

Huaiping Yang · Bert Jüttler

# 3D Shape Metamorphosis Based on T-spline Level Sets

**Abstract** We propose a new method for 3D shape metamorphosis, where the in-between objects are constructed by using T-spline scalar functions. The use of T-spline level sets offers several advantages: First, it is convenient to handle complex topology changes without the need of model parametrization. Second, the constructed objects are smooth ( $C^2$  in our case). Third, high quality meshes can be easily obtained by using the marching triangulation method. Fourth, the distribution of the degrees of freedom can be adapted to the geometry of the object.

Given one source object and one target object, we firstly find a global coordinate transformation to approximately align the two objects. The T-spline control grid is adaptively generated according to the geometry of the aligned objects, and the initial T-spline level set is found by approximating the signed distance function of the source object. Then we use an evolution process, which is governed by a combination of the signed distance function of the target object and a curvature-dependent speed function, to deform the T-spline level set until it converges to the target shape. Additional intermediate objects are inserted at the beginning/end of the sequence of generated T-spline level sets, by gradually projecting the source/target object to the initial/final T-spline level set. A fully automatic algorithm is developed for the above procedures. Experimental result are presented to demonstrate the effectiveness of our method.

**Keywords** Computer animation · Morphing · T-spline · level sets

---

## 1 Introduction

Given two geometric models, one source  $\Omega_A$  and one target  $\Omega_B$ , shape metamorphosis generates a sequence

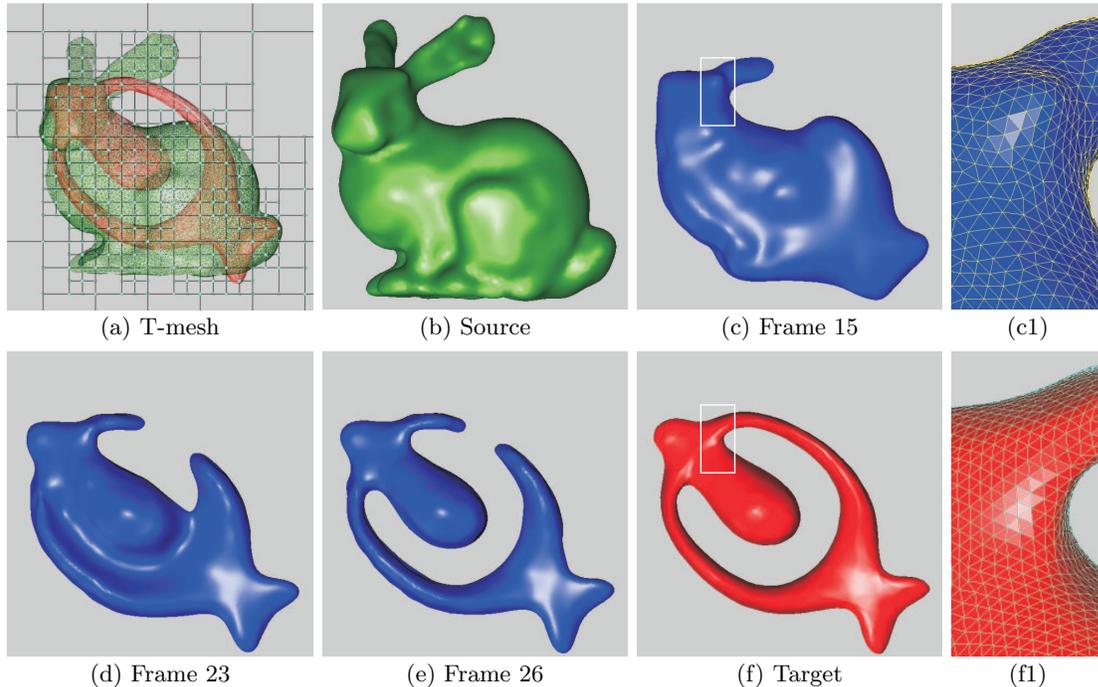
---

H. Yang, B. Jüttler  
Institute of Applied Geometry, Johannes Kepler University  
4040 Linz, Austria  
E-mail: yang.huaiping@jku.at, bert.juettler@jku.at

of in-between models  $\{\Omega_t | t \in [0, 1]\}$ , where  $M_t$  continuously changes its shape from the source ( $\Omega_0 = \Omega_A$ ) into the target ( $\Omega_1 = \Omega_B$ ). It is also called *morphing* within the community of image processing and computer graphics. Shape metamorphosis has been used frequently in computer animation, medical imaging and scientific visualization [1]. In comparison with *image morphing*, where the 2D image (appearance) of an object is gradually transformed into the image of another object, 3D shape metamorphosis directly manipulates the 3D object itself such that the shape of one object is smoothly changed into the shape of another object. While 2D image morphing techniques are unable to correctly handle changes in illumination and visibility, the morphing results created by 3D metamorphosis is independent of the viewing and lighting parameters.

Depending on the different representations of the intermediate objects, 3D shape morphing techniques can be divided into two categories: surface-based (or boundary-based) methods and volume-based methods. The surface-based morphing methods transform the surface patches (usually polygonal meshes) of the source model into the surface patches of the target model. The volume-based morphing methods represent 3D objects as zero level sets of volumetric implicit functions, and manipulate the function values in order to make one embedded object become another.

Most existing 3D morphing techniques are based on the polygonal mesh representation, due to the popularity and widespread use of mesh geometry in graphics. Mesh-based morphing methods are typically faster to compute and require less memory because they operate on a lower-dimensional representation of an object than do volume-based methods. However, transforming between objects of different topologies is considerably more difficult with mesh-based methods, since it is really far from trivial to define a good mapping (correspondence) between the source object and the target object. Also, even the correspondence problem is solved, a reasonable path still has to be found in order to prevent self-intersections of the in-between mesh surfaces. On the contrary, volume-based



**Fig. 1** Morphing a bunny into a petal torus. The T-mesh is constructed for the two objects that have already been aligned in (a). The morphing sequence of T-spline level sets is shown in (b), (c), (d), (e) and (f). Close-up views of (c) and (f) are shown in (c1) and (f1) (flat shading used).

morphing methods gracefully handle changes in topology between objects and do not create self-intersecting surfaces.

In this paper, we present a new volume-based method for 3D shape metamorphosis. Different from most commonly used level set approaches [6,8], where discretized voxel grids are used to store the implicit function values, we use smooth T-spline scalar functions to implicitly define the in-between objects. As the result, the generated in-between objects are smooth ( $C^2$  for cubic T-splines), and the differential geometry (normals and curvatures) of them can be analyzed and exactly computed. Since the T-spline control grid (or T-mesh, see Figure 1(a)) is fixed during the morphing process, only the T-spline control coefficients need to be stored for the intermediate objects. High quality meshes (see Figure 1 (c1) and (f1)) can be easily obtained from the implicit T-spline functions by using the marching triangulation [10] method. Our method is general enough to produce transformations between shapes of any topology.

