SIAM J. IMAGING SCIENCES
Vol. 6, No. 4, pp. 1880–1902

Texture Map and Video Compression Using Beltrami Representation*

Lok Ming Lui†, Ka Chun Lam†, Tsz Wai Wong‡, and Xianfeng Gu§

Abstract. Surface parameterizations and registrations are important in computer graphics and imaging, where 1-1 correspondences between meshes are computed. In practice, surface maps are usually represented and stored as three-dimensional coordinates each vertex is mapped to, which often requires lots of memory. This causes inconvenience in data transmission and data storage. To tackle this problem, we propose an effective algorithm for compressing surface homeomorphisms using Fourier approximation of the Beltrami representation. The Beltrami representation is a complex-valued function defined on triangular faces of the surface mesh with supreme norm strictly less than 1. Under suitable normalization, there is a 1-1 correspondence between the set of surface homeomorphisms and the set of Beltrami representations. Hence, every bijective surface map is associated with a unique Beltrami representation. Conversely, given a Beltrami representation, the corresponding bijective surface map can be exactly reconstructed using the linear Beltrami solver introduced in this paper. Using the Beltrami representation, the surface homeomorphism can be easily compressed by Fourier approximation, without distorting the bijectivity of the map. The storage requirement can be effectively reduced, which is useful for many practical problems in computer graphics and imaging. In this paper, we propose applying the algorithm to texture map compression and video compression. With our proposed algorithm, the storage requirement for the texture properties of a textured surface can be significantly reduced. Our algorithm can further be applied to compressing motion vector fields for video compression, which effectively improves the compression ratio.

Key words. Beltrami representation, registration, linear Beltrami solver, texture map compression, video compression, motion vector compression

AMS subject classifications. 65D18, 68U10, 68W05

DOI. 10.1137/120866129

1. Introduction.

Surface registration and parameterization are important processes in computer graphics and imaging, where 1-1 correspondences between meshes are computed. For example, in video compression, registrations between image frames are necessary to capture the deformation of objects in images [42, 47]. Also, in computer graphics, surface parameterizations are needed for texture mapping [11, 19]. There are many different applications and approaches for surface registration and parameterization. A commonly used method is to find surface maps satisfying certain constraints, such as matching landmarks, matching intensity or minimizing distortions [7, 54]. Surface maps computed from these processes can

*Received by the editors February 15, 2012; accepted for publication (in revised form) July 5, 2013; published electronically October 8, 2013.
http://www.siam.org/journals/siims/6-4/86612.html

†Department of Mathematics, The Chinese University of Hong Kong, Shatin, Hong Kong (lmlui@math.cuhk.edu.hk, kclam@math.cuhk.edu.hk). The first author's research was supported by the RGC GRF CUHK (project ID: 401811) and the CUHK Direct Grant (project ID: 2060413).

‡Department of Mathematics, University of California at Irvine, Irvine, CA 92697-3875 (tszww@uci.edu).
§Department of Computer Science, State University of New York at Stony Brook, Stony Brook, NY 11794-4400 (gu@cs.stonybrook.edu).

be highly convoluted and are usually represented and stored as three-dimensional (3D) coordinate functions in $\mathbb{R}^3$. A huge amount of storage is therefore required, especially when a large set of fine surface meshes is to be processed. For instance, in computer graphics, mesh parameterization that maps each vertex of the surface to a two-dimensional (2D) position on an image is required for texture mapping. In order to have a high-quality textured mesh, mesh parameterizations of high resolution are necessary. Usually, a great amount of memory and bandwidth are needed to store and transmit the data of the surface map, which causes a great deal of inconvenience. Nevertheless, very little work has been done on the compression of bijective surface maps. This motivates us to look for a compression scheme for surface homeomorphisms, which can significantly reduce the storage requirement.

In this work, we propose an effective algorithm for compressing surface homeomorphisms using the Beltrami representation. The Beltrami representation is a complex-valued function defined on triangular faces of the surface mesh with supreme norm strictly less than 1. It measures the local conformality distortion of the surface map. Every surface map is associated with a unique Beltrami representation. According to the quasi-conformal Teichmüller theory, under suitable normalization, there is a 1-1 correspondence between the set of surface homeomorphisms and the set of Beltrami representations. In other words, every surface map can be represented by a unique Beltrami representation. Conversely, given a Beltrami representation, one can reconstruct the unique surface map associated with it. In this paper, we propose an algorithm called the *linear Beltrami solver* to reconstruct the surface map associated with a given Beltrami representation. Using the Beltrami representation, 1/3 of the required storage space for a bijective surface map is saved. Furthermore, the Beltrami representation has very few constraints. The only notable constraint is that its supreme norm has to be strictly less than 1. It does not have any requirements for injectivity or surjectivity. This allows us to further compress the Beltrami representation using Fourier approximations, without distorting the bijectivity of the map. The memory required can then be significantly reduced. However, Fourier compression is not possible for other representations such as 3D coordinate functions, as the bijective property (1-1 and onto) of the resulting maps cannot be guaranteed (see Figure 4).

Our proposed algorithm for surface map compression can be practically applied to problems in computer graphics and imaging. In this paper, we propose applying the algorithm to texture map compression and video compression. With our proposed algorithm, the storage requirement for the texture properties of a textured surface can be significantly reduced. Our algorithm can further be applied to compressing motion vector fields for video compression, which effectively improves the compression ratio.

In short, the contribution of this paper is threefold: 1. we propose a compression algorithm for surface homeomorphisms using the Fourier approximation of the Beltrami representation; 2. we propose the linear Beltrami solver to exactly reconstruct a surface map from its associated Beltrami representation; and 3. we apply the proposed algorithm to texture map compression and video compression, which significantly reduces the storage requirement.

This paper is laid out as follows. In section 2, we describe the relevant works closely related to this paper. In section 3, we describe some basic mathematical concepts related to our algorithms. In section 4, we describe in detail the main algorithm we use to compress bijective surface maps with their Beltrami representations. We also describe how surface

maps can be efficiently and accurately reconstructed. Algorithms for texture map and video compression are also presented. Details of the numerical implementation are included in section 6. In section 7, experimental results are reported to show the feasibility of our proposed algorithms.

2. Related works. In this section, we give an overview of the previous works mostly related to our paper.

Surface parameterization/registration. Surface registration and surface parameterization have been extensively studied. A variety of approaches have been proposed to find meaningful bijective maps between images or surfaces. For image registration, different intensity-based registration algorithms have been developed [1, 2, 9, 12, 46], which aim to obtain 1-1 correspondences between images based on image intensity. Landmark-based registration approaches have also been studied [4, 6, 26, 49], which compute registrations that match landmark features consistently. As for surface registration or parameterization, different methods have been invented, such as conformal approaches [16, 18, 21, 32], curvature matching approaches [37, 48], LDDMM approaches [44], and so on. Various landmark matching surface registration algorithms have also been developed [34, 35, 45].

Texture mapping. The technique of texture mapping has been extensively employed on rendering 3D graphics in animation and video gaming. The basic idea of it is to map an image onto a given surface so as to increase the realism of the 3D model [13, 19]. The problem of finding a suitable parameterization for texturing a polygonal mesh has been widely studied [5, 29, 31, 38, 51]. Besides, the memory for texture properties contributes a significant portion of the total file size of a textured mesh. Texture map compression is therefore necessary. Recently, much attention has been focused on compressing texture images while preserving the quality of the textured 3D model [3, 23, 50]. Furthermore, to overcome the bandwidth problem in 3D rendering, mesh compression algorithms, which reduce the memory for mesh geometry and mesh connectivity, have been proposed [10, 27, 41, 43]. However, compression of texture coordinates has received less attention. Isenburg and Snoeyink [24] proposed an algorithm in which texture coordinates are predicted from mesh vertices through the parallelogram rule. However, this algorithm reduces only half of the storage requirement and depends greatly on the similarities between meshes and the texture map.

Video compression. Video compression has developed rapidly over the last decades. Many techniques and algorithms for video compression have been proposed [20, 47]. The basic idea to achieve the compression is to remove the temporal and spatial redundancies existing in video sequences. The first generation of video compression involves techniques for intraframe coding and simple interframe coding [15]. Motion-compensated predictive coding was later proposed, which has been exploited in all recent video coding standards, such as MPEG-2, MPEG-4, or H.264. The major component in motion-compensated coding is the motion vector (MV) estimation. Various techniques for MV estimation have been proposed [25, 52, 53].

Surface map compression. Compression of mappings has also been studied. Chai et al. [8] proposed the depth map compression algorithm by encoding mappings as simplified triangular meshes. Lewis and English [33] described a technique for compressing surface potential mapping data using transform techniques. All these methods deal with the compression of real-valued functions defined on 2D domains. For vector-valued functions, Stachera

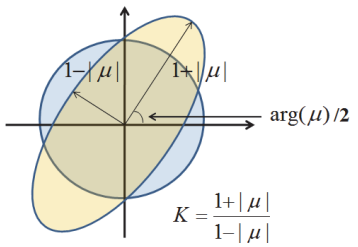

Figure 1. *Illustration of how the Beltrami coefficient measures the conformality distortion of a quasi-conformal map.*

and Rokita [40] developed an algorithm to compress normal maps by decomposing them in the frequency domain. Ioup, Gendron, and Lohrenz [22] also proposed compressing vector map data in the frequency domain. Kolesnikov and Akimov [28] proposed an algorithm for distortion-constrained compression of vector maps, based on optimal polygonal approximations and dynamic quantizations of vector data. Furthermore, surface parameterization can also be represented by a triangular mesh of the parameter domain. Under this framework, compression of the parameterization can be done through mesh compression, such as the spectral compression method [27] and the parallelogram rule for predicting vertex positions [43]. All these methods do not deal with preserving bijective maps between surfaces. The bijectivity (1-1, onto) of the maps can be easily lost due to lossy compression.

Compression of surface registrations that preserves the bijectivity was preliminary studied by Lui et al. [36]. In that work, Beltrami coefficients (BCs) defined on every vertex of the mesh were proposed to represent bijective surface maps. BCs can approximate the associated surface map well only when the triangulation is regular. Also, Beltrami holomorphic flow (BHF) was used to iteratively reconstruct surface maps from their BCs. Integration has to be computed in each iteration, which causes inefficiency in many practical applications. BHF was further applied for the optimization of surface registrations in [37].

3. Mathematical background.

In this section, we describe some basic mathematical concepts related to our algorithms. For details, we refer the readers to [14].

Quasi-conformal maps are a generalization of conformal maps, which are orientation preserving homeomorphisms between Riemann surfaces with bounded conformality distortion. Mathematically, $f : \mathbb{C} \to \mathbb{C}$ is quasi-conformal provided that it satisfies the Beltrami equation

$$\frac{\partial f}{\partial \overline{z}} = \mu(z)\frac{\partial f}{\partial z} \tag{3.1}$$

for some complex-valued Lebesgue measurable function μ satisfying $||\mu||_\infty < 1$. μ is called the BC of f. The BC μ gives us all the information about the conformality of f (see Figure 1).

Theorem 3.1 (measurable Riemann mapping theorem). *Suppose $\mu : \mathbb{C} \to \mathbb{C}$ is Lebesgue measurable and satisfies $\|\mu\|_\infty < 1$; then there is a quasi-conformal homeomorphism ϕ from $\mathbb{C}$ onto itself, which is in the Sobolev space $W^{1,2}(\mathbb{C})$ and satisfies the Beltrami equation (3.1) in the distribution sense. Furthermore, by fixing 0, 1, and ∞, the associated quasi-conformal homeomorphism ϕ is uniquely determined.*

By reflection, the above theorem can be further extended to BCs defined on the unit disk $\mathbb{D}$ [14, 37].

Theorem 3.2. *Suppose $\mu : \mathbb{D} \to \mathbb{C}$ is Lebesgue measurable and satisfies $\|\mu\|_\infty < 1$; then there is a quasi-conformal homeomorphism ϕ from the unit disk to itself, which is in the Sobolev space $W^{1,2}(\Omega)$ and satisfies the Beltrami equation (3.1) in the distribution sense. Furthermore, by fixing 0 and 1, the associated quasi-conformal homeomorphism ϕ is uniquely determined.*

Given an orientation preserving homeomorphism ϕ, we can find the corresponding BCs from the Beltrami equation:

$$\mu_\phi = \frac{\partial \phi}{\partial \bar{z}} \Big/ \frac{\partial \phi}{\partial z}. \tag{3.2}$$

The Jacobian J of ϕ is related to μ_ϕ as follows:

$$J(\phi) = \left| \frac{\partial \phi}{\partial z} \right|^2 \left(1 - |\mu_\phi|^2 \right). \tag{3.3}$$

Since ϕ is an orientation preserving homeomorphism, $J(\phi) > 0$ and $|\mu_\phi < 1$ everywhere. Hence, we must have $\|\mu_\phi\|_\infty < 1$. Furthermore, Theorems 3.1 and 3.2 suggest that under suitable normalization, every μ with $\|\mu\|_\infty < 1$ is associated with a unique homeomorphism. We can therefore conclude that there is a 1-1 correspondence between the set of all quasi-conformal homeomorphisms and the set of all BCs with supreme norm less than 1. In other words, a homeomorphism from $\mathbb{C}$ or $\mathbb{D}$ onto itself can be uniquely determined by its associated BC. Theorems 3.1 and 3.2 can also be extended to homeomorphisms between Riemann surfaces [37]. For example, by giving four points of correspondence on the boundaries of two open simply connected Riemann surfaces, the surface homeomorphism can be uniquely determined by its Beltrami coefficient (See Figure 2(A)). These observations play important roles for the main algorithms proposed in this paper.

4. Main algorithms.

In this section, we describe in detail the main algorithms proposed in the paper to compress the surface homeomorphisms using Beltrami representation. We first introduce the Beltrami representation for the bijective surface map. Second, we present a fast reconstruction algorithm of the surface map from its Beltrami representation. We then introduce the Fourier approximation of the Beltrami representation for the compression of surface homeomorphisms. Based on these techniques, we introduce new storage formats to compress texture maps and videos.

4.1. Beltrami representation for bijective surface maps.

Surface registration and parameterization are commonly represented by 3D coordinate functions. This representation requires lots of storage space and is difficult to manipulate. For example, the Jacobian of the 3D coordinate functions has to be strictly greater than zero in order to preserve the 1-1 correspondence of the surface maps. Enforcing this constraint adds extra difficulty in manipulating or compressing surface maps. It is therefore important to have a simpler representation with as few constraints as possible.

According to quasi-conformal Teichmüller theory, a surface homeomorphism can be uniquely determined by its BC by fixing few points of correspondence (two points if the surfaces are

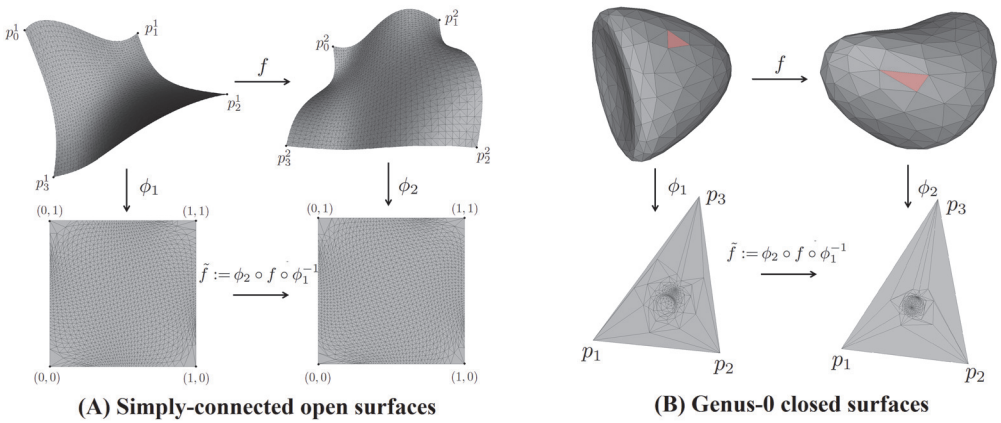

Figure 2. *Illustration of how Beltrami representations for homeomorphisms between meshes can be computed.* (A) *shows the case of a homeomorphism between simply connected open meshes. The two meshes are mapped to a unit square by harmonic parameterizations.* (B) *shows the case of a homeomorphism between genus-0 closed surface meshes. The two meshes are parameterized onto a triangle in* $\mathbb{R}^2$*, after cutting away a triangular face on each mesh.*

of disk topology and three points if the surfaces are genus-0 closed surfaces). It is similar to the fact that a conformal map between two unit spheres is uniquely determined by fixing three points of correspondence. It motivates us to represent the surface homeomorphism using the BC.

Suppose M_1 and M_2 be two simply connected open surfaces. Let $f : M_1 \to M_2$ be an orientation preserving homeomorphism between M_1 and M_2. Suppose $\{p_i \in \partial M_1\}_{i=1}^4$ and $\{q_i \in \partial M_2\}_{i=1}^4$ are the two sets of corresponding points between M_1 and M_2, where ∂M_1 and ∂M_2 are the boundaries of M_1 and M_2, respectively. We first parameterize M_1 and M_2 onto a unit square $R = [0,1] \times [0,1]$ by harmonic maps: $\phi_1 : M_1 \to R$ and $\phi_2 : M_2 \to R$ such that $\phi_1(p_i)$ and $\phi_2(q_1)$ are the four corners of the unit square R. In case M_1 and M_2 are two genus-0 close surfaces, we can also parameterize them by $\phi_1 : M_1 \to \mathbb{S}^2 \cong \overline{\mathbb{C}}$ and $\phi_2 : M_2 \to \mathbb{S}^2 \cong \overline{\mathbb{C}}$, respectively. For details of computing the harmonic parameterizations, please refer to [17, 39].

The bijective map $f : M_1 \to M_2$ can now be represented by the BC μ_f of the composition map $\tilde{f} = \phi_2 \circ f \circ \phi_1^{-1} : D \to D$, where D is the parameter domain in $\mathbb{R}^2$. (see Figure 2). Since ϕ_1, ϕ_2, and f are all orientation preserving homeomorphisms, $\tilde{f}$ is also an orientation preserving homeomorphism. We get $||\mu_{\phi_2 \circ f \circ \phi_1^{-1}}||_\infty < 1$. Hence, $\mu_{\tilde{f}}$ uniquely determines the surface map $\tilde{f}$. The map $f : M_1 \to M_2$ can then be reconstructed by $f := \phi_2^{-1} \circ \tilde{f} \circ \phi_1$. Hence, the complex-valued function $\mu_f := \mu_{\tilde{f}} \circ \phi_1$ uniquely determines the surface map $f : M_1 \to M_2$. We call μ_f the *Beltrami representation* of f. Given a Beltrami representation, the corresponding surface homeomorphism can be reconstructed, which will be described in the next subsection.

Representing surface homeomorhpisms by Beltrami representations is beneficial. First, 1/3 of the required storage space for a bijective surface map is saved. Second, the Beltrami representation has very few constraints. The only notable constraint is that its supreme norm

has to be strictly less than 1. It does not have any requirements of injectivity or surjectivity. This allows us to further compress the Beltrami representation using Fourier approximations, without distorting the bijectivity of the map. The memory required can then be significantly reduced.

4.2. Fast reconstruction of surface map from Beltrami representation. Given a Beltrami representation, it is important to have an efficient algorithm to reconstruct the associated quasi-conformal homeomorphism.

Suppose M_1 and M_2 are two surfaces with the same topology (either topological disks or genus-0 closed surfaces). Given a Beltrami representation μ on M_1, our goal is to reconstruct the associated homeomorphism $f : M_1 \to M_2$ between M_1 and M_2.

To reconstruct the surface homeomorphism, we first compute the harmonic parameterizations $\phi_1 : M_1 \to D$ and $\phi_2 : M_2 \to D$. We shall look for the quasi-conformal mapping $\tilde{f} : D \to D$ with BC $\mu_{\tilde{f}} := \mu \circ \phi_1^{-1}$. The desired homeomorphism f can then be reconstructed by taking the composition: $f = \phi_2^{-1} \circ \tilde{f} \circ \phi_1$.

To reconstruct the quasi-conformal mapping $\tilde{f}$, let $\tilde{f} = u + iv$, where $i = \sqrt{-1}$. From the Beltrami equation (3.1), we have

$$\mu(\tilde{f}) = \frac{(u_x - v_y) + i(v_x + u_y)}{(u_x + v_y) + i(v_x - u_y)}. \tag{4.1}$$

Let $\mu(\tilde{f}) = \rho + i\tau$. We can write v_x and v_y as linear combinations of u_x and u_y,

$$\begin{aligned} -v_y &= \alpha_1 u_x + \alpha_2 u_y; \\ v_x &= \alpha_2 u_x + \alpha_3 u_y. \end{aligned} \tag{4.2}$$

where $\alpha_1 = \frac{(\rho-1)^2+\tau^2}{1-\rho^2-\tau^2}$; $\alpha_2 = -\frac{2\tau}{1-\rho^2-\tau^2}$; $\alpha_3 = \frac{(\rho+1)^2+\tau^2}{1-\rho^2-\tau^2}$.

Similarly,

$$\begin{aligned} -u_y &= \alpha_1 v_x + \alpha_2 v_y; \\ u_x &= \alpha_2 v_x + \alpha_3 v_y. \end{aligned} \tag{4.3}$$

Since $\nabla \cdot \binom{-v_y}{v_x} = 0$, we obtain

$$\nabla \cdot \left(A \begin{pmatrix} u_x \\ u_y \end{pmatrix} \right) = 0 \ \text{ and } \ \nabla \cdot \left(A \begin{pmatrix} v_x \\ v_y \end{pmatrix} \right) = 0, \quad A = \begin{pmatrix} \alpha_1 & \alpha_2 \\ \alpha_2 & \alpha_3 \end{pmatrix}. \tag{4.4}$$

Besides, $\tilde{f}$ has to satisfy certain constraints on the boundary. For example, when M_1 is a topological disk, the parameter domain D is a unit square. In this case, the desired quasi-conformal map should satisfy

$$\begin{gathered} \tilde{f}(0) = 0; \quad \tilde{f}(1) = 1; \quad \tilde{f}(i) = i; \quad \tilde{f}(1+i) = 1+i; \\ \mathbf{Re}(\tilde{f}) = 0 \text{ on arc } [0, i]; \ \mathbf{Re}(\tilde{f}) = 1 \text{ on arc } [1, 1+i]; \\ \mathbf{Imag}(\tilde{f}) = 0 \text{ on arc } [0, 1]; \ \mathbf{Imag}(\tilde{f}) = 1 \text{ on arc } [i, 1+i]. \end{gathered} \tag{4.5}$$

Hence, by solving the partial differential equations with Dirichlet boundary conditions mentioned above, we can obtain the x-coordinate and y-coordinate functions of $\tilde{f}$. The details of the numerical implementation will be discussed in the next section.

4.3. Compression of surface homeomorphisms. With the Beltrami representation introduced above, we can further compress the surface homeomorphisms using the Fourier approximation. An important consideration is to preserve the bijectivity of the reconstructed surface map after the compression.

Under the representation by coordinate functions, the surface map cannot be easily compressed using the Fourier approximation without distorting the bijectivity. In order to preserve the bijectivity, the Jacobian of the coordinate functions has to be greater than 0. This constraint is equivalent to an inequality in the partial derivatives of the coordinate functions. Enforcing this constraint is difficult during compression, and the bijective property can be easily lost. The Beltrami representation, however, is advantageous because it does not have any requirement for the injectivity or surjectivity, making the Jacobian constraint unnecessary. The only requirement for the Beltrami representation μ is that it be a complex-valued function with supreme norm less than 1. We can therefore compress μ using the Fourier approximation without losing the bijectivity (see Figure 4).

Let M_1 and M_2 be simply connected open surfaces. Suppose μ is the Beltrami representation of the surface map $f : M_1 \to M_2$. We first parameterize M_1 to the unit square D by the parameterization $\phi_1 : M_1 \to D$. We can then obtain the BC $\mu_{\tilde{f}} := \mu \circ \phi_1^{-1}$, which is defined on parameter domain $D = [-T, T] \times [-T, T]$. $\mu_{\tilde{f}}$ can be expressed by the Fourier expansion as

$$\mu_{\tilde{f}}(x, y) = \sum_{m,n=-\infty}^{\infty} c_{m,n} e^{i\pi mx/T} e^{i\pi ny/T}, \tag{4.6}$$

where

$$c_{m,n} = \frac{1}{4\pi} \int_{-T}^{T} \int_{-T}^{T} u_{\tilde{f}}(x, y) e^{-i\pi jx/T} e^{-i\pi ky/T} dxdy.$$

In practice, we can use the fast Fourier transform (FFT) to compute the coefficients $c_{m,n}$ efficiently. One can then take fewer Fourier coefficients to approximate the BC $\mu_{\tilde{f}}$, which can significantly reduce the storage requirement. With the Fourier coefficients of the truncated Fourier series, the compressed BC defined on D and hence the compressed Beltrami representation μ^c defined on M_1 can be obtained. μ^c accurately approximates μ, and hence the quasi-conformal map associated with μ^c closely resembles f.

4.4. Compression of texture mapping. Texture mapping techniques have been extensively studied to provide realistic 3D rendering in movies, animation, and video gaming. The basic idea of texture mapping is to map a texture image onto a given surface, so as to increase the realism of the 3D model. Typically, a textured surface mesh is represented by its mesh geometry, mesh connectivity, texture map, and texture image. Usually, the memory requirement for texture properties contributes a significant portion of the total file size of the textured mesh. In addition, the demand for a higher level of realism is rising, which in turn requires a higher resolution of texture mapping for producing more detailed models. The need for a compact representation of the texture map is therefore crucial.

Using the Beltrami representation of the texture map and the reconstruction procedure mentioned in previous subsections, we introduce a new format to store the texture map for

compression. In the following, we propose an effective algorithm to encode and decode the texture map using the new storage format.

Consider a surface mesh M with texture maps $\{\rho_i : M_i \to [0,1] \times [0,1]\}_{i=1}^n$, where M_i's are patches of M. Our goal is to compress the texture mappings ρ_i. We first parameterize M_i onto a unit square D by harmonic parameterizations $\phi_i : M_i \to D$. The BC $\tilde{\mu_i}$ of $\tilde{\rho_i} := \rho \circ \phi_i^{-1}$ can then be computed. As described in section 4.1, the Beltrami representation $\mu_i := \tilde{\mu_i} \circ \phi_i$ together with the boundary map $\rho_i|_{\partial M_i}$ uniquely determines ρ_i. Using the Fourier compression scheme as described in section 4.3, the Beltrami representation μ_i can be effectively compressed. Hence, the texture map ρ_i can now be represented by (1) the Fourier coefficients $c_{j,k}^i$ in the truncated Fourier series, and (2) the texture coordinates of all boundary vertices of M_i. This significantly reduces the storage requirