Feature-Based Surface Light Field Morphing

Eunhee Jeong[1] Mincheol Yoon[1] Yunjin Lee[1] Minsu Ahn[1] Seungyong Lee[1] Baining Guo[2]

[1]POSTECH* [2]Microsoft Research Asia†

Abstract

A surface light field is a function that gives the colors of each object point viewed from different directions. An object representation with a surface light field provides a nice structure for 3D photography. This paper presents a feature-based morphing technique for two objects equipped with surface light fields. The technique consists of geometry morphing and in-between light field mapping. Geometry morphing is accomplished by 3D mesh morphing, where we introduce a vertex merging technique to generate a simpler metamesh. In in-between light field mapping, an in-between object is rendered by extracting necessary fragments from input surface light fields. We also propose an acceleration technique for rendering an in-between object. Experimental results with real and synthetic data show natural and plausible morphing between objects with surface light fields. The proposed morphing technique can be used for an editing tool for 3D photography.

1. Introduction

Metamorphosis, or morphing, is a popular technique for visual effects, which generates a smooth transformation of an object into another. Several approaches have been developed for morphing of objects in different representations. 2D image morphing manipulates the images of objects to achieve compelling illusions of object transformation [3, 12, 15]. 3D object morphing deals with 3D geometry-based objects [10], where techniques were proposed for handling polygonal meshes (e.g., [9, 11]) and volumetric representations (e.g., [7, 13, 5]). Recently, a morphing technique for image-based objects was proposed [17], which generates an in-between light field from two given light fields.

In this paper, we consider morphing of two objects represented with surface light fields. A surface light field contains the colors of each object point seen from different view directions [14, 16, 4]. With a surface light field, we can represent surface properties, such as textures and specularity variations, and render accurate images of the object from arbitrary viewpoints. Recently, an object representation with a surface light field has attracted much attention because it provides a nice structure for 3D photography. Surface light field morphing can be used for an editing tool for 3D photography, which enables warping and morphing of 3D photographs.

Different representations were proposed for compression and rendering of surface light fields [14, 16, 4]. In this paper, we chose the representation proposed by Chen *et al.* [4] because it produces extremely compact data sets and allows hardware-supported rendering. In the representation, a surface light field is partitioned into vertex light fields, which are factorized into surface and view maps. In the rendering process, called *light field mapping*, the color of each triangle is determined by the fragments in surface and view maps corresponding to the triangle.

For morphing two objects with surface light fields, we should consider three problems: geometry transformation, surface light field transformation, and rendering acceleration. First, the geometry transformation can be achieved by a 3D mesh morphing technique but the resulting in-between mesh will have a more complicated structure than input. Since surface light field data should be equipped for each component of the in-between mesh, the complicated structure incurs much computational and memory overhead. Hence, it is important to reduce the complexity of an in-between mesh for efficient morphing of surface light fields. Second, the transformation of surface light fields cannot be handled by previous morphing techniques, including light field morphing [17], and will be the main problem of this paper. Finally, since an in-between object simultaneously contains the shape and surface characteristics of two input objects, it has a more complicated data set and takes more time to render than an input object. Thus, an acceleration technique will be useful for rendering an in-between object.

*Dept. of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), Pohang, 790-784, Korea

†3F, Beijing Sigma Center, No. 49, Zhichun Road, Haidian District, Beijing 100080, P R China.

In this paper, we present effective solutions for the problems of surface light field morphing. The main contributions of this paper are:

- **simpler metamesh construction:** We adopt the technique proposed by Alexa [1] to generate a metamesh that merges the geometry of input meshes. However, to produce a simpler metamesh, this paper proposes a vertex merging technique, which drastically reduces the number of vertices in a metamesh. A simpler metamesh allows efficient generation and rendering of the surface light field of an in-between object.

- **in-between light field mapping:** To handle the transformation of surface light field, this paper presents a rendering technique for the triangles of a metamesh, which we call *in-between light field mapping*. We do not explicitly construct an in-between surface light field for a metamesh. Instead, a triangle of a metamesh is rendered by dynamically extracting the necessary fragments from the input surface and view maps. With this approach, we can avoid the computational and memory overhead for constructing an in-between surface light field, while properly interpolating the surface characteristics from the source to target objects.

- **rendering acceleration:** This paper presents an approximation technique for the in-between images from surface light field morphing. The interior of an in-between object is rendered by applying in-between light field mapping to the triangles of input objects which have been transformed to in-between positions. Only the parts around the silhouettes are rendered with metamesh triangles. With this approximation, we can accelerate the generation of an in-between image without much degrading the image quality.

The remainder of this paper is organized as follows. Section 2 gives preliminaries for surface light field morphing. In Section 3, we present the overview of the proposed techniques, which will be detailed in the following three sections. Section 7 shows several morphing examples and Section 8 concludes this paper.

