivations for this two step process are similar to ours, the details of the two steps (energy that is minimized, dimension, and discretization) are completely different from our method.

Zonenschein et al. [1997, 1998a] texture map an implicit surface by generating an explicit mapping between the implicit surface and a support surface that can be described both implicitly and parametrically, such as a sphere or cylinder. They create the mapping using a particle system. Particles are initialized on the implicit surface and move toward the support surface under the influence of a force defined by the gradients of the implicit functions (both the support surface and the surface being textured). In Zonenschein et al. [1998b], they extend this method to composite implicit shapes that can deform using multiple support surfaces. Tigges and Wyvill [1999] increase the flexibility of the algorithm by using skeletons as the support structure instead of a surface. The above approach is Lagrangian, and hence is susceptible to the disadvantages of Lagrangian methods, such as sensitivity to the time step governing the particles' movement and the need to maintain a uniform sampling of particles. Instead, we present a completely automatic, Eulerian approach that operates on a mesh, and is thus independent of particles.

In Benson and Davis [2002] and Debry et al. [2002], the authors construct an octree that spans the surface of the model and is indexed by the three dimensional coordinates of the model. The octree's leaf nodes contain color samples forming the texture, and the model's texture coordinates are the same as its vertex coordinates. The texture stored in the octree is generated by 3D painting programs, requiring significant user input.

2.3.2 *Minimizing Distortions in Texture Mapping.* An important metric used to measure the quality of a texture map is the amount by which the 2D domain is distorted over the textured surface. Distortions include the stretching, compressing, or skewing of the 2D image. Pioneering work in non-distorted texture mapping includes that of Bennis et al. [1991] in surface flattening and Maillot et al. [1993] in interactive texture mapping. Another frequently referenced work is that of Floater, in which he develops a method to compute the parametric coordinates for a surface point as the convex combination of neighboring points [Floater 1997]. This produces a barycentric mapping of the interior surface point such that the parameterization mimics chord length between the surface points, thereby minimizing distortion. Levy and Mallet [1998] obtain texture coordinates for a triangulated mesh by iteratively minimizing a distortion criteria that includes perpendicularity and constant spacing of the parametric curves defining the texture mapping. Haker et al. [2000] construct a *conformal* mapping that is angle preserving and changes distance and areas only by a scaling factor. They obtain the mapping through solving a PDE formulated as a system of linear equations. Additional work in texture mapping includes constrained texture mapping [Levy 2001], creating texture atlases using conformal mapping [Levy et al. 2002], and parameterization of polygonal models [Sheffer and de Sturler 2001] that minimize distortion.

In our work, we use a geometric texture stretch metric introduced by Sander et al. [2001]. This metric measures the largest and smallest length obtained when mapping a unit length vector from the texture domain to the surface. They define two stretch norms—the worst-case and root-mean-square stretches. The worst-case stretch is the maximum stretch over the entire surface, while the root-mean-square stretch combines both the largest and smallest stretch lengths into one value. Sander et al. minimize both metrics iteratively by changing the texture coordinate of each vertex while all others are held fixed. In Section 9, we use stretch norms to verify the quality of the texture mapping generated from our explicit correspondences. A useful survey of stretch metrics can be found in Zhang et al. [2003] where they combine Sander's area-preserving metric with conformal preservation using the Green-Lagrange tensor.

3. TERMINOLOGY

In this article we will use the term *mapping* as a shorthand for a diffeomorphism between surfaces. A diffeomorphism is a mapping function F from one surface A to another surface B that is bijective, differentiable and has a differentiable inverse. In a diffeomorphism, the function F defines a unique point $F(\mathbf{p})$ on B to each point $\mathbf{p}$ on A, and it also gives a unique point $F^{-1}(\mathbf{q})$ on A for any point $\mathbf{q}$ on B. That is, a point on either of the surfaces is put into correspondence with a unique point on the other surface. The differentiability of the mapping ensures that it is smooth from one point to the next on both surfaces.

When two surfaces have different topologies it is impossible to create a diffeomorphism. For the examples in this article, it is possible to remove a set of measure zero (infinitesimally small) from each surface and define a piecewise diffeomorphism between the remaining portions of the surface. We conjecture that this will always be possible, but we do not have a proof. As an example, consider a morph from a sphere to a torus that is made by pinching the north and south poles together. If we remove the north and south poles of the sphere and an infinitely thin band around the torus hole, the two remaining surfaces are topologically equivalent. A diffeomorphism can be created for these two modified surfaces. This is the form of the solution that our method produces in the case of surfaces with different topologies, although we do not actually remove points from the surfaces.

4. EULERIAN CORRESPONDENCE TRAJECTORIES

Our method for creating a mapping was inspired by a mathematical approach for computing the thickness of regions with two distinct boundaries using a pair of linear partial differential equations. This approach was first used for computing tissue thickness on rectangular grids [Yezzi and Prince 2001]. In this article we generalize the method to polyhedral grids in order to compute correspondences between boundaries of triangular surfaces and tetrahedral volumes (contours and surfaces). Specifically, we use a simple linear PDE to generate trajectories from one boundary to another, and a second linear PDE to transport information along these trajectories using only the structure of the original grid (i.e., without "particle tracing" along the trajectories). In our algorithm, both PDEs are applied discretely to an n-manifold in $n+1$ dimensions, rather than to the Cartesian grid as in Yezzi and Prince [2001]. For creating correspondences between 2D contours morphing in time, we apply the PDEs to vertices of a triangular mesh extracted from the implicit transformation defined in 2D space and time. For morphing 3D shapes, we apply the PDEs to a tetrahedral mesh (3D manifold) spanning 3D space and time.

The first step in our approach is to build a scalar field s on the grid whose gradient forms a vector field $\mathbf{T}$ defining flow lines that run bijectively between the source and target boundaries of the grid. This may be done by solving Laplace's equation ($\Delta s = 0$) with boundary conditions of 0 and 1 on the source and target boundaries respectively. The flow lines run bijectively between the source and target because the solution of the Laplacian takes on maximum and minimum values only at the boundaries. Hence, the flow lines will terminate only at the boundaries, which, in our case, are the source and target shapes. The next step involves solving a second PDE, which describes the differential structure of some other function along these trajectories. In Yezzi and Prince [2001], this function was the arclength of the trajectories (used to define the thickness between the two boundaries).

To create surface mappings we use the same two step PDE-based procedure, but make two changes. First, we apply the scheme on triangulated and tetrahedral meshes of non-flat domains and therefore substitute the Laplacian operator with the more general Laplace-Beltrami operator in the first PDE [Pinkall and Polthier 1993]. Second, we are not interested in the lengths of the correspondence trajectories but are instead interested in their endpoints. These give us the pointwise correspondences between the two boundaries. As such, the second transport PDE dictates that the derivative of the correspondence vector field (which we call *labels* $\mathbf{L}$) along the gradient of s is zero:

$$\nabla\mathbf{L} \cdot \mathbf{T} = 0 \quad \text{where} \quad \mathbf{T} = \frac{\nabla s}{\|\nabla s\|}. \tag{1}$$

Note that both the first and second PDEs can be solved via computations on the n-manifold (triangular or tetrahedral mesh). In other words, the scalar field s and vector fields $\mathbf{T}$ and $\mathbf{L}$ are computed at each vertex in the mesh using only information from neighboring vertices. This Eulerian framework avoids the complexities of tracing particles along the trajectories embedded intrinsically within the solution of the first PDE. Figure 2 diagrams the steps involved. After discussing mesh creation, we show how these PDEs can be solved on arbitrary meshes.

5. CREATING THE HYPERSURFACE

Because we are solving the problem of transferring textures during an implicit shape transformation, we first review the implicit morphing algorithms we use. For the case of morphing between 3D surfaces, the implicit transformation is described as a scalar function over $\mathbf{R}^4$. In order to discretely solve the two PDEs that will create the mapping, we need to extract an explicit representation—a mesh—of the morph. In this section we briefly describe how we generate the implicit morph and the tetrahedral mesh.

Our approach to creating mappings between surfaces can use any implicit method that creates a morph, including Wyvill [1997], Hughes [1992], Payne and Toga [1992], He et al. [1994], Cohen-Or et al. [1998], Turk and O'Brien [1999], and Breen and Whitaker [2001]. To show that our method is independent of the implicit transformation scheme, we use two different morphing algorithms—distance field interpolation [Payne and Toga 1992] and the variational implicit method [Turk and O'Brien 1999]. We give a sketch of each approach below. We refer the interested reader to the original papers for details.

Payne and Toga [1992] construct a shape transformation between two shapes by cross-dissolving the distance fields of the two shapes, resulting in a linear interpolation described by the following equation:

$$d(\mathbf{x}) = (1-t)d_A(\mathbf{x}) + t d_B(\mathbf{x}). \tag{2}$$

In the above equation, $d(\mathbf{x})$ is the interpolated distance, t is time, d_A is the distance transform for the source shape A, and d_B is the distance transform for the target shape B. At any particular time slice, d is an implicit function describing the shape. At $t = 0$, d is completely defined by the source A, and at $t = 1$, d is completely defined by the target B. Intermediate shapes are extracted where d evaluates to zero.

The variational approach to shape transformation creates an implicit function in 4D that is a weighted sum of thin-plate radial basis functions:

$$G(\mathbf{x}) = P(\mathbf{x}) + \sum_{i=1}^{n} w_i \phi(|\mathbf{x} - \mathbf{c}_i|). \tag{3}$$

In the above equation, $G(\mathbf{x})$ is an implicit function that evaluates to zero on the surface, negatively outside, and positively inside; ϕ is a radially symmetric basis function; n is the number of the basis; $\mathbf{c}_i$ are the centers of the basis functions; and w_i are the weights for the basis functions. The term $P(\mathbf{x})$ is a linear polynomial. The basis functions are created in two places: on the surface of the shape and near the surface in a direction given by the surface normal. Because we are defining a 4D function over (x, y, z, t), all of the basis functions for the source shape A are placed in the sub-space $t = 0$, and the basis functions for shape B are placed at $t = 1$. The weights of the basis functions are determined by solving a matrix equation. Once the weights have been found, we have a 4D implicit function (Equation 3) that describes the entire shape transformation. In Section 9.1, we discuss how the different morphing algorithms affect the mapping generated between the shapes.

In order to solve PDEs on the 4D implicit function that describes the shape transformation, we first extract a tetrahedral mesh spanning 3D space and time. We use a variant of the Marching Cubes method that has been generalized to higher dimensions [Bhaniramka et al. 2000]. In 3D, the Marching Cubes method extracts an iso-surface from an implicit function by dividing the space spanned by the surface into cubes. The implicit function is evaluated at each vertex, and the configuration of positive and negative values indexes into a case table that provides the triangulation within the cube. In 4D, instead of cubes, the space is divided into hypercubes that span (x, y, z, t), resulting in 65,536 possible combinations. Bhaniramka et al. present a method to automatically enumerate all the possible cases for n dimensions. Once the case table is constructed, iso-surfacing in higher dimensions is conducted in the same manner as Marching Cubes in 3D. With a tetrahedral mesh in hand, we can now turn to the task of solving the PDEs over the mesh.

6. VECTOR FIELD CREATION

6.1 First PDE: Constructing a Mapping by Solving Laplace's Equation

The purpose for solving the first PDE is to obtain a vector field, **T**, that indicates how points on the first surface flow to the second surface. We define **T** as the gradient of a scalar field s that we initialize as follows: $s = 0$ at the source shape A, $s = 1$ at the target shape B, and s is interpolated in between A and B. We then solve Laplace's equation ($\Delta s = 0$) to smooth out the values. In Figure 2b, the surface spanning 2D space and time is colored by the value of s (black at $t = 0.0$ and white at $t = 1.0$). By solving Laplace's equation, we are performing diffusion, which minimizes the integral of the scalar field's gradient. Following the characteristic curves of the diffused scalar field generates correspondence trajectories that are uniformly spread throughout the transformation space. Trajectories flow over protrusions (or into concavities) rather than around them, and never intersect. Solving the first PDE produces a unique harmonic function s over the space that interpolates between 0 along the source shape and 1 along the target shape.

The value of Δs for a function s on a manifold depends upon two factors—the function s and the local coordinates used to discretize the manifold. To make the value of Δs depend exclusively upon the geometry of the manifold rather than the choice of local coordinates, we use the generalization of the Laplacian from flat spaces to manifolds known as the Laplace-Beltrami operator. Evaluation of the Laplace-Beltrami operator yields a result equivalent to the evaluation of the standard Laplacian in the special case that the local coordinates of the manifold are chosen geometrically to yield mutually orthogonal curves each with unit speed (i.e., parameterized by arclength). By solving Laplace's equation using the Laplace-Beltrami operator, we are performing *geometric* heat diffusion on our tetrahedral mesh. We note that Bertalmio et al. [2001] and Memoli et al. [2004] have devised a method for evaluating this operator over a thin slice of the Cartesian grid that surrounds an implicit surface, and this is a reasonable alternative to our own approach. The Laplace-Beltrami operator for triangulated meshes and for 3-manifolds in n dimensions is given below [Pinkall and Polthier 1993; Meyer et al. 2002].

$$\int_{\Omega} \mathbf{K}(s) d\,\Omega = \frac{1}{2} \sum_{j \in N(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(s_i - s_j). \tag{4}$$

In the above equation, the region of interest Ω is a surface for a triangulated mesh and a volume for a tetrahedral mesh. $N(i)$ is the 1-ring neighboring vertices of i. For a triangulated mesh, α_{ij} and β_{ij} are the two angles opposite to the edge (i, j). For a tetrahedral mesh, α_{ij} and β_{ij} are the dihedral angles opposite to edge (i, j) (note that there are generally more than two dihedral angles for any edge in a tetrahedral mesh). We apply the Laplace-Beltrami operator to the mesh until convergence of the scalar field—the total change in the scalar values is below a desired threshold. The runtime is on the order of

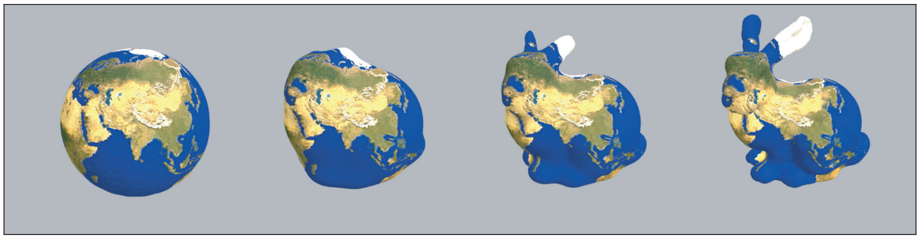

Fig. 4. Sphere to Stanford Bunny transformation—a morph between two shapes with a common topology. A texture of the Earth is transferred from the sphere to the Stanford Bunny using the explicit mapping between implicit surfaces generated by our approach.

a few seconds to several minutes depending on the mesh resolution. Alternately, we can apply implicit integration and directly solve for s using conjugate gradient as prescribed in Desbrun et al. [1999]. As expected, fewer iterations are required to generate similar results, but each iteration takes longer (see Section 9 for runtimes). The final tangent vector field, $\mathbf{T}$, is obtained by taking the gradient of s, which we describe in the next section.

6.2 Tangent Field Calculation

Once we have the scalar field s, we construct the tangent vector field $\mathbf{T}$ by calculating the gradient, $\nabla s = \partial s/\partial \mathbf{x}$ where $\mathbf{x} = (x, y, z, t)$, throughout the mesh. For each vertex i, the tangent vector $\mathbf{T}_i$ is calculated by weighted averaging (based on tetrahedra volume) of the gradients of s in the tetrahedra that are in the 1-ring neighborhood of the vertex. Within each tetrahedron, we solve the following system to obtain the gradient of s, where (x_i, y_i, z_i, t_i) are the 4D coordinates of vertex i in the tetrahedron, and s_i is the vertex's scalar value.

$$\begin{bmatrix} x_1 & y_1 & z_1 & t_1 \\ x_2 & y_2 & z_2 & t_2 \\ x_3 & y_3 & z_3 & t_3 \\ x_4 & y_4 & z_4 & t_4 \end{bmatrix} \begin{bmatrix} \partial s/\partial x \\ \partial s/\partial y \\ \partial s/\partial z \\ \partial s/\partial t \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix}. \tag{5}$$

6.3 Discussion

Why solve a PDE in order to create our vector field? As described in Section 6.1, the initial scalar field s is such that $s = 0$ at the source shape A, $s = 1$ at the target shape B, and is linearly interpolated in between according to the t coordinate. Without diffusion, the gradient of this scalar field is still a vector field that flows from shape A to shape B. Propagating a particle along this field in the 4D transformation space is analogous to moving the particle in the direction of maximum change in time. Unfortunately, such a simple field may not uniformly distribute the particles as they are propagated in time. Instead, trajectories describing corresponding points can spread too far apart or compress together too closely, as shown in Figure 5. Pedersen [1995] also describes this problem to motivate his two step approach for generating iso-parametric curves within a patch. The method we described in Section 6.1 overcomes the problem by applying geometric heat diffusion to s, which smooths out the scalar field.

One way to visualize the spread or compression of trajectories is to measure how a texture that has been transferred from A to B is stretched or compressed. In Sander et al. [2001], the authors develop a texture stretch metric to compute the largest and smallest lengths to which a unit-length vector in the texture domain gets mapped to the surface. The worst-case norm is the maximum texture stretch over the surface. Stretch values below 1.0 indicate compression of the texture. When there is no distortion

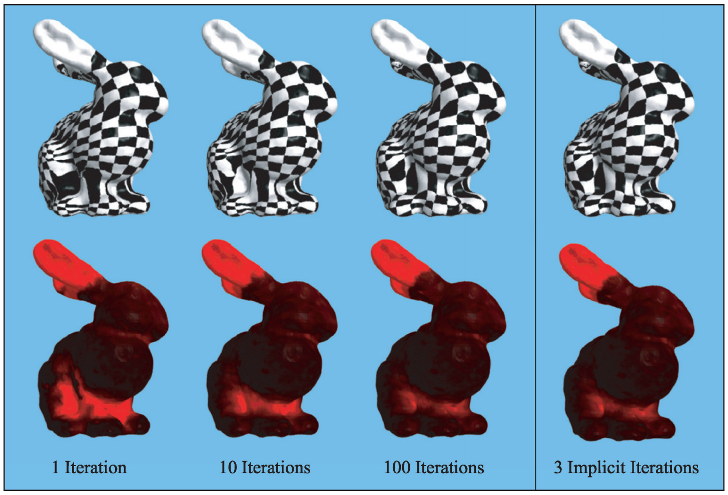

Fig. 5. Above: Checkerboard pattern is mapped onto the Bunny using the texture coordinates transferred from the sphere. Below: The Bunny is shaded according to the amount of texture stretch. Red indicates high stretch or compression. Black indicates minimal stretching and compression. Left to right: Results after 1, 10, and 100 iterations of geometric heat diffusion, and after 3 iterations of diffusion using implicit integration.

Table I. Measure of Stretch in Sphere to Stanford Bunny Morph

Diffusion Iterations	Convergence	Time (secs.)	Min (Compression)	Max (Stretch)	Mean	Variance
1	0.010904	9	0.012069	99.790815	1.787130	7.618893
10	0.001187	9	0.139142	51.168358	1.613149	2.476958
100	0.000204	16	0.162943	19.135322	1.507618	1.112487
3 Implicit	0.000037	16	0.151568	26.397383	1.549555	1.455317

of the texture, the max and min stretch lengths are 1.0. In Figure 5, we have applied this metric to the sphere to Stanford Bunny transformation (shown in Figure 4) while varying the number of iterations of geometric heat diffusion. We have also compared the iterative (Gauss-Siedel) method to implicit integration. Fewer iterations of implicit integration are required to reach convergence, but at a cost of solving a linear system, requiring more computation per iteration. No distortion is assumed for the sphere, so the amount of texture stretch on the Bunny is relative to the sphere. The results show that the areas of texture stretch are reduced as more iterations of diffusion are applied. Only 3 implicit integration steps are required to obtain results comparable to 100 forward Euler steps. We have used a regular checkerboard pattern to show the distortion of the texture. Table I shows that, quantitatively, measures of stretch across the surface improve with diffusion—the max, min, and mean stretch lengths improve (progress towards 1.0), and the variance in stretch is reduced.

Note that heat diffusion actually minimizes a conformal metric, but we have used the stretch metric of Sander et al. because we want to show that our trajectories flow over protrusions (or into concavities) rather than around them. The stretch metric can reveal areas where the trajectories are spread too far apart or are compressed. A conformal metric would not show how trajectories progressively move into concavities or over protrusions as the diffusion converges. An alternative metric to use for visualizing our results is one that combines both stretch and conformal metrics such as found in Zhang et al. [2003].

7. SECOND PDE: LABEL PROPAGATION

Recall that the tangent vector field, $\mathbf{T}$, indicates how points on the source surface flow to the target surface. We determine a complete correspondence between the source and target shapes by propagating position labels, $\mathbf{L}$, from the source shape to the target shape along characteristic curves defined by the vector field (shown as red lines in Figure 2b). The label $\mathbf{L}$ is simply a position on the first shape that we wish to pass along to the second shape. For 2D contour mapping, the labels are 2D positions $\mathbf{L} = (x, y)$, while for 3D shape morphing, they are 3D positions $\mathbf{L} = (x, y, z)$. The purpose of the second PDE is to "push" these position labels across the mesh from shape A to shape B. Labels are only assigned to vertices $\mathbf{p} = (p_x, p_y, p_z)$ of the first shape A, and for these initially labeled vertices, $\mathbf{L}(\mathbf{p}) = (p_x, p_y, p_z)$. Once the labels have been propagated, each point on the target shape maps to the point on the source shape with the corresponding label.

One method of propagating the labels without solving a PDE is to treat each label as a particle and trace its path as it moves along $\mathbf{T}$, from the source to the target shape. Particle tracing is, however, compute intensive since particles must be moved in small steps to closely follow the characteristic curve. In addition, particles must be created and destroyed in order to maintain a consistent particle sampling. We avoid particle tracing and propagate labels more robustly by solving a second PDE over the mesh:

$$\nabla \mathbf{L} \cdot \mathbf{T} = 0. \tag{6}$$

This equation states that the direction of maximum change of $\mathbf{L}$ should be perpendicular to the vector field $\mathbf{T}$. By solving the above PDE, the resulting values for the label, $\mathbf{L}$, do not change along the characteristic curves described by the vector field, $\mathbf{T}$. In other words, $\mathbf{L}$ is constant along a trajectory. The characteristic curve drawn out by an iso-contour of $\mathbf{L}$ maps a point from the source shape to a point on the target shape.

7.1 Solving the PDE on a Tetrahedral Mesh

When solving the PDE of Equation 6, it is important to use a numerical scheme that respects the fact that information flows exactly along the characteristic curves (i.e., the correspondence trajectories). Central differencing schemes are not appropriate because they use information from all directions around a given vertex rather than exclusively from directions aligned with the characteristic curves flowing through a local neighborhood of the vertex. Instead, we devise an upwind differencing scheme that exploits the fact that the appropriate upwind neighboring vertices can be determined from the tangent field $\mathbf{T}$. Only these vertices and the current vertex should then be used to approximate the gradient of $\mathbf{L}$ when numerically solving Equation 6.

On a flat rectangular grid, the upwind neighbors are easy to determine. The upwind neighbor to a given grid point in the x-direction is the grid point to the left if the x-component of $-\mathbf{T}$ is negative, and to the right otherwise. The upwind neighbors are similarly determined in the other Cartesian directions, and $\nabla\mathbf{L}$ can then be computed using one-sided difference approximations. Based upon this approximation, the transport equation (6) yields a linear equation that can be used to solve for $\mathbf{L}$ at each grid point. This forms the basis of an iterative procedure that is applied at each grid point until convergence.

On a more complicated mesh, such as a triangular or tetrahedral surface, we determine the upwind neighbors of a particular vertex by projecting the vertices of the triangles (or tetrahedra) in the star around that vertex onto the tangent plane where $\mathbf{T}$ resides. Note that the tangent plane is computed by weighted averaging of the normal vector of the triangles (or tetrahedra) in the star of the vertex. The upwind vertices are those of the projected triangle (or tetrahedron) that intersects $-\mathbf{T}$, as shown in Figures 6a and 6b .

After the labels have been propagated using upwinding on the mesh, every vertex of the second object has a label $\mathbf{L}$ that corresponds to a location on the first object. These labels define our mapping from

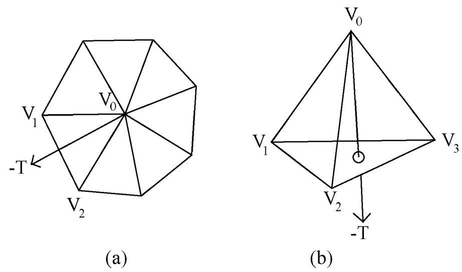

Fig. 6. The upwind direction, specified by $-\mathbf{T}$, intersects one triangle in the star of vertex V_0 in (a) and intersects the tetrahedron in (b). (a) Vertices V_1 and V_2 are the upwind neighbors used in determining the label values of V_0. (b) Vertices V_1, V_2, and V_3 are the upwind neighbors used in determining the label values of V_0. In both cases, the label of V_0 is computed to satisfy Equation 6.

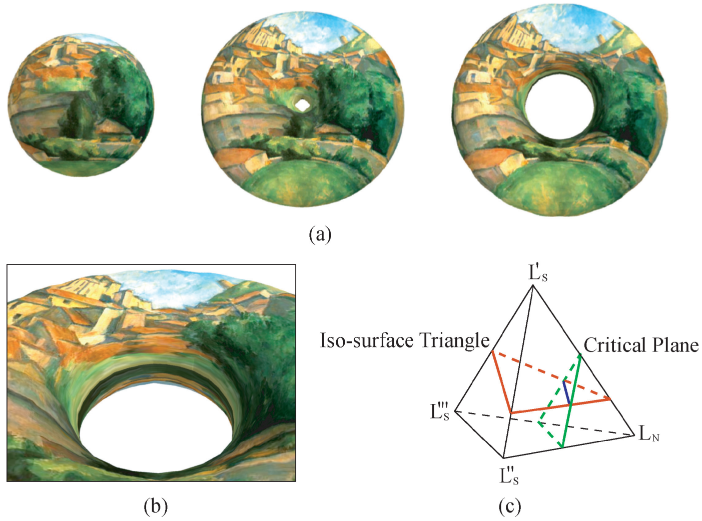

Fig. 7. (a) Sphere to torus transformation—a morph between two shapes of different topologies, involving a hole creation. A texture of Cezanne's *Gardanne* is transferred from the sphere to the torus. (b) Topological seam at the inner ring of the torus. (c) A triangle (red) of the target surface is split by the critical plane (green) embedded in the tetrahedron (blue line shows the split). The tetrahedron spans a topological seam with labels $\mathbf{L}_S$ that correspond to points near one pole and $\mathbf{L}_N$ that corresponds to a point near the other pole.

one surface to another. We may now make use of this mapping to perform operations such as texture transfer.

8. TOPOLOGY CHANGES AND TEXTURE SEAMS

Once a mapping between surfaces has been created, it can be put to a number of uses. In this article we demonstrate its use in transferring texture between surfaces. When performing texture transfer, two kinds of seams must be handled: topological seams and texture seams. In this section we describe how we treat both of these issues.

8.1 Topological Seams

When surfaces with different topologies morph from one to the other, critical points arise at the transition between topologies. For example, when a sphere transforms into a torus, a hole appears during the transformation (see Figure 7a). The point in 4D at which this hole appears is the critical point.

At this junction, labels from opposite poles of the sphere come together and form a *topological seam*. The resulting seam sits at the inner ring of the torus. On one side of the seam, all the points along the inner ring of the torus correspond to one pole of the sphere, while on the other side all the points correspond to the opposite pole. Since we need to interpolate a sparse field of labels, we need to be careful of interpolated labels that could potentially correspond to points between the two poles. Such a label would be erroneous because it is not near the set of labels assigned to the source shape. We prevent incorrect interpolations by explicitly creating a seam in the triangulation corresponding to the topological seam. Labels are then never interpolated across the seam. Figure 7b shows the topological seam in the sphere to torus transformation.

The topological change in the sphere to torus morph is an example of hole creation. There are eight different topology changes that may occur [Stander and Hart 1997], and these may be grouped into two categories. The first category consists of hole creation/destruction and object merging/splitting. The second category consists of object creation/destruction and bubble creation/destruction. Creation and destruction of objects and bubbles do not give rise to a topological seam because whole components appear or disappear. Hole creation and destruction must be treated explicitly, and object merging and splitting may be handled in exactly the same manner. Figure 9 is an example of object splitting.

Finding the exact location of a critical point requires root-finding. Critical points occur where the tangent vector vanishes ($||\mathbf{T}|| = 0$). One option for handling topology changes is to locate all critical points in the tetrahedral mesh and then trace them through the 4D mesh. The path through which a critical point passes, forms a surface that cuts through tetrahedra. All incident tetrahedra need to be split so that a topological seam is explicitly created within the tetrahedralization. For efficiency, we choose not to divide the entire 4D mesh in this manner. Instead, we handle topological seams only at the desired time slice. We explicitly create the topological seam at the target shape B by identifying all triangles on B that span a large distance in label space. The labels assigned to the vertices of a triangle are compared. If they are very far apart (farther than a few marching tetrahedra steps), then the triangle spans a topological seam and must be split. In many cases, the distance between label values in a triangle where a topological seam occurs, is large. For example, in the sphere to torus transformation of Figure 7a, labels at the inner ring of the torus come from the two poles of the sphere. Triangles will either span very small distances in label space (indicating no seam), or very large distances (indicating a seam). There is little ambiguity in the classification, and the distance threshold used to determine if a topological seam exists is easy to choose and is not sensitive to minor differences in the size of tetrahedra or triangles in the mesh. In cases where the labels across a topological seam do not span a large distance, we must resort to the more complex algorithm of locating and tracing critical points described above.

Triangles can be split along the midpoint of edges that span a topological seam, but this would result in a jagged seam. Instead, we split a triangle using the tetrahedron from which it came. Like the triangle, the tetrahedron also spans a topological seam. We assume that the critical plane traced out by the critical point passes through the midpoints of edges that span the topological seam in the incident tetrahedron. The triangle extracted at the desired time slice is split by intersecting it with the critical plane. Figure 7c shows how the critical plane and the extracted triangle interact within a tetrahedron that spans the topological seam. The labels $\mathbf{L}_S$ correspond to distinct points near one pole, while $\mathbf{L}_N$ correspond to a point near the other pole. Our approach produces a smooth seam because the critical plane is coherent from one tetrahedron to the next along the topological seam. Once an explicit topological seam is constructed in the mesh of the target shape, the labels are assigned to correspond to one pole on one side of the seam and to the other pole on the other side of the seam. Thus, labels are never interpolated across seams. Figure 7b shows the smooth topological seam generated by this method. A topologically more complex example is shown in Figure 3.

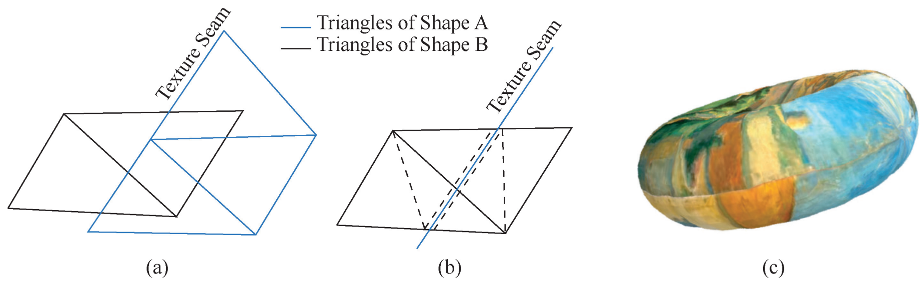

Fig. 8. (a) A texture seam is generated by intersecting triangles of the target shape B with the texture seam of the source shape A. (b) Dashed lines show how triangles of B that are incident to the texture seam are split and retriangulated. (c) The texture seam transferred from the sphere onto the torus.

8.2 Texture Seams

We can transfer texture from the source shape A to the target shape B via texture coordinates. The propagated labels represent an explicit mapping, and we use this mapping to directly transfer texture coordinates from A to B. Note that this method can also be used to transfer other surface properties, such as color, normal vectors, and wavelet coefficients.

Texture seams are present on a textured object when a single texture patch wraps back on itself or when two different textures abut one another. Because we use a sparse distribution of labels (only at the mesh vertices), the texture seam is not explicitly transferred. As a result, the texture coordinates of some triangles on the target surface B could potentially span a texture seam, causing a large portion of the texture to be incorrectly squeezed into the space of one triangle. To prevent this from occurring, we split the triangle along the texture seam so that the seam is explicitly defined on B. We first identify all triangles in B whose texture coordinates span a large distance in texture space or cross a seam defined in the texture (e.g., chart seams in texture atlases). All triangles tagged as spanning a texture seam are then subdivided using seam edges from surface A in texture space, as shown in Figure 8. Note that it is necessary to split the tagged triangles in texture space (s, t), and then map the new vertices generated by the splitting back into spatial coordinates (x, y, z) using barycentric coordinates. Figure 8c shows the correct handling of a texture seam when texture is transferred from one object to another.

9. RESULTS

We have created mappings between a variety of 3D surfaces using our PDE-based approach. In order to illustrate these mappings we show texture being transferred from one model to another. For each example, intermediate models from the morph are created by slicing the 4D tetrahedral mesh with a hyperplane at various t values. Each slice gives the collection of triangles, which we then texture using the mapping information.

Figure 4 shows a morph between two models of a common genus—a bunny and a sphere. Figures 7a and 9 are examples where the genus changes. These morphs were generated using Turk and O'Brien's variational implicit method. In Figures 3 and 10, we have used Payne and Toga's distance field interpolation algorithm. Note that we have used spatial warping using thin-plate radial basis functions as described in Cohen-Or et al. [1998] to constrain the cow to horse morph so that the nose of the cow morphs into that of the horse. All of these mappings were produced entirely automatically—no user intervention was required to define the mappings, even in the case of genus changes. The running times to create the mappings are reasonable. Table II gives the solution times for both PDEs. All of the PDE solutions are on the order of minutes for even our largest models. Note that the cow to horse

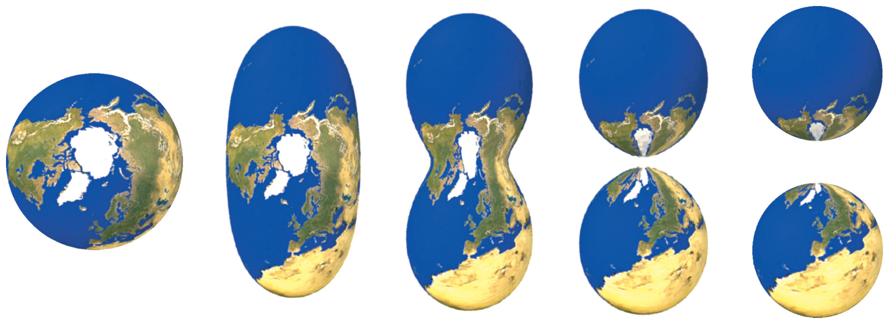

Fig. 9. One sphere splitting into two—a morph involving object splitting. The two poles of the original Earth merge to form monopoles in each of the new spheres.

Table II. Execution Statistics (Iterations, Time, and Convergence) on a PC with a 2 GHz AMD Processor

Morph	Tetrahedra Count	Time Steps	Diffusion (iters.,secs.,conv.)	Implicit Integration (iters.,secs,conv.)	Label Prop. (iters.,secs.)
Sphere to Torus	219998	10	469, 24, 0.00001	2, 11, 0.0	5, 3
Sphere to Bunny	416747	6	535, 46, 0.00001	3, 16, 0.000037	7, 10
Splitting Sphere	858006	10	824, 159, 0.00001	3, 21, 0.000010	6, 13
Sphere to Buddha	1360634	10	93, 61, 0.00012	—	6, 25
Cow to Horse	1692940	10	7, 41, 0.00250	—	10, 50

transformation obtained using distance field interpolation required few iterations (we were unable to use implicit integration for this morph, and the Sphere to Buddha morph due to memory limitations). In the next section, we discuss how the implicit transformation that is used affects the mapping that we generate.

9.1 Comparing Results from Distance Field and Variational Implicit Morphs

Because our mapping algorithm is intended for application to implicit shape transformation methods, it is important to examine the affect that the transformation algorithm has on the results. The tetrahedral mesh that is extracted from the implicit morph differs depending on the shape transformation algorithm that is used. The primary difference between the two algorithms we have used (distance field [Payne and Toga 1992] and variational implicit morphs [Turk and O'Brien 1999]) is that the variational implicit morph (VIM) minimizes thin-plate energy, whereas the distance field morph (DFM) is a linear transformation that minimizes length (or membrane energy). The result is that the trajectory from a source to an target point in the VIM is smoother, while in the DFM, the trajectory is more direct and minimal in distance. We show this difference in Figure 11 by extracting a spatial slice of the torus to sphere transformation. This hyper-slice shows the (y, z, t) surface at $x = 0.0$. At $t = 0$, the contour is a 2D slice of the torus with two small circles forming the torus ring. At $t = 1$, the contour is a 2D slice of the sphere, comprising one large circle. The top-left row is the $x = 0.0$ slice of the DFM. The bottom-left row is the same slice of the VIM. The corresponding texture mapping results are shown in the right panels of Figure 11. Note that we show only the resulting texture mapping at the sphere (target shape). Remarkably, the shape transformation generated by DFM required little diffusion to obtain similar results. This is because the DFM generates more direct trajectories as described above, and so does not require much diffusion to uniformly spread the trajectories over curved regions between the source and target surfaces.

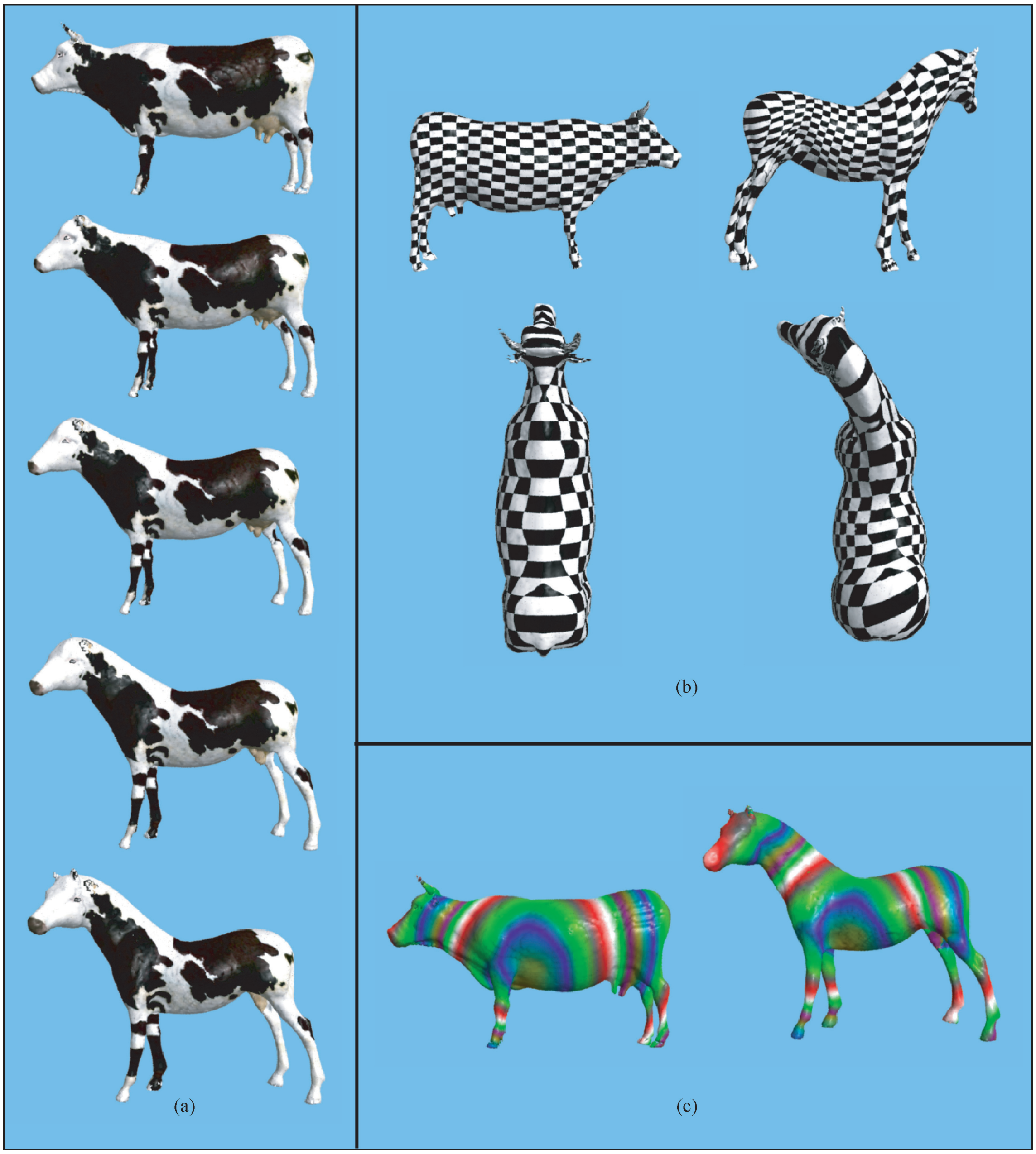

Fig. 10. The transformation from a cow to a horse. (a) The texture from the cow is transferred to the horse model. (b) A checkerboard pattern shows how the mapping smoothly deforms the texture from the cow to the horse. (c) Rings of color identify what parts of the cow are mapped to the horse.

Although DFM is successful in this example, it is sensitive to scale and resolution due to its discrete nature. DFM tends to generate a higher percentage of obtuse dihedral angles with larger obtuse angles which can cause the diffusion to become unstable. In contrast, VIM is less sensitive to the spatial or

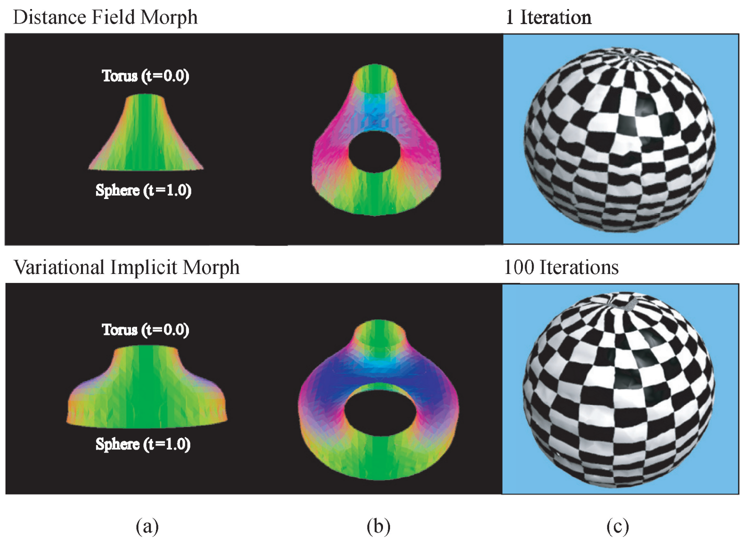

Fig. 11. $x = 0.0$ slice of the torus to sphere transformation. The surface is colored according to surface normal. Side (a) and oblique (b) views of the hyper-slice show that the VIM is smoother but has more surface area than the DFM. (e) Texture mapping results from DFM and VIM using texture coordinates transferred from the torus to the sphere.

temporal resolution used to generate the tetrahedral mesh because VIM is an analytical representation of the morph and generates a smoother morph. We speculate that the difference in the quality of the meshes between DFM and VIM is due to the difference in smoothness and the discrete versus continuous nature of the transformation algorithms. The key advantage of DFM over VIM is its simplicity. We are able to create complex transformations using DFM in Figures 3 and 10.

10. FUTURE WORK

We have presented an automated method to create a smooth and uniform mapping from one surface to another in an implicit morph. Although the method is largely successful, issues remain that would improve the stability and practicality of the method. These include the quality of the tetrahedral mesh and the ability to control the resulting explicit mappings.

10.1 Mesh Quality

Obtuse dihedral angles can introduce instability into the diffusion process. The reason for this is evident in Equation 6.1 for the Laplace-Beltrami operator. The updated value s_i at vertex i is a convex combination of neighboring values only when the dihedral angles are acute. Obtuse dihedral angles generate negative weights that introduce instability into the diffusion process. The need for acute triangulations (in our case, tetrahedralizations) is well documented in the literature on solving PDEs on triangulations [Pinkall and Polthier 1993; Kimmel and Sethian 1998; Meyer et al. 2002]. Although there are algorithms for acute triangulation of surfaces, we have found no algorithm for partitioning an arbitrary polyhedral domain into acute tetrahedra. Only recently has there been a method for acute refinement of arbitrary tetrahedral meshes (note that acute refinement differs from algorithms for acute tiling of 3D space [Üngör 2001]). In Plaza and Rivara [2003], the authors describe a mesh refinement algorithm based on the partitioning of each tetrahedron into 8 tetrahedra with the first bisection performed on the longest edge. A key property of their method is that for each obtuse face of a tetrahedron, all new faces that are created from partitioning are of better quality than the parent face. This property implies that

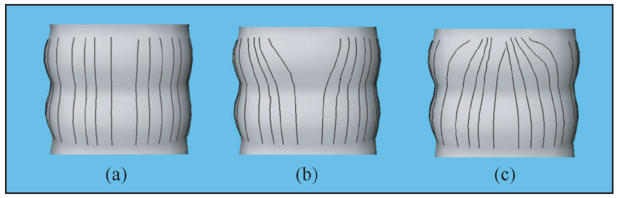

Fig. 12. 2D example of controlling the mapping from a circle at the bottom to another circle at the top (note that another circular constraint was defined in the middle, causing the slight dip). (a) Trajectories overlayed on the surface show the mapping. Repulsive (b) and attractive (c) forces were introduced in a small region at the top.

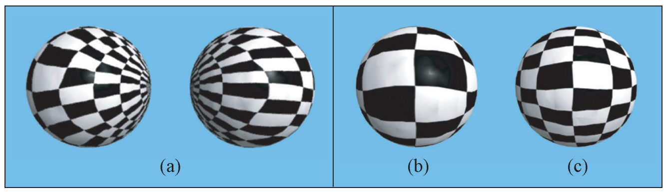

Fig. 13. 3D example of controlling the mapping from one sphere splitting into two. Note that only the target pair of spheres is shown. (a) Applying attractive forces to the right hemisphere of the right sphere pulls the texture towards the right side. (b) View of left side of the left sphere where no forces were applied. (c) View of right side of the right sphere where attractive forces were applied. More checkers appear in (c) because the texture gravitated towards the right side due to the controlling forces. Without any forces, (c) would be identical to (b).

obtuse angles are reduced after each partitioning, but does not give a bound on the number of iterations required to remove all obtuse angles, nor is a bound provided on the number of tetrahedra that may be generated from partitioning one obtuse tetrahedron. Hence, this algorithm is likely to produce a large number of tetrahedra in the refinement process.

10.2 Controlling the Mapping

The approach we have presented automatically generates a mapping between surfaces in a shape transformation. It is often desirable, however, to have control over the mapping. For example, in the splitting spheres transformation, we may want to decompress the texture that is stretched out over the left and right hemispheres of the resulting sphere pair. Zonenschein et al. [1998a], introduce a number of methods for controlling the texture mapping of implicit surfaces that can also be applied here. One of these methods uses attractive and repulsive forces that indirectly influence the resulting mapping. We can implement such forces by modifying the scalar field s at the boundary surfaces, A and B. Recall that we initialized the scalar field s to be 0 at the source shape A, 1 at the target shape B, and interpolated in between A and B. To create an attractive force at some region on B, we initialize s to be greater than 1 within that region. To create a repulsive force, we initialize s to be less than 1 within that region. By changing s, we are effectively changing the gradient of s on the hyper-surface and thus, altering the resulting vector field $\mathbf{T}$ so that the vectors converge toward attractive regions and disperse away from repulsive regions. Figure 12 shows examples of these forces on a mapping generated between identical 2D contours at the top and bottom, and Figure 13 shows a 3D example with the splitting sphere. Because our method generates an explicit mapping between the source and target shapes, we can also apply control methods that have traditionally been used for mesh morphing, such as that of Alexa [2001]. In his work, the meshes of the source and target shapes are described differentially so that local control of the morph can be obtained with a smooth transition.

11. CONCLUSION

Given a morph between two implicit surfaces, we have demonstrated how to create an explicit mapping from one surface to the other. Characteristics of our approach include:

—Produces a smooth mapping between surfaces completely automatically.
—Works for any form of implicit morph.
—Gracefully handles topology changes.
—Preserves texture seams.

We have demonstrated our approach on the transfer of texture between any given pair of objects, regardless of topology. The same approach can be used to transfer other surface information such as a bump or displacement map. Multi-way morphing and analyzing corresponding features of a collection of surfaces are other potential applications.

ACKNOWLEDGMENTS

We thank Gabriel Brostow for support software used in visualizing our results, Peter Lindstrom for rendering software, and James O'Brien for the software to specify constraints used in constructing the spatial warp for the cow to horse transformation. We also thank Eugene Zhang and Sing Bing Kang for helpful suggestions to improve this work.

REFERENCES

ALEXA, M., 2001. Local control for mesh morphing. In *Proceedings of Shape Modeling International*, pp. 209–215.

BENNIS, C., VÉZIEN, J. M., AND IGLÉSIAS, G. 1991. Piecewise surface flattening for non-distorted texture mapping. In *Computer Graphics Proceedings*, Annual Conference Series (SIGGRAPH 1991), pp. 237–246.

BENSON, D. AND DAVIS, J. 2002. Octree textures. In *Computer Graphics Proceedings*, Annual Conference Series (SIGGRAPH 2002), July, pp. 785–790.

BERTALMIO, L., CHENG, T., OSHER, S., AND SAPIRO, G. 2001. Variational problems and partial differential equations on implicit surfaces. *J. Computat. Phys. 174*, 2, 759–780.

BERN, M., CHEW, L. P., EPPSTEIN, D., AND RUPPERT, J. 1995. Dihedral bounds for mesh generation in high dimensions. *6th ACM-SIAM Symposium on Discrete Algorithms*, pp. 189–196.

BHANIRAMKA, P., WENGER, R., AND CRAWFIS, R. 2000. Isosurfacing in higher dimensions. In *Proceedings of Visualization 2000*, Salt Lake City, Utah, October 8–13, pp. 267–273.

BREEN, D. AND MAUCH, S. 1999. Generating shaded offset surfaces with distance, closest-point and color volumes. In *Proceedings of the International Workshop on Volume Graphics*, March, pp. 307–320.

BREEN, D., MAUCH, S., AND WHITAKER, R. T. 2000. 3D scan conversion of CSG models into distance, closest-point and color volumes. M. Chen, A. Kaufman, and R. Yagel, editors, *Volume Graphics*, Springer, London, Chapter 8.

BREEN, D. AND WHITAKER, R. T. 2001. A level-set approach for the metamorphosis of solid models. *IEEE Trans. Visual. Comput. Graph. 7*, 2, April-June, pp. 173–192.

COHEN-OR, D., LEVIN, D., AND SOLOMOVICI, A. 1998. Three dimensional distance field metamorphosis. *ACM Trans. Graph. 17*, 2, April, 116–141.

DEBRY, D., GIBBS, J., PETTY, D. D., AND ROBINS, N. 2002. Painting and rendering textures on unparameterized models. *Computer Graphics Proceedings*, Annual Conference Series (SIGGRAPH 2002), July, pp. 763–768.

DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. H. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. *Computer Graphics Proceedings*, Annual Conference Series (SIGGRAPH 99), August, pp. 317–324.

EELLS, J. AND SAMPSON, J. H. 1998. Another report on harmonic maps. *Bulletin of the London Mathematical Society*, *Vol. 20*, 385–524.

FLOATER, M. S. 1997. Parameterization and smooth approximation of surface triangulations. *Computer Aided Geometric Design Vol. 14*, 231–250.

FUJIWARA, K. 1995. Eigenvalues of laplacians on a closed riemannian manifold and its nets, In *Proceedings of AMS, 123*, pp. 2585–2594.

GREGORY, A., STATE, A., LIN, M. C., MANOCHA, D., AND LIVINGSTON, M. A. 1998. Feature-based surface decomposition for correspondence and morphing between polyhedra. In *Proceedings of Computer Animation*, Philadelphia, PA., pp. 64–71.

HAKER, S., ANGENENT, S., TANNENBAUM, A., KIKINIS, R., SAPIRO, G., AND HALLE, M. 2000. Conformal surface parameterization for texture mapping. *IEEE Trans. Visual. Comput. Graph.*, April—une, Vol. 6(2), 181–187.

HE, T., WANG, S., AND KAUFMAN, A. 1994. Wavelet-based volume morphing. In *Proceedings of Visualization '94*, Washington, D. C., edited by Daniel Bergeron and Arie Kaufman, October 17–21, pp. 85–92.

HUGHES, J. F. 1992. Scheduled fourier volume morphing. *Comput. Graph.* 26, 2 (SIGGRAPH 92), July pp. 43–46.

KANAI, T., SUZUKI, H., AND KIMURA, F. 1997. 3D geometric metamorphosis based on harmonic map. *IEEE Proceedings of the 5th Pacific Conference on Computer Graphics and Applications*, pp. 97–104.

KANAI, T., SUZUKI, H., AND KIMURA, F. 2000. Metamorphosis of arbitrary triangular meshes. *IEEE Comput. Graph. Appl.* pp. 62–75.

KAUL, A. AND ROSSIGNAC, J. 1991. Solid-interpolating deformations: Construction and animation of PIPS. In *Proceedings of Eurographics*, September, pp. 493–505.

KENT, J. R., CARLSON, W. E., AND PARENT, R. E. 1992. Shape transformation for polyhedral objects. *Computer Graphics*, *26*, 2 (SIGGRAPH 92), July, pp. 47–54.

KIMMEL, R. AND SETHIAN, J. A. 1998. Computing geodesic paths on manifolds. In *Proceedings of National Academy of Sciences*, July, Vol. 95(15), pp. 8431–8435.

KOROTOV, S. AND KRIZEK, M. 2001. Acute type refinements of tetrahedral partitions of polyhedral domains. *SIAM J. Numer. Anal. 39*, 2, April—June, pp. 724–733.

LAZARUS, F. AND VERROUST, A. 1998. Three-dimensional metamorphosis: A survey. *The Visual Computer*, Vol. 14(8/9), December, pp. 373–389.

LEE, A., SWELDENS, W., SCHRÖDER, P., COWSAR, L., AND DOBKIN, D. 1998. MAPS: Multiresolution adaptive parameterization of surfaces. *Computer Graphics Proceedings*, Annual Conference Series (SIGGRAPH 98) July, pp. 95–104.

LEE, A., DOBKIN, D., SWELDENS, W., AND SCHRÖDER, P. 1999. Multiresolution mesh morphing. *Computer Graphics Proceedings*, Annual Conference Series (SIGGRAPH 99) August, pp. 343–350.

LEVY, B. AND MALLET, J. 1998. Non-distorted texture mapping for sheared triangulated meshes. *Computer Graphics Proceedings*, Annual Conference Series (SIGGRAPH 98), July, pp. 343–352.

LEVY, B. 2001. Constrained texture mapping for polygonal meshes. *Computer Graphics Proceedings*, Annual Conference Series (SIGGRAPH 2001), pp. 417–424.

LEVY, B., PETITJEAN, S., RAY, N., AND MAILLOT, J. 2002. Least squares conformal maps for automatic texture atlas generation. *Computer Graphics Proceedings*, Annual Conference Series (SIGGRAPH 2002), pp. 362–371.

LORENSON, W. AND CLINE, H. E. 1987. Marching cubes: A high resolution 3-D surface construction algorithm. *Computer Graphics*, *21*, 4 (SIGGRAPH 87), July, pp. 163–169.

MAILLOT, J., YAHIA, H., AND VERROUST, A. 1993. Interactive texture mapping. *Computer Graphics Proceedings*, Annual Conference Series (SIGGRAPH 93), pp. 27–34.

MEMOLI, F., SAPIRO, G., AND OSHER, S. 2004. Solving variational problems and partial differential equations mapping into general target manifolds. *J. Computat. Phys. 195*, 1, March, pp. 263–292.

MEYER, M., DESBRUN, M., SCHRÖDER, P., AND BARR, A. 2002. Discrete differential-geometry operators for triangulated 2-Manifolds. *Proceedings of VisMath*.

OHBUCHI, R., KOKOJIA, Y., AND TAKAHASHI, S. 2001. Blending shapes by using subdivision surfaces. *Computers and Graphics*, February Vol. 25, pp. 41–58.

PAYNE, B. A. AND TOGA, A. W. 1992. Distance field manipulation of surface models. *IEEE Comput. Graph. Appl. 12*, 1, January, pp. 65–71.

PEDERSEN, H. K. 1995. Decorating implicit surfaces. *Computer Graphics Proceedings*, Annual Conference Series (SIGGRAPH 1995), August, pp. 291–300.

PINKALL, U. AND POLTHIER, K. 1993. Computing discrete minimal surfaces and their conjugates. *Exper. Math. 2*, 1, pp. 15–36.

PLAZA, A. AND RIVARA, M. 2003. Mesh refinement based on the 8-tetrahedra longest-edge partition. In *Proceedings, 12th International Meshing Roundtable*, Sandia National Laboratories, Sept. pp. 67–78.

PRAUN, E., SWELDENS, W., AND SCHRÖDER, P. 2001. Consistent mesh parameterizations. *Computer Graphics Proceedings*, Annual Conference Series (SIGGRAPH 2001), August, pp. 179–184.

SANDER, P. V., SNYDER, J., GORTLER, S. J., AND HOPPE, H. 2001. Texture mapping progressive meshes. *Computer Graphics Proceedings*, Annual Conference Series (SIGGRAPH 01), July, pp. 409–416.

SHEFFER, A. AND DE STURLER, E. 2001. Parameterization of faceted surfaces for meshing using angle based flattening. *Engineering with Computers*, Vol. 17, pp. 326–337.

STANDER, B. T. AND HART, J. C. 1997. Guaranteeing the topology of an implicit surface polygonization for interactive modeling. *Computer Graphics Proceedings*, Annual Conference Series (SIGGRAPH 97), August, pp. 279–286.

TAKAHASHI, S., KOKOJIMA, Y., AND OHBUCHI, R. 2001. Explicit control of topological transitions in morphing shapes of 3D meshes. *Pacific Graphics 2001 Proceedings*, Tokyo, Japan, October, pp. 70–79.

TIGGES, M. AND WYVILL, B. 1999. A field interpolated texture mapping algorithm for skeletal implicit surfaces. *Computer Graphics International 1999*, June, pp. 25–32.

TURK, G. AND O'BRIEN, J. F. 1999. Shape transformation using variational implicit functions. *Computer Graphics Proceedings*, Annual Conference Series (SIGGRAPH 99) August, pp. 335–342.

ÜNGÖR, A. 2001. Tiling 3D euclidean space with acute tetrahedra. *13th Canadian Conference on Computational Geometry*, August, pp. 169–172.

WYVILL, G. 1997. *Animation and Special Effects*, chapter in *Introduction to Implicit Surfaces*, J. Bloomenthal (ed.), Morgan Kaufmann Publishers, Inc., San Francisco, California, pp. 242–269.

YEZZI, A. AND PRINCE, J. 2001. A PDE approach for measuring tissue thickness. *Computer Vision and Pattern Recognition 2001*, Kauai, Hawaii, December, pp. 87–92.

ZHANG, E., MISCHAIKOW, K., AND TURK, G. 2003. Feature-based surface parameterization and texture mapping. *Tech. Rept. 03-29, Graphics, Visualization and Usability Center, Georgia Tech*.

ZONENSCHEIN, R., GOMES, J., VELHO, L., AND DE FIGUEIREDO, L. H. 1997. Texturing implicit surfaces with particle systems. *Visual Proceedings of ACM SIGGRAPH 97*, August, p. 172.

ZONENSCHEIN, R., GOMES, J., VELHO, L., AND DE FIGUEIREDO, L. H. 1998a. Controlling texture mapping onto implicit surfaces with particle systems. In *Proceedings of 3rd Eurographics Workshop on Implicit Surfaces*, June, pp. 131–138.

ZONENSCHEIN, R., GOMES, J., VELHO, L., DE FIGUEIREDO, L. H., TIGGES, M., AND WYVILL, B. 1998b. Texturing composite deformable implicit objects. In *Proceedings of 11th International Symposium on Computer Graphics, Image Processing, and Vision*, October, pp. 346–353.

Received August 2003; revised June 2004; accepted October 2004