The remainder of the paper is organized as follows: Section 2 describes related work. Section 3 describes the key part of our method – the T-spline level set model for metamorphosis. Section 4 describes the complete algorithm of our morphing approach and some implementation details. After presenting some experimental results in Section 5, we conclude this paper and discuss future work.

## 2 Related Work

As described previously, most existing 3D morphing algorithms generally fall into two categories: surface-based approaches and volume-based approaches. Besides these two categories, Turk and O’Brien [26] perform shape transformation between 3D shapes by solving a 4D interpolation problem. Bao et al. [3] present a physically based morphing method via dynamic meshless simulation on point-sampled surfaces.

Surface-based approaches are usually used for morphing 3D polygonal meshes, where the *correspondence* problem and the *path* [27] problem are believed to be the two main difficulties for mesh morphing methods. The *correspondence* problem means to find a good mapping (correspondence) between pairs of locations on the boundaries of two meshes. The *path* problem is to create smooth paths between corresponding vertices of the two meshes, such that no self-intersections happen in the intermediate meshes. Numerous surface-based approaches have been proposed to deal with these two problems. The readers are referred to [1] for more details.

Volume-based approaches do not suffer from these problems [17]. They deal with sampled or volumetric representations of the objects, where the objects are described as (zero) level sets of functions defined in the whole 3D space. Although it seems that any kind of continuous interpolation between the functions defining the source object and the target object will at least pro-

duce some "smooth" transformation [22], a simple linear interpolation scheme yields unsatisfactory results in some cases. A nice morphing process should avoid unnecessary distortions or change in topology [17], such as the creation of many connected components. Hughes [12] performs interpolation on the frequency domain of the shapes by utilizing Fourier transforms of the two volumetric models. He et al. [11] decompose the volumetric functions with a wavelet transform. They also present a technique for establishing a suitable correspondence among object voxels. Kaul and Rossignac [15,23] use a weighted Minkowski sum with time changing coefficients to compute the metamorphosis. Galin and Akkouché [9] also use Minkowski sums in order to characterize the skeletons of intermediate shapes. Jin et al. [13] propose a practical method for blob-based liquid morphing by employing the medial axis sphere-tree of a polygonal model. Chen et al. [7] develop a volume distortion algorithm based on the control of two sets of disk fields in an interactive environment. Nieda et al. [20] propose an approach to detect and classify topological changes of shape metamorphosis based on R-functions [21].

Lerios et al. [19] present a two-staged algorithm for creating volume morphs: first a *warping* of the two input volumes, then a *blending* of the resulting warped volumes. The first stage is just a 3D extension of Beier and Neely's [4] 2D image warping technique. In order to reduce the distortion of the intermediate shapes, Cohen-Or et al. [8] decompose the warp function into a *rigid* (rotation and translation) part and an *elastic* part, which is based upon a set of user-supplied corresponding anchor points. Breen et al. [6] describe the metamorphosis as an optimization process of an objective function which measures the similarity between two shapes. The proposed metric is simple, which is just trying to maximize the volume shared by the interiors of the two objects until the two objects become the same. Based on this metric, the level set deformation is governed by the signed distance transform of the target surface.

We use a similar strategy as proposed by Breen et al. [6]. The difference is that we use smooth T-splines instead of discretized voxels to define the level set functions. The number and distribution of T-spline control points can be made adaptive to the geometry of the two shapes. In order to reduce the possibility of unnecessary topology changes during the morphing process, we incorporate a curvature-dependent speed term into the evolution speed function, instead of use the signed distance transform only as in [6]. We also insert additional animation frames into the beginning/end of the sequence of intermediate T-spline level sets, by continuously projecting the source/target object onto the initial/final T-spline level set. This is to make sure that the morphing process starts from the source object and ends at the target object, since the corresponding T-spline level sets are only approximations of them.

### 3 T-spline Level Sets for Metamorphosis

In this section, we give the definition of T-spline level sets, and describe how to (approximately) convert the given objects (e.g. mesh surfaces) into T-spline level sets. Then we discuss the evolution process of T-spline level sets, in order to transform the shape of the source object  $\Omega_A$  into that of the target shape  $\Omega_B$ . We assume  $\Omega_A$  and  $\Omega_B$  are given by triangular meshes, although other kinds of representations can also be handled by our method.

#### 3.1 Definition of T-spline level sets

T-splines [24] are generalizations of tensor product B-splines. We now introduce *T-spline level sets* in 3D. Let  $f(x, y, z)$  be a trivariate T-spline function defined over some domain  $D$ ,

$$f(x, y, z) = \frac{\sum_{i=1}^n c_i B_i(x, y, z)}{\sum_{i=1}^n B_i(x, y, z)}, \quad (x, y, z) \in D \subset \mathbb{R}^3 \quad (1)$$

with the real coefficients (control points)  $c_i$ ,  $i = 1, 2, \dots, n$ , where  $n$  is the number of control points. For cubic T-splines, the basis functions are  $B_i(x, y, z) = N_{i0}^3(x)N_{i0}^3(y)N_{i0}^3(z)$  where  $N_{i0}^3(x)$ ,  $N_{i0}^3(y)$  and  $N_{i0}^3(z)$  are certain cubic B-splines, whose associated knot vectors are determined by the T-spline control grid (T-mesh). The zero set of the function  $f$  is defined by

$$\Gamma(f) = \{ (x, y, z) \in D \subset \mathbb{R}^3 \mid f(x, y, z) = 0 \}, \quad (2)$$

and it is called a *T-spline level set*.

In order to simplify the notation, we use  $\mathbf{x}$  to represent the point  $\mathbf{x} = (x, y, z)$  in 3D, and gather the control coefficients (in a suitable ordering) in a column vector  $\mathbf{c}$ . The T-spline basis functions form another column vector  $\mathbf{b} = [b_1, b_2, \dots, b_n]^T$ , where

$$b_i = B_i(\mathbf{x}) / \sum_{i=1}^n B_i(\mathbf{x}), \quad i = 1, 2, \dots, n.$$

The *T-spline level set*  $\Gamma(f)$  is defined as the zero set of the function  $f(\mathbf{x}) = \mathbf{b}(\mathbf{x})^T \mathbf{c}$ . For a fixed T-spline control grid (thus  $\mathbf{b}$  is fixed), the T-spline level set is determined only by the control coefficients  $\mathbf{c}$ .

Since a T-spline function is piecewise rational, the T-spline level sets are piecewise algebraic surfaces. Moreover, if no singular points are present, they inherit the order of differentiability of the basis functions, i.e., they are  $C^2$  in the cubic case. Derivatives of  $f$ , which will be needed for formulating the evolution, can easily be evaluated.

### 3.2 Conversion to T-spline level sets

We use T-spline level sets to represent 3D objects. The T-spline control grid (T-mesh) is fixed from the beginning, and only the T-spline control coefficients are changed during the morphing process. Before the generation of T-mesh, the source object and the target object are aligned with each other, such that the two shapes are made relatively similar. An automatic alignment method will be described in Section 4.2.

After the alignment, we normalize the two objects to be within a cubic domain  $D = [-1, 1] \times [-1, 1] \times [-1, 1]$ , and uniformly sample a set of points on the surfaces (triangular meshes) of the two objects. Then we generate the T-mesh in the domain, through octree subdivision of those cells containing more than  $n_T$  sample points. The sampling density and the threshold value  $n_T$  are pre-scribed constant coefficients. In this way, the distribution of T-spline control coefficients is adapted to the geometry of the objects. In order to speed up the computation of T-spline functions, we use a similar octree to store the information of associated T-spline control coefficients for any point in the domain. In our experiments, the maximum subdivision depth for T-mesh generation is always no more than 6.

After the T-mesh is obtained, we initialize the T-spline level set as an approximation of the source object. The initial T-spline control coefficients  $\mathbf{c}$  are computed by minimizing the following objective function

$$E_0 = \int_D \omega(\mathbf{x})(f(\mathbf{x}) - d_A(\mathbf{x}))^2 dD \rightarrow \text{Min}, \quad (3)$$

where  $f(\mathbf{x})$  is the trivariate T-spline function as defined in (1),  $d_A(\mathbf{x})$  is the signed distance function to the source object  $\Omega_A$ , and  $\omega(\mathbf{x}) = e^{-d_A^2(\mathbf{x})}$  is a positive weighting function. More precisely, we choose the T-spline function  $f$  as a weighted least-squares approximation of the signed distance function  $d_A$ . The weighting function  $\omega(\mathbf{x})$  takes a smaller value when the point  $\mathbf{x}$  is farther from the source object.

For the actual computation, a discretized version (which can be seen as a numerical integration) is more appropriate, i.e., we replace  $E_0$  with

$$E = \frac{V(D)}{N_0} \sum_{j=1}^{N_0} \omega(\mathbf{x}_j)(f(\mathbf{x}_j) - d_A(\mathbf{x}_j))^2, \quad (4)$$

where  $\mathbf{x}_j$ ,  $j = 1, \dots, N_0$  ( $N_0 \gg n$ ) is a sequence of sampling points, which are uniformly distributed in the T-spline function domain  $D$ .  $V(D)$  is the volume of the domain  $D$ .

In our case, the function  $f$  has the form  $f(\mathbf{x}) = \mathbf{b}(\mathbf{x})^\top \mathbf{c}$ , hence the function  $E$  is a non-negative definite quadratic function of the unknown T-spline control coefficients  $\mathbf{c}$ . The solution  $\mathbf{c}$  is found by solving a sparse linear system of equations,  $\nabla E = 0$ , and the initial T-spline level set  $L_0$  is obtained.

If the accuracy of the approximation is not sufficient, a better  $L_0$  can be found by using more degrees of freedom (T-spline control coefficients) and applying the 'final refinement' step [29]. The same strategy can be also used for the approximation of the target object after the evolution of T-spline level sets stops.

### 3.3 Metamorphosis of T-spline level sets

Since the T-mesh is fixed during the morphing process, the T-spline level set function can be written as

$$f(\mathbf{x}, \tau) = \mathbf{b}(\mathbf{x})^\top \mathbf{c}(\tau), \quad (5)$$

with the time variable  $\tau$ . Consider the evolution process

$$\dot{\mathbf{x}} = v(\mathbf{x}, \tau)\mathbf{n}, \quad \mathbf{x} \in \Gamma(f), \quad (6)$$

where the dot of  $\dot{\mathbf{x}}$  means the time derivative, and  $v$  is a scalar-valued *speed function* along the normal direction  $\mathbf{n} = \nabla f / |\nabla f|$  of  $\Gamma$ . In [29], we have shown that this kind of T-spline level sets evolution can be formulated as a least squares problem, where a distance field constraint is incorporated to avoid additional branches and singularities without having to use re-initialization steps. An extended version of this paper is available as a technical report on the webpage [28].

There are a number of choices of the speed function  $v$  for the metamorphosis from  $\Omega_A$  to  $\Omega_B$ . In order to avoid numerical difficulties and to avoid discontinuities in the solution,  $v$  should be continuous. Furthermore,  $v$  should carry information about the shape of the target into 3D so that shapes tend to "look like" the target as they get nearer. Breen and Whitaker [6] suggest that a natural choice of  $v$  is the *signed distance transform* of the target surface  $\Omega_B$  or some monotonic function thereof, i.e.,

$$v(\mathbf{x}) = g(d_B(\mathbf{x})), \quad g(0) = 0 \quad \text{and} \quad g'(\mathbf{x}) > 0, \quad (7)$$

where  $d_B$  is the signed distance function to the target object  $\Omega_B$ . The source object will shrink in those areas where it is outside the target object and will expand in those areas inside the target object. It is also proved [6] that if the initial object ( $L_0$ ) and the target object overlap, the final solution of the metamorphosis will be identical to the target.

However, direct use of  $d_B$  as the speed function may cause additional topology changes, which is undesired for a nice morphing process. Figure 2 shows a morphing example when  $v(\mathbf{x}) = d_B(\mathbf{x})$ , where the source object ("┌" shape) and the target object ("┐" shape) have the same topology (genus 0). The morphing sequence in Figure 2 demonstrates that the "┌" shape is first split into two components, then one component is vanished and the other component is transformed into the final "┐" shape. This undesirable artifact is a typical problem for Breen and Whitaker's method [6] when the source object

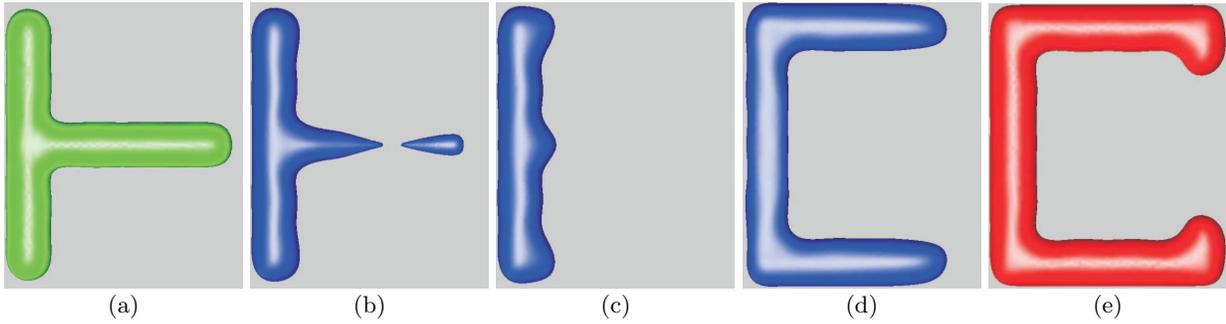


Fig. 2 Morphing a "T" into a "C" by using the original speed function proposed in [6].

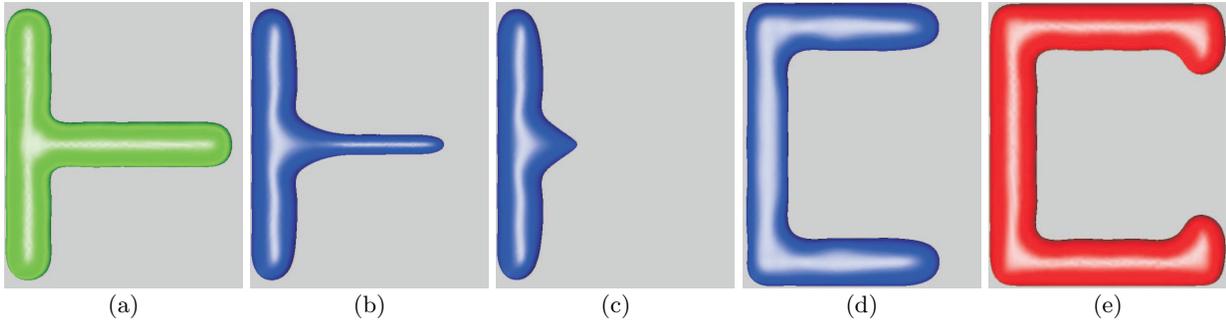


Fig. 3 Morphing a "T" into a "C" by using the improved speed function.

has some long narrow features, which are not matched by the target object.

In order to handle this problem, we use the speed function  $v$  as a linear combination of two terms, the signed distance function  $d_B$  and the mean curvature  $\kappa$  of the morphing object, i.e.,

$$v(\mathbf{x}) = \lambda(\mathbf{x})d_B(\mathbf{x}) + (1 - \lambda(\mathbf{x}))\kappa(\mathbf{x}), \quad (8)$$

where  $\lambda(\mathbf{x}) = e^{-d_B^2(\mathbf{x})}$  is a variable weighting coefficient. By using this improved speed function, the mean curvature term will play the key rule when the morphing object is far away from the target ( $d_B(\mathbf{x})$  is large), which is helpful to avoid additional topology changes of the morphing object. When the morphing object gets close to the target  $\Omega_B$ , the signed distance term will converge the morphing process to  $\Omega_B$ . Of course there are many choices of the weighting function  $\lambda(\mathbf{x})$ . In our experiments we chose  $\lambda(\mathbf{x}) = e^{-d_B^2(\mathbf{x})}$  to get successful examples.

Figure 3 shows the morphing sequence by using the new speed function. Thanks to the incorporated mean curvature speed term, now the right bar of the "T" shape is smoothly dissolved without unwanted splitting. Hence, the new speed function generates a better morphing sequence than the signed distance function, see Fig. 2.

As described in our previous paper [29], for each evolution (morphing) step, the time derivatives  $\dot{\mathbf{c}}(\tau)$  of the T-spline control coefficients can be found by solving a sparse linear system of equations. Then we generate the

updated control coefficients

$$\mathbf{c}(\tau + \Delta\tau) = \mathbf{c}(\tau) + \dot{\mathbf{c}}\Delta\tau. \quad (9)$$

by using an explicit Euler step  $\Delta\tau$ .  $\Delta\tau$  is chosen as  $\min(1, \{h/v(\mathbf{x}_j, \tau)\}_{j=1, \dots, N_1})$ , where  $\mathbf{x}_j$ ,  $j = 1, \dots, N_1$  ( $N_1 \gg n$ ) are a set of uniformly sampled points along the T-spline level set  $L_\tau$ , and  $h$  is a user-defined constant to indicate the morphing step size. We use the marching triangulation [10] method to get the uniformly distributed sample points  $\mathbf{x}_j$ .

## 4 Algorithm and Implementation

This section describes the whole algorithm of 3D shape metamorphosis based on the proposed T-spline level set models. The algorithm takes two triangular meshes (the source object  $\Omega_A$  and the target object  $\Omega_B$ ) and the morphing step size  $h$  as input, and produces a sequence of in-between objects (represented by T-spline level sets or triangular meshes) as output.

### 4.1 Outline of the Algorithm

The presented algorithm can be divided into three stages: initialization, evolution, and post-processing. Figure 4 shows the flow chart of our algorithm.

In the initialization stage (stage 1), the source object  $\Omega_A$  is aligned with the target object  $\Omega_B$  such that the

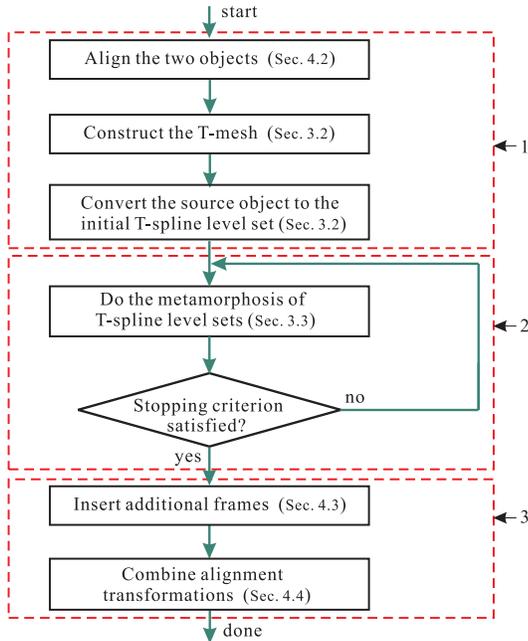


Fig. 4 Flow chart of the algorithm.

two objects have similar shapes and orientations, which will be described in Section 4.2. After alignment, the T-mesh is generated and the T-spline level set  $L_0$  is initialized as an approximation of the source object  $\Omega_A$ , following the steps described in Section 3.2.

During the evolution stage (stage 2), the T-spline level set  $L_\tau$  is evolved towards the target shape  $\Omega_B$  step by step, guided by the morphing speed function, as described in Section 3.3, until some stopping criteria is satisfied (e.g.  $L_\tau$  is close enough to  $\Omega_B$ ). If  $n_l$  T-spline level sets are generated, then the  $n_l$  sequences of T-spline control coefficients  $\{\mathbf{c}(\tau_i)|i = 0, 1, \dots, n_l - 1\}$  are stored.

Finally, in the post-processing stage (stage 3), we insert additional frames (meshes) to the beginning/end of the sequence of generated T-spline level sets  $\{L_i|i = 0, 1, \dots, n_l - 1\}$ , by gradually projecting  $\Omega_A/\Omega_B$  to the initial/final T-spline level set ( $L_0/L_{n_l-1}$ ) (cf. Section 4.3). The final morphing sequence is obtained by combining the generated objects with the alignment transformation (cf. Section 4.4).

## 4.2 Objects Alignment

There are at least three reasons to align the two objects ( $\Omega_A$  and  $\Omega_B$ ) for metamorphosis: First, the convergence of the T-spline level sets evolution is guaranteed only if the two objects overlap [6]. Second, a more natural and realistic metamorphosis may be produced after alignment. Third, it provides an intuitive way for the user to control the morphing process [2].

The authors in [6] present an automatic method to accomplish the alignment by calculating two affine trans-

formations (consisting of rotation, translation, and non-uniform scaling), which are defined by the centroids and principal axes of the two objects. The principal axes are computed as the eigenvectors of the covariance matrix associated with each object. We have developed a similar technique for automatically aligning objects  $\Omega_A$  and  $\Omega_B$ , such that the transformation matrix  $T = T_s T_r T_t$  is acquired ( $T_s$ ,  $T_r$ , and  $T_t$  represent for the scaling, rotation and translation respectively). The updated objects after alignment are represented by  $\Omega'_A$  and  $\Omega'_B$  respectively.

The alignment also can be done interactively by the user. For this purpose, we have also developed a software tool that allows a user to interactively position, rotate and scale the source and target objects in order to produce the transformation  $T$ . The coordinate systems of the two objects are aligned and the user is able to manipulate the objects until they are properly overlapped.

## 4.3 Insertion of Additional Frames

The evolution stage generates the sequence of T-spline level sets  $\{L_i|i = 0, 1, \dots, n_l - 1\}$ . Since  $L_0$  is an approximate representation of the aligned source object  $\Omega'_A$ , it may contain aliasing artifacts, especially when  $\Omega'_A$  has some sharp features. In order to make sure that the morphing animation smoothly starts from the source object, we insert additional frames between  $\Omega'_A$  and  $L_0$ . These inserted objects are represented by triangular meshes  $M_A(\tau)$  and obtained by continuously projecting  $\Omega'_A$  to  $L_0$ :

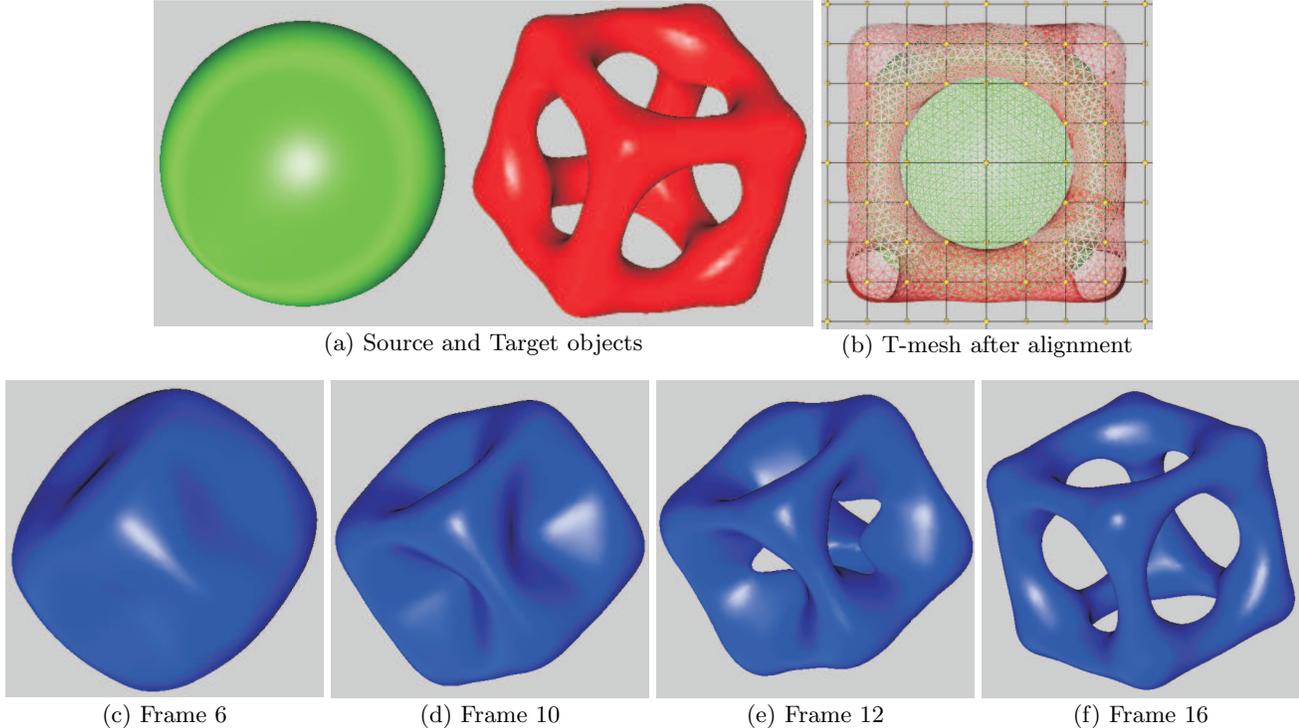
$$\dot{\mathbf{v}}_i = \mathbf{q}_i - \mathbf{v}_i, \quad i = 0, 1, \dots, n_A - 1 \quad (10)$$

where  $\{\mathbf{v}_i|i = 0, 1, \dots, n_A - 1\}$  are the vertices of  $M_A(\tau)$  ( $M_A(0) = \Omega'_A$ ),  $\mathbf{q}_i$  is the closest point (foot point) on  $L_0$  to  $\mathbf{v}_i$ , and  $\dot{\mathbf{v}}_i$  is the time derivative of  $\mathbf{v}_i$ . Thanks to the implicit representation of  $L_0$ , the foot point  $\mathbf{q}_i$  can be efficiently computed by a Newton iteration. By using an explicit Euler integration (similar as that described in Section 3.3), and restricting the update of  $\mathbf{v}_i$  to be within the indicated morphing step size  $h$ , the inserted frames  $\{M_A(\tau_i)|i = 0, 1, \dots, m_A - 1\}$  are computed. Due to the projection operator, no self-intersections are introduced into  $M_A(\tau_i)$ , as long as the initial mesh  $\Omega'_A$  is self-intersection free.

Similarly, we generate additional frames  $\{M_B(\tau_i)|i = 0, 1, \dots, m_B - 1\}$  ( $M_B(\tau_{m_B-1}) = \Omega'_B$ ) by projecting the aligned target mesh  $\Omega'_B$  to the final T-spline level set  $L_{n_l-1}$ . Now the complete morphing sequence has totally  $m_A + n_l + m_B$  frames, including  $m_A$  anterior meshes,  $n_l$  T-spline level sets and  $m_B$  posterior meshes.

## 4.4 Combine Alignment Transformations

As noted previously, until now, the generated morphing sequence are with respect to the local coordinates which



**Fig. 5** Morphing a sphere into a cube with holes.

are created by the alignment of two objects  $\Omega_A$  and  $\Omega_B$ . Suppose  $T_A = T_{s,A}T_{r,A}T_{t,A}$  is the transformation matrix for the alignment of  $\Omega_A$  ( $\Omega_A \rightarrow \Omega'_A$ ), where  $T_{s,A}$ ,  $T_{r,A}$ , and  $T_{t,A}$  represent for the scaling, rotation and translation matrices respectively. Similarly  $T_B = T_{s,B}T_{r,B}T_{t,B}$  is the alignment matrix for  $\Omega_B$  ( $\Omega_B \rightarrow \Omega'_B$ ). Then the inverse transformation matrix for the  $i$ -th object in the morphing sequence is

$$W_i = W_{t,i}W_{r,i}W_{s,i}, \quad (11)$$

where  $W_{t,i} = (1-u)T_{t,A}^{-1} + uT_{t,B}^{-1}$  is the linear interpolation of the translation matrices between  $T_{t,A}^{-1}$  and  $T_{t,B}^{-1}$ ,  $W_{s,i} = (1-u)T_{s,A}^{-1} + uT_{s,B}^{-1}$  is the linear interpolation of the scaling matrices between  $T_{s,A}^{-1}$  and  $T_{s,B}^{-1}$ , and  $u = i/(m_A + n_l + m_B - 1)$ . For the interpolation of rotation matrices  $W_{r,i}$ , we use the spherical linear interpolation based on the quaternion representation [25]. To transform the intermediate objects back to the world coordinate system, the T-spline level sets are first converted into triangular meshes by using the marching triangulation [10] method (which generates over 10,000 triangles within seconds), then each vertex is transformed by  $W_i$  to produce the final morphing sequence of triangular meshes.

Note that the interpolation of alignment transformations is symmetric for the morphing between  $\Omega_A$  and  $\Omega_B$ , i.e.,  $W(u)_{A \rightarrow B} = W(1-u)_{B \rightarrow A}$ . But the final morph of  $\Omega_A \rightarrow \Omega_B$  is different from that of  $\Omega_B \rightarrow \Omega_A$ , since the speed function  $v$  is not symmetric. A possible solution is

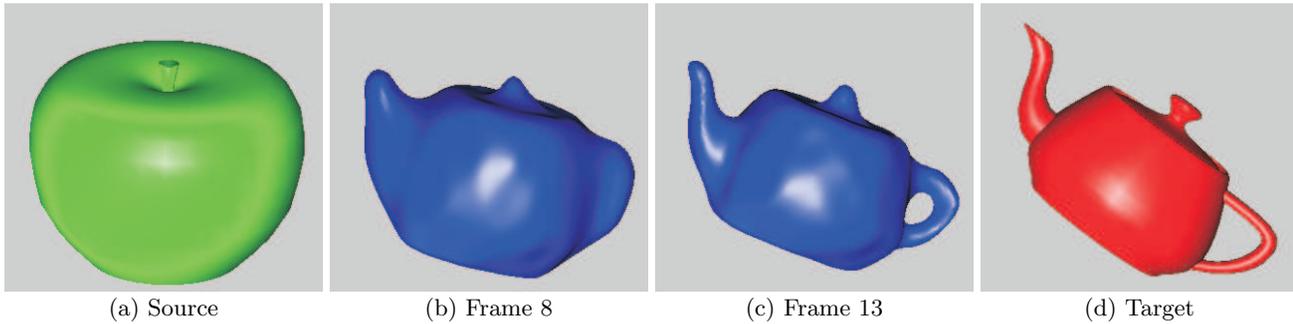
to average the two shapes  $\Omega(u)_{A \rightarrow B}$  and  $\Omega(1-u)_{A \rightarrow B}$ . As a future work we will try to find a symmetric speed function  $v$  for the shape metamorphosis between two objects.

## 5 Experimental Results and Discussion

In this section, we present some examples to demonstrate and discuss the effectiveness of our method. The given source objects and target objects are aligned and normalized within a cubic domain  $D = [-1, 1] \times [-1, 1] \times [-1, 1]$ . The experiments are run on a PC with AMD Opteron(tm) 2.20GHz CPU and 3.25G RAM. Please note that all the presented examples are generated fully automatically without any user interaction, although interactive controls are also possible in the developed software.

### 5.1 Examples

*Example 1: A bunny morphing into a petal torus.* In the first example (see Figure 1), the source object (a bunny) is deforming into the target (a petal torus). The T-spline control grid is constructed with 2449 control coefficients and shown in (a). The morphing sequence is shown between (b) and (f), where the topology of the objects adaptively changes from genus-0 to genus-1. Since the bunny and the petal torus are appropriately overlapped



**Fig. 6** Morphing an apple into a teapot.

under the initial configuration, no alignment transformation is needed to interpolate the intermediate shapes. The entire computation took about 5 minutes.

*Example 2: A sphere morphing into a cube with holes.* The second example illustrates the morphing process between a sphere and a cube with holes (produced by boolean operations between a cube and three cylinders), as shown in Figure 5). The T-spline control grid is constructed with 651 control coefficients. Note that the alignment transformation is pre-computed and interpolated for the morphing sequence, such that the orientation of the objects gradually changes into that of the target. The entire computation took about 1 minute.

*Example 3: An apple morphing into a teapot.* In the third example (see Figure 6), the source object (an apple) is deforming into the target (a teapot). The T-spline control grid is constructed with 1104 control coefficients. The entire computation took about 2 minutes.

*Example 4: A sculpture morphing into the Happy Buddha.* The fourth example illustrates the morphing process between a sculpture and the Happy Buddha (Figure 7). The T-spline control grid is constructed with 4327 control coefficients. The entire computation took about 12 minutes.

*Example 5: A dragon morphing into a tricorn.* The last example illustrates the morphing process between a dragon and a tricorn (Figure 8). The T-spline control grid is constructed with 4915 control coefficients. During the metamorphosis, the translation, rotation and scaling of the objects are smoothly interpolated according to the alignment transformation. The entire computation took about 14 minutes.

## 5.2 Discussion

In our algorithm, most in-between shapes (except the additional frames) are represented by T-spline level sets.

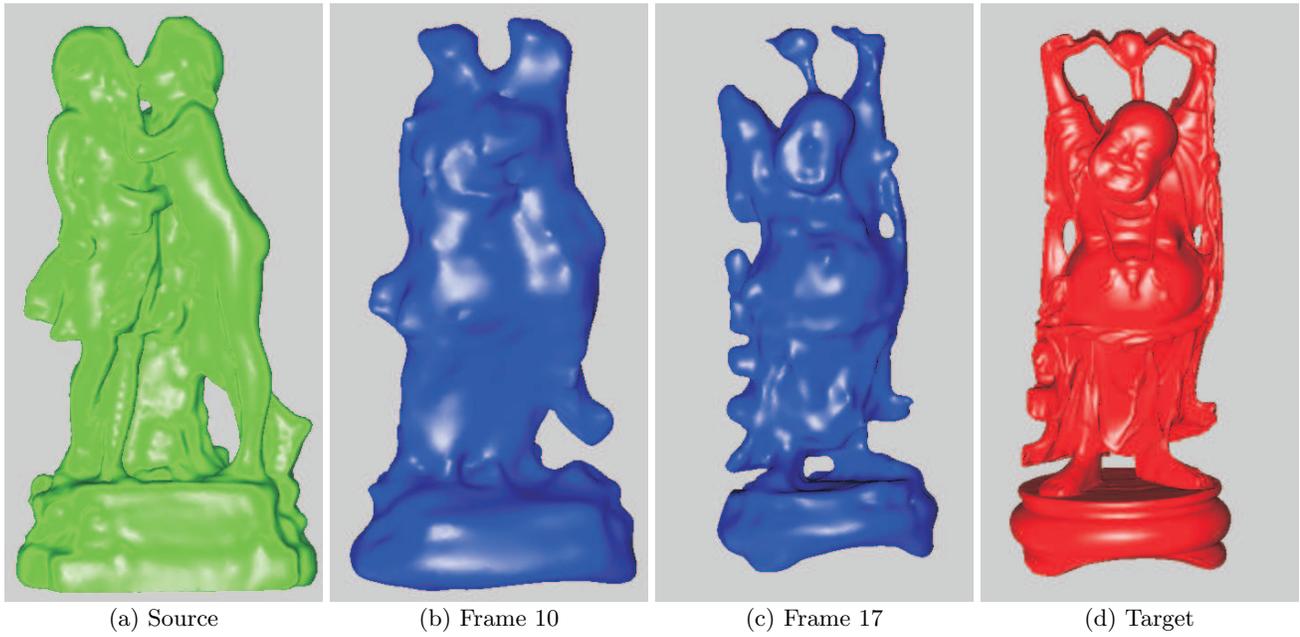
Since the T-spline control grid is fixed during the morphing process, we only need to store the T-spline control coefficients for each T-spline level set. Thus the space complexity is linearly dependent on the number of T-spline control coefficients. Due to the local refinement property of T-splines, the distribution of T-spline control coefficients can be made adaptive to the geometry of the source and target objects (after alignment), which means that ideally the number of T-spline control coefficients increases roughly linearly with the area of the objects.

For each evolution step of a T-spline level set, a sparse linear system is to be solved, and the computation time is dependent on the number of T-spline control coefficients and different solvers used (See [5] for a detailed complexity analysis for solvers for sparse linear systems arising in geometry processing). For the combination of alignment transformations, the T-spline level sets need to be first converted into triangular meshes, and the computation time is linearly dependent on the number of mesh vertices (the resolution of the mesh surface).

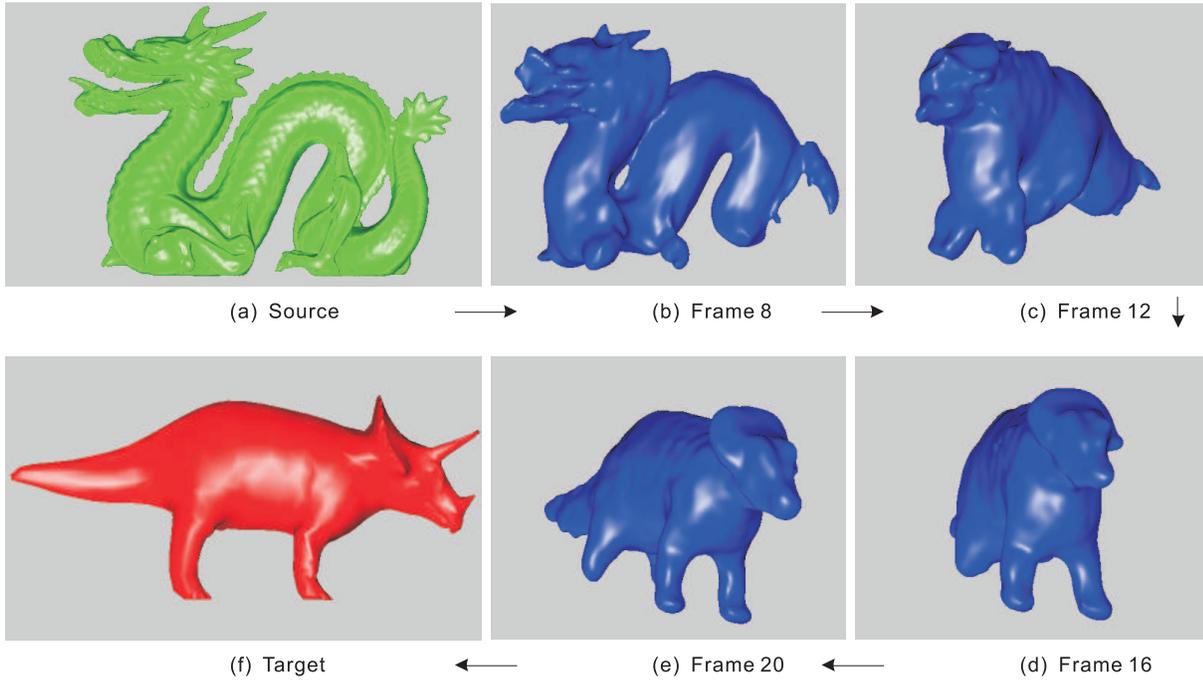
## 6 Conclusions and Future Work

We have introduced a method for 3D shape metamorphosis based on the evolution of T-spline level sets. The T-spline representation of the level set function is sparse and piecewise rational, the distribution of T-spline control coefficients can be made adaptive to the geometry of the objects to be morphed. We have also shown that the morphing process of T-spline level sets can be formulated as least squares problems. A fully automatic algorithm is developed to produce metamorphosis between shapes of any topology.

For the mesh-based morphing methods, the correspondence problem is difficult [16], especially for two objects with different genus [18]. However, this correspondence does provide a powerful way for the user to define a desired morphing process [14]. Since the volume-based morphing methods are parametrization-free, on the one hand they can easily handle complex topology changes, but on the other hand they have problems to (dynamically) maintain the correspondence of features between



**Fig. 7** Morphing a sculpture into the Happy Buddha.



**Fig. 8** Morphing a dragon into a tricorn.

two objects. Thus, in the future, we plan to couple the two approaches in order to obtain a more powerful and flexible hybrid model for metamorphosis.

**Acknowledgements** The first author was supported by a Marie Curie Incoming International Fellowship of the European commission (project 022073 ISIS). The second author was supported by the Austrian Science Fund through the na-

tional research network on Industrial Geometry (S9201). The data sets used for our experiments are courtesy of Stanford computer graphics laboratory and UCI (University of California, Irvine) computer graphics laboratory.

## References

1. Alexa, M.: Recent advances in mesh morphing. *Computer Graphics forum* **21**(2), 173–197 (2002)
2. Bao, H., Peng, Q.: Interactive 3d morphing. *Comput. Graph. Forum* **17**(3), 23–30 (1998)
3. Bao, Y., Guo, X., Qin, H.: Physically based morphing of point-sampled surfaces: Animating geometrical models. *Comput. Animat. Virtual Worlds* **16**(3-4), 509–518 (2005)
4. Beier, T., Neely, S.: Feature-based image metamorphosis. In: *Proc. SIGGRAPH'92*, pp. 35–42 (1992)
5. Botsch, M., Bommers, D., Kobbelt, L.: Efficient linear system solvers for mesh processing. In: R.M. et al. (ed.) *Mathematics of Surfaces XI, LNCS*, vol. 3604, pp. 62–83. Springer, Berlin (2005)
6. Breen, D.E., Whitaker, R.T.: A level-set approach for the metamorphosis of solid models. *IEEE Transactions on Visualization and Computer Graphics* **7**(2), 173–192 (2001)
7. Chen, M., Jones, M.W., Townsend, P.: Volume distortion and morphing using disk fields. *Computers and Graphics* **20**(4), 567–575 (1996)
8. Cohen-Or, D., Solomovic, A., Levin, D.: Three-dimensional distance field metamorphosis. *ACM Trans. Graph.* **17**(2), 116–141 (1998)
9. Galin, E., Akkouché, S.: Blob metamorphosis based on Minkowski sums. *Computer Graphics Forum (Proc. Eurographics'96)* **15**(3), 143–152 (1996)
10. Hartmann, E.: A marching method for the triangulation of surfaces. *The Visual Computer* **14**(3), 95–108 (1998)
11. He, T., Wang, S., Kaufman, A.: Wavelet-based volume morphing. In: *Proc. VIS'94*, pp. 85–92 (1994)
12. Hughes, J.F.: Scheduled Fourier volume morphing. In: *Proc. SIGGRAPH'92*, pp. 43–46 (1992)
13. Jin, X., Liu, S., Wang, C.L., Feng, J., Sun, H.: Blob-based liquid morphing. *Journal of Visualization and Computer Animation* **16**(3-4), 391–403 (2005)
14. Kanai, T., Suzuki, H., Kimura, F.: Metamorphosis of arbitrary triangular meshes. *IEEE Computer Graphics and Applications* **20**(2), 62–75 (2000)
15. Kaul, A., Rossignac, J.: Solid-interpolating deformations: construction and animation of PIPS. In: *Proc. Eurographics'91*, pp. 493–505 (1991)
16. Kraevoy, V., Sheffer, A.: Cross-parameterization and compatible remeshing of 3d models. *ACM Trans. Graph. (Proc. SIGGRAPH'04)* **23**(3), 861–869 (2004)
17. Lazarus, F., Verroust, A.: Three-dimensional metamorphosis: a survey. *The Visual Computer* **14**(8-9), 373–389 (1998)
18. Lee, T.Y., Yao, C.Y., Chu, H.K., Tai, M.J., Chen, C.C.: Generating genus- $n$ -to- $m$  mesh morphing using spherical parameterization. *Journal of Visualization and Computer Animation* **17**(3-4), 433–443 (2006)
19. Leros, A., Garfinkle, C.D., Levoy, M.: Feature-based volume metamorphosis. In: *Proc. SIGGRAPH'95*, pp. 449–456 (1995)
20. Nieda, T., Pasko, A., Kunii, T.L.: Detection and classification of topological evolution for linear metamorphosis. *The Visual Computer* **22**(5), 346–356 (2006)
21. Pasko, A., Adzhiev, V., Sourin, A., Savchenko, V.: Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer* **11**(8), 429–446 (1995)
22. Payne, B., Toga, A.: Distance field manipulation of surface models. *IEEE Computer Graphics and Applications* **12**(1), 65–71 (1992)
23. Rossignac, J., Kaul, A.: AGRELS and BIBs: metamorphosis as a Bézier curve in the space of polyhedra. In: *Proc. Eurographics'94*, pp. 179–184 (1994)
24. Sederberg, T.W., Zheng, J., Bakenov, A., Nasri, A.: T-splines and T-NURCCs. *ACM Transactions on Graphics* **22**(3), 477–484 (2003)
25. Shoemake, K.: Animating rotation with quaternion curves. In: *Proc. SIGGRAPH'85*, pp. 245–254 (1985)
26. Turk, G., O'Brien, J.F.: Shape transformation using variational implicit functions. In: *Proc. SIGGRAPH'99*, pp. 335–342 (1999)
27. Yan, H.B., Hu, S.M., Martin, R.: 3d morphing using strain field interpolation. *Journal of Computer Science and Technology* **22**(1), 147–155 (2007)
28. Yang, H., Fuchs, M., Jüttler, B., Scherzer, O.: Evolution of T-spline level sets with distance field constraints for geometry reconstruction and image segmentation. *Technical Report 01*, <http://www.ig.jku.at> (2005)
29. Yang, H., Fuchs, M., Jüttler, B., Scherzer, O.: Evolution of T-spline level sets with distance field constraints for geometry reconstruction and image segmentation. In: *Proc. SMI'06*, pp. 247–252 (2006)



ization.



(World Scientific) and serves on the program committees of various international conference (e.g., the SIAM conference on Geometric Design and Computing 2007).

**HUAPING YANG** received the BE (1998) degree in hydraulic engineering and the PhD (2004) degree in computer science from Tsinghua University, China. In 2004, he did a one-year postdoc in the computer graphics group at the University of Hong Kong. In 2005, he starts a postdoc at JKU Linz, Austria, in the field of applied geometry, funded by a Marie Curie incoming international fellowship. His research interests include geometric modeling, computer

graphics, and scientific visual-

**BERT JUETTLER** is professor of Mathematics at Johannes Kepler University of Linz, Austria. He did his PhD studies (1992-94) at Darmstadt University of Technology under the supervision of the late Professor Josef Hoschek. His research interests include various branches of applied geometry, such as Computer Aided Geometric Design, Kinematics and Robotics. Bert Jüttler is member of the Editorial Boards of *Computer Aided Geometric Design* (Elsevier) and the *Int. J. of Shape Modeling*