2. Preliminaries

2.1. Surface light field representation

Chen *et al.* proposed a compact representation and a hardware-supported rendering technique for a surface light field [4]. In the representation, a surface light field is partitioned into a set of vertex light fields. Each vertex light field $f(r, s, \theta, \phi)$ is decomposed into several surface maps $g_k[r, s]$ and view maps $h_k[\theta, \phi]$ which satisfy

$$f(r, s, \theta, \phi) \approx \sum_{k=1}^{K} g_k[r, s] h_k[\theta, \phi]. \quad (1)$$

In Eq. (1), the first pair (r, s) indicates a surface location, the second pair (θ, ϕ) indicates a viewing direction, and k is the number of approximation terms.

With the representation, an object triangle is rendered by blending three vertex light fields at the vertices of the triangle. To render a vertex light field at a vertex, fragments are extracted from the surface and view maps, $g_k[r, s]$ and $h_k[\theta, \phi]$, and multiplied pixel-by-pixel by multitexturing. The results of fragments from $g_k[r, s]$ and $h_k[\theta, \phi]$, $k = 1, \ldots, K$, are added to determine the final color provided by a vertex light field.

2.2. Mesh morphing

Mesh morphing techniques based on polygonal representations generally consist of two steps: correspondence establishment and geometry interpolation [10]. In the first step, the vertices and edges of the source and target meshes are embedded onto a common domain such as a sphere [9, 1], 2D polygon [8, 2, 6], and base mesh [11]. Then, a metamesh is created by merging the embedded vertex and edge sets on the common domain. For each vertex in the metamesh, the source and target positions are determined by mapping the vertex onto the surfaces of the source and target meshes, respectively. In the interpolation step, an in-between mesh is generated by interpolating the vertices in the metamesh between the source and target positions.

3. Overview

We first introduce the notations used in this paper. The source and target objects with surface light fields are denoted by SLF^s and SLF^t, respectively. The source object SLF^s consists of a light field map L^s and a 3D mesh M^s. Similarly, the light field and the 3D mesh of SLF^t are denoted by L^t and M^t, respectively. In this paper, we assume that M^s and M^t are 2-manifold triangular meshes. An in-between object, its light field, and its 3D mesh are denoted by SLF^i, L^i, and M^i, respectively. For the blending rate α, 0 and 1 imply the source and target objects, respectively. We represent SLF^i with the rate α as SLF^i_α, which implies $SLF^i_0 = SLF^s$ and $SLF^i_1 = SLF^t$. When α changes from 0 to 1, SLF^i_α should smoothly transform from SLF^s to SLF^t.

The overall process for morphing two objects with surface light fields consists of two main components, geometry morphing and in-between light field mapping, as illustrated in Figure 1.

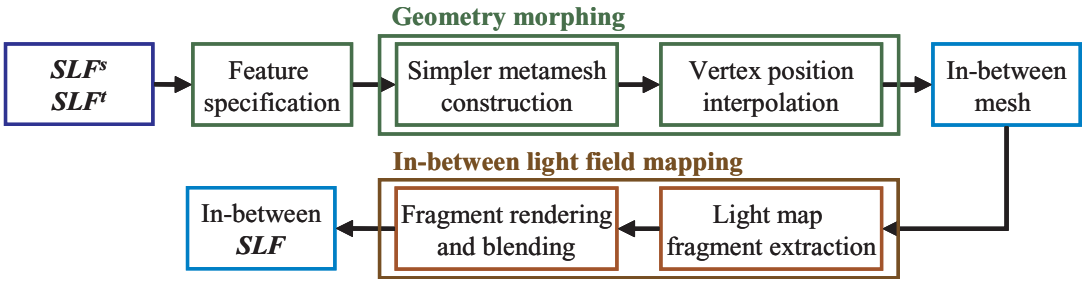


Figure 1. Overall process

- **geometry morphing**: We first construct a metamesh from input meshes M^s and M^t with the feature correspondence specified by the user. In the construction, we perform vertex merging to reduce the number of vertices in the metamesh. To obtain an in-between mesh M^i_α, the vertex positions of the metamesh are determined by interpolating those of M^s and M^t with α. Note that the metamesh construction is needed only once for different α values if the feature correspondence is not changed.

- **in-between light field mapping**: To render M^i_α with the in-between light field map L^i_α, we first extract the fragments from the source light file map L^s which correspond to the triangles of M^i_α. Remind that we do not explicitly construct L^i_α. The extracted fragments are mapped onto the triangles of M^i_α by multitexturing as in [4]. Similarly, fragments are extracted from the target light field map L^t and mapped onto M^i_α. The colors of triangles of M^i_α are finally determined by blending the values from L^s and L^t with the rate α.

In the rendering process of an in-between object SLF^i, fragment extraction from given light field maps and multitexturing are performed for each triangle of M^i. The rendering process may take much time if M^i has much larger number of triangles than M^s and M^t, despite of the vertex merging. In that case, we can accelerate the process by approximating the images of SLF^i with those of transformed SLF^s and SLF^t.

4. Geometry Morphing

4.1. In-between mesh generation

To generate a plausible and desired shape of an in-between mesh M^i, the correspondence of features should be established between the source and target meshes, M^s and M^t. In this paper, feature correspondence is specified by pairs of vertices on M^s and M^t. Figure 2 shows an example. Several points on input meshes are selected by the user, where the same colored points on M^s and M^t correspond to each other.

After the feature correspondence between M^s and M^t has been specified, we construct a metamesh by merging the vertices and edges of M^s and M^t. In this paper, we adopt the metamesh construction technique proposed by Alexa [1], which is based on spherical embedding of meshes (see Figure 2).

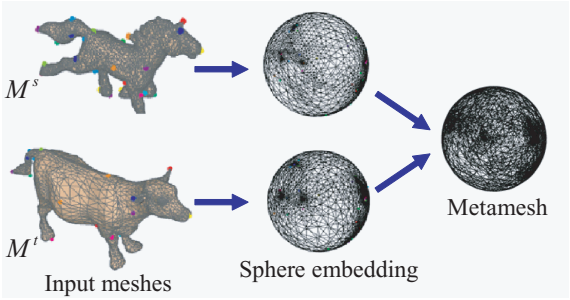


Figure 2. Metamesh construction

Regardless of the blending rate α, M^i_α has the same connectivity as the metamesh constructed from M^s and M^t. Hence, to obtain M^i_α for a given α, it is sufficient to interpolate the vertex positions of the metamesh from M^s to M^t. In this paper, the position p^i_α of a vertex in M^i_α is determined by linearly interpolating the corresponding positions p^s in M^s and p^t in M^t; that is,

$$p^i_\alpha = (1-\alpha)\cdot p^s + \alpha \cdot p^t. \quad (2)$$

4.2. Vertex merging

The vertex set of a metamesh contains the vertices of the source and target meshes, M^s and M^t, and the intersection points of the edges in the embedding of M^s and M^t onto the common spherical domain. The edges and faces in M^s and M^t may be partitioned into several pieces on a metamesh. Hence, a metamesh usually has much more complicated structure than M^s and M^t, which increases the

required memory and rendering time for an in-between object SLF^i.

Let v^s and v^t denote the vertices of M^s and M^t embedded onto the common spherical domain, respectively. To reduce the complexity of a metamesh, we merge the pairs of vertices (v^s, v^t) which are close to each other before we compute the intersections of edges on the common spherical domain. The detailed steps of the vertex merging are as follows.

1. For each target vertex v^t, we find the source triangle f^s to which v^t belongs on the common spherical domain (see Figure 3). We make three vertex pairs with v^t and three vertices of f^s. Each vertex pair is inserted into a priority queue with the distance between the two vertices in the pair.

2. From the priority queue, we extract the vertex pair one by one in the increasing order of the distance. Let (v^s, v^t) be such a vertex pair. If v^s has already been merged with another vertex $v^{t'}$, we discard the vertex pair because $v^{t'}$ is closer to v^s than v^t. Otherwise, we merge v^t with v^s. The vertex merging is not allowed when it causes a triangle flipping.

3. After processing all the vertex pairs in the queue, we apply the relaxation to the vertex positions on the spherical domain in the same way used for sphere embedding [1]. The relaxation step resolves the distortions introduced by the vertex merging.

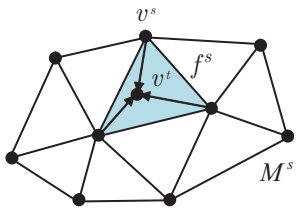


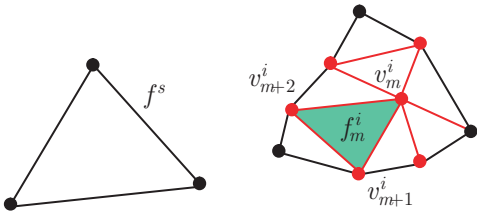
Figure 3. Vertex pairs in vertex merging

The vertex merging process considerably decreases the number of vertices in a metamesh, as demonstrated in Section 7. The proposed vertex merging technique can be applied to other metamesh construction techniques without much modification.

5. In-between Light Field Mapping

To explicitly construct the light field map L^i for an in-between object SLF^i, we have to compute the vertex light field for each vertex v^i of the mesh M^i. The vertex light field of v^i can be obtained by transforming and blending those of the vertices in M^s and M^t which are adjacent to v^i in the spherical embedding. However, M^i has a larger number of vertices than M^s and M^t, and the vertex light field of v^i should be changed with the blending rate α. Hence, explicit construction of L^i for SLF^i is inefficient in terms of the required memory and computation.

In this paper, instead of constructing L^i, we render the triangles in M^i by extracting and blending the necessary fragments from the source and target light field maps, L^s and L^t. Consider a triangle f^s in M^s which is partitioned into several triangles in M^i (see Figure 4). To render a triangle f^i_m in M^i, we first extract the fragments from the surface and view maps on f^s which correspond to f^i_m in the spherical embedding. We can then obtain the color of f^i_m given by L^s with Eq. (1). Similarly, we can derive the color of f^i_m given by L^t from the surface and view maps on the face of M^t which contains f^i_m. The final color of f^i_m is determined by blending those given by L^s and L^t.



(a) source polygon (b) in-between polygon

Figure 4. Polygons in input and in-between meshes

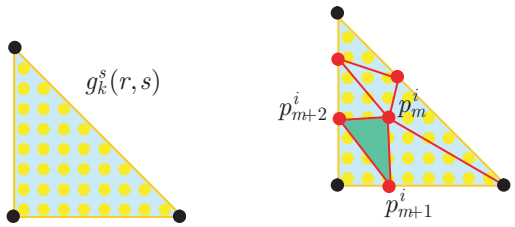
In the remainder of this section, we explain the details of the fragment extraction from the surface and view maps on f^s. The same technique can be applied to the extraction from a target face.

5.1. Surface map fragment extraction

Let $g_k[r, s]$ be a surface map on the triangle f^s in M^s (see Figure 5). The fragment of $g_k[r, s]$ corresponding to the face f^i_m in M^i is determined by the position of f^i_m in f_s on the spherical embedding. We can easily obtain the fragment by using the barycentric coordinates of the vertices of f^i_m computed with respect to f^s. In Figure 5(b), p^i_m, p^i_{m+1}, and p^i_{m+2} are the positions on $g_k[r, s]$ determined by the barycentric coordinates, and the green region is the fragment extracted for f^i_m. Note that the extracted fragment does not change regardless of the blending rate α and the rendering viewpoint.

5.2. View map fragment extraction

Contrary to the case of a surface map, the fragment extracted from a view map changes with the viewpoint. In

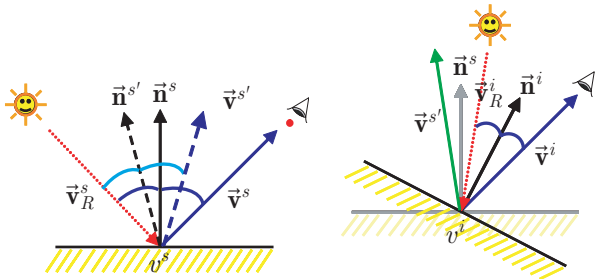


(a) source surface map (b) in-between surface map

Figure 5. Surface map fragment extraction

addition, the blending rate α has influence on the extracted fragment. For different values of α, the vertex normals of an in-between mesh M^i vary due to the shape change. Then, the color of a vertex in M^i viewed from a viewpoint also changes with the vertex normal. Hence, the view map fragment extracted for a triangle f^i in M^i is determined by the vertex normals of f^i and the viewpoint.

Let $h_k[\theta,\phi]$ be a view map on the triangle f^s in M^s which contains the triangle f^i in M^i on the spherical embedding. For a view direction $\vec{\mathbf{v}}^s = (\theta_m, \phi_m)$, $h_k[\theta_m, \phi_m]$ contains the incoming radiance reflected through the normal vector $\vec{\mathbf{n}}^s$ (see Figure 6(a)). With the assumption of a reflective BRDF, the incoming radiance information can be specified by the reflection vector $\vec{\mathbf{v}}_R^s$ [16]. When the normal vector changes to $\vec{\mathbf{n}}^{s'}$, the radiance is reflected to another view direction $\vec{\mathbf{v}}^{s'}$.



(a) source (b) in-between

Figure 6. View direction mapping

For a given viewpoint, the fragment extracted for f^i is designated by three points in $h_k[\theta,\phi]$ which are obtained from the three vertices of f_i. Let v^i be a vertex of f^i and $\vec{\mathbf{v}}^i$ be the view direction to v^i (see Figure 6(b)). The incoming radiance for v^i can be specified by the reflection vector $\vec{\mathbf{v}}_R^i$. In the view map $h_k[\theta,\phi]$, the radiance information is stored at the view direction $\vec{\mathbf{v}}^{s'}$, which is the reflection vector of $\vec{\mathbf{v}}_R^i$ through the normal vector $\vec{\mathbf{n}}^s$ on f^s. Hence, we use the direction $\vec{\mathbf{v}}^{s'}$ to select the point in $h_k[\theta,\phi]$ which corresponds to the vertex v^i. The points in $h_k[\theta,\phi]$ for the other vertices of f^i can be obtained in the same way. If the direction $\vec{\mathbf{v}}^{s'}$ exceeds the range of $h_k[\theta,\phi]$, we select the nearest direction in $h_k[\theta,\phi]$.

6. Rendering Acceleration

To accelerate the rendering of an in-between object SLF^i, we can blend two images generated by rendering the source and target objects, SLF^s and SLF^t. In that case, before rendering SLF^s and SLF^t, we transform the vertex positions of M^s and M^t to the corresponding positions in M^i (see Figure 7). Remind that M^s and M^t are much simpler than M^i and so SLF^s and SLF^t can be rendered much faster than SLF^i. Figure 8 shows an example with images from SLF^s and SLF^t and their blending.

Figure 7. Geometry transformation

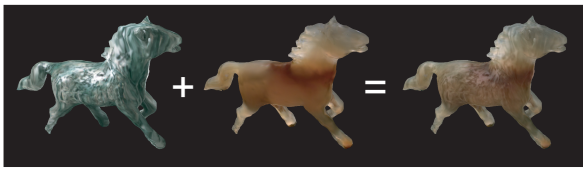
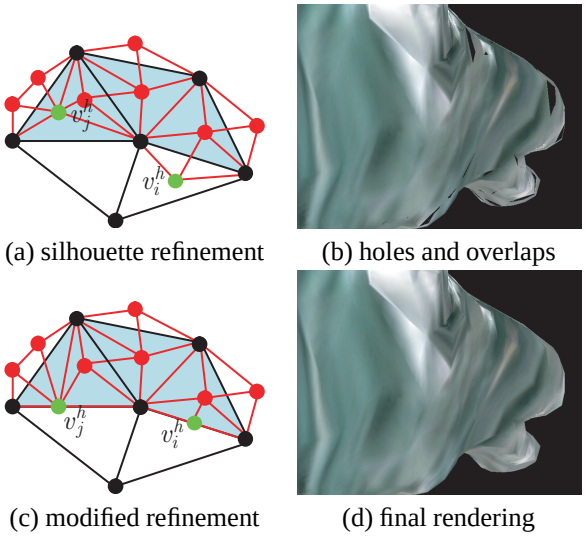


Figure 8. Approximate in-between image

When we transform the geometry of M^s, we refine the regions on M^s which are mapped to the silhouettes of the in-between mesh M^i. The refined triangles of M^s are replaced by the corresponding triangles of M^i. Figure 9(a) shows an example, where the shaded triangles of M^s are replaced by several triangles of M^i shown with red edges. The silhouette refinement is also performed on M^t when it is transformed. This refinement is needed because vertex position changes are not enough for transforming M^s and M^t to have aligned silhouettes. For rendering SLF^s and SLF^t with M^s and M^t transformed by vertex position change and silhouette refinement, we apply the in-between light field mapping, described in Section 5, to the transformed geometry.

As shown in Figure 9(b), holes and overlaps may occur around the boundary between the refined and interior regions. We can remove such holes and overlaps by mapping the vertices on the refined region boundary (v_i^h and v_j^h in Figure 9(a)) onto the edges on the interior region boundary, as illustrated in Figure 9(c). Figure 9(d) shows the final result with the modified silhouette refinement, where artifacts have been appropriately removed.



(a) silhouette refinement (b) holes and overlaps

(c) modified refinement (d) final rendering

Figure 9. Silhouette refinement

7. Experimental Results

We have implemented the proposed surface light field morphing technique on a Pentium IV 2.0GHz PC. In this section, we show several morphing examples for the objects with real or synthetic surface light fields. All synthetic data used in this paper were rendered by MaYa^{TM}.

Figures 10 through 14 show morphing examples, where the horse and turtle models are equipped with real surface light fields and all the others have synthetic data. In the examples, we can see that the surface characteristics of objects, as well as the geometry, are smoothly transformed in the morphing sequence.

Figures 15 shows an example of texture transfer [17]. In the example, the surface light field of a turtle in Figure 11 is transferred to the 3D mesh of bunny in Figure 11. The texture transfer generates a plausible object with realistic surface characteristics.

For all morphing examples in this paper, feature specification could be completed in 10 to 20 minutes. The metamesh construction takes few minutes on a Pentium IV 2.0GHz PC. Table 1 shows other statistics of the examples. The number of specified feature points increases when the shapes of input objects are much different. With the vertex merging technique, the number of triangles in a simpler metamesh is reduced to about a half of the original metamesh. We could interactively render an in-between object for all morphing examples. The rendering acceleration technique increases the frame rates about twice while it generates approximate images of an in-between object.

8. Conclusion

In this paper, we presented a morphing technique for two objects represented with surface light fields. The technique consists of geometry morphing and in-between light field mapping. Geometry morphing is accomplished by 3D mesh morphing using a metamesh. We introduced the vertex merging technique that generates a simpler metamesh. In in-between light field mapping, an in-between object is rendered by extracting necessary fragments from source and target surface light fields. We also proposed a rendering acceleration technique to speed up the image generation for an in-between object. Experimental results with real and synthetic data show that the proposed technique produces natural and plausible morphing between objects with surface light fields.

Acknowledgments

The authors would like to thank Radek Grzeszczuk for the help on light field map data and the implementation of light field mapping.

References

[1] M. Alexa. Merging polyhedral shapes with scattered features. *The Visual Computer*, 16(1):26–37, 2000.

[2] H. Bao and Q. Peng. Interactive 3D morphing. *Computer Graphics Forum (Proc. of Eurographics '98)*, 17(3):23–30, 1998.

[3] T. Beier and S. Neely. Feature-based image metamorphosis. *ACM Computer Graphics (Proc. of SIGGRAPH '92)*, 26(2):35–42, 1992.

[4] W.-C. Chen, J.-Y. Bouguet, M. H. Chu, and R. Grzesczuk. Light field mapping: Efficient representation and hardware rendering of surface light fields. *ACM Computer Graphics (Proc. of SIGGRAPH 2002)*, 2002.

[5] D. Cohen-Or, A. Solomovici, and D. Levin. Three-dimensional distance field metamorphosis. *ACM Transactions on Graphics*, 17(2):116–141, April 1998. ISSN 0730-0301.

[6] A. Gregory, A. State, M. Lin, D. Manocha, and M. Livingston. Feature-based surface decomposition for correspondence and morphing between polyhedra. *Proc. Computer Animation '98*, page To appear, 1998. IEEE Computer Society Press.

[7] J. F. Hughes. Scheduled fourier volume morphing. *ACM Computer Graphics (Proc. of SIGGRAPH '92)*, pages 43–46, 1992.

[8] T. Kanai, H. Suzuki, and F. Kimura. 3D geometric metamorphosis based on harmonic map. In *Proc. of Pacific Graphics '97*, pages 97–104, 1997.

[9] J. R. Kent, R. E. Parent, and W. E. Carlson. Establishing correspondences by topological merging: a new approach to 3-d shape transformation. *ACM Computer Graphics (Proc. of SIGGRAPH '92)*, 26(2):47–54, 1992.

models	feature points	# triangle				rendering speed (fps)	
		M^s	M^t	metamesh	simpler metamesh	in-between	approximation
horse and cow	34	7,002	5,150	49,168	22,968	1.5∼2	4∼5
turtle and bunny	45	3,704	5,026	34,564	17,550	2∼2.5	5∼7
moai and rabbit	15	4,976	4,004	37,132	16,716	2∼2.5	6∼9
rabbit and bunny	28	4,004	5,026	39,738	19,736	1.5∼2.5	5∼6
pawn and rook	18	2,256	2,400	18,754	6,516	5∼6	15∼17

Table 1. Statistics: The rendering speed is measured on a Pentium IV 2.0GHz PC.

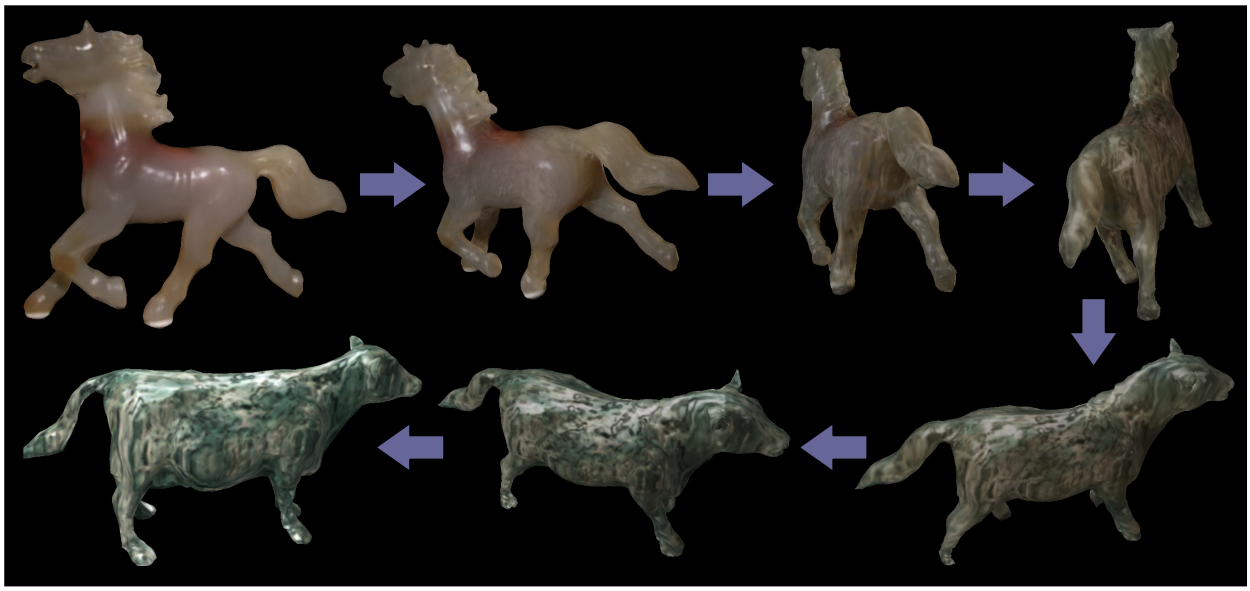


Figure 10. Morphing from a horse to a cow

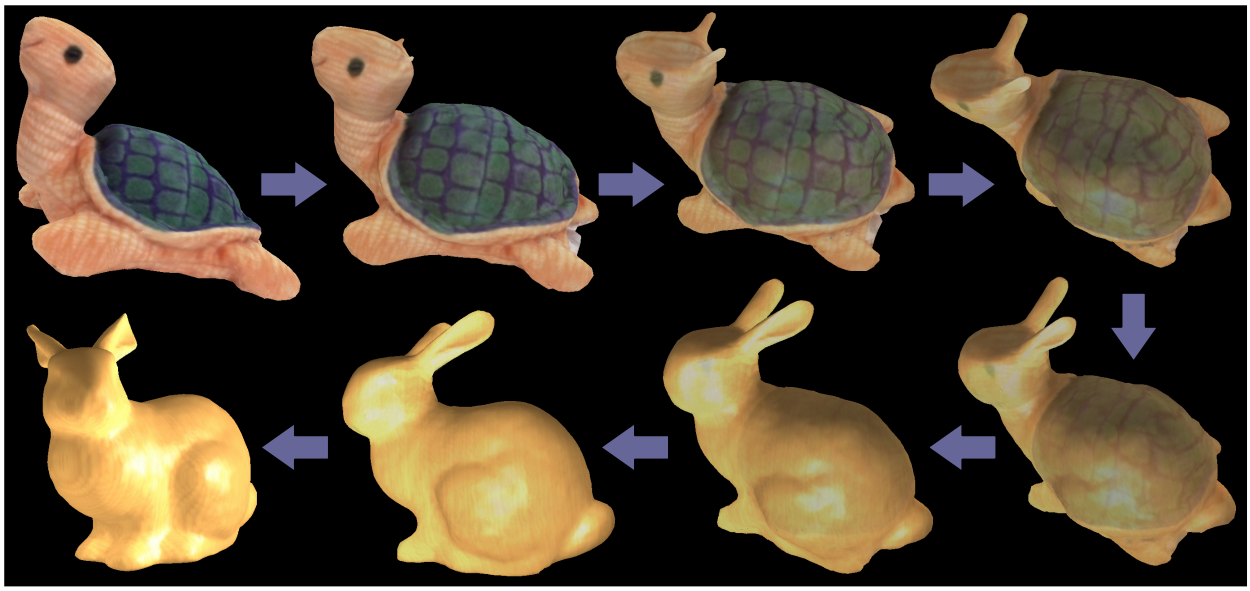


Figure 11. Morphing from a turtle to bunny

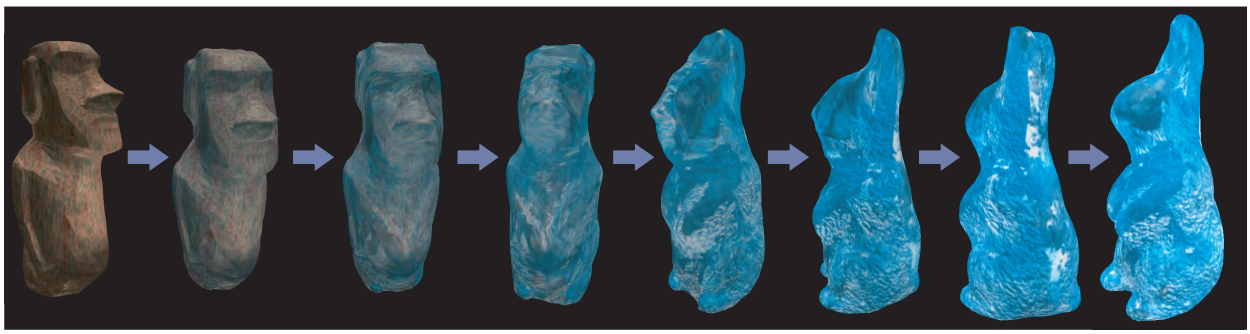


Figure 12. Morphing from a moai to rabbit

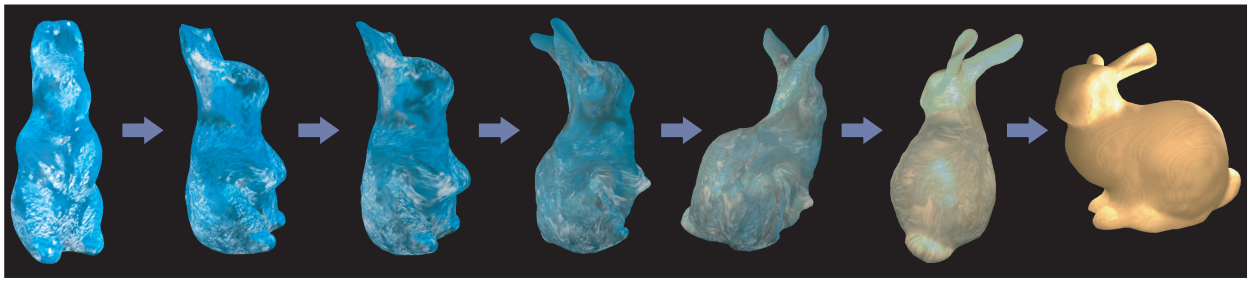


Figure 13. Morphing from a rabbit to bunny

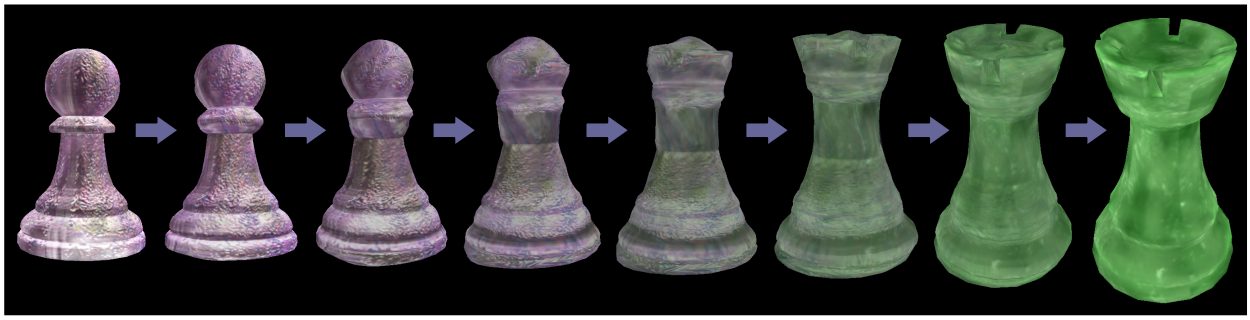


Figure 14. Morphing from a pawn to a rook

Figure 15. Texture transfer from a turtle to bunny

[10] F. Lazarus and A. Verroust. Three dimensional metamorphosis: A survey. *The Visual Computer*, 14(8/9):373–389, 1998.

[11] A. W. Lee, D. Dobkin, W. Sweldens, and P. Schröder. Multiresolution mesh morphing. *ACM Computer Graphics (Proc. of SIGGRAPH '99)*, 1999.

[12] S.-Y. Lee, K.-Y. Chwa, S. Y. Shin, and G. Wolberg. Image metamorphosis using snakes and free-form deformations. *ACM Computer Graphics (Proc. of SIGGRAPH '95)*, pages 439–448, 1995.

[13] A. Lerios, C. D. Garfinkle, and M. Levoy. Feature-based volume metamorphosis. *ACM Computer Graphics (Proc. of SIGGRAPH '95)*, pages 449–456, 1995.

[14] G. S. P. Miller, S. M. Rubin, and D. Ponceleon. Lazy decompression of surface light fields for precomputed global illumination. In *Eurographics Rendering Workshop 1998*, pages 281–292, June 1998.

[15] G. Wolberg. Image morphing: a survey. *The Visual Computer*, 14(8/9):360–372, 1998.

[16] D. N. Wood, D. I. Azuma, K. Aldinger, B. Curless, T. Duchamp, D. H. Salesin, and W. Stuetzle. Surface light fields for 3d photography. *ACM Computer Graphics (Proc. of SIGGRAPH 2000)*, 2000.

[17] Z. Zhang, L. Wang, B. Guo, and H.-Y. Shum. Feature-based light field morphing. *ACM Computer Graphics (Proc. of SIGGRAPH 2002)*, 2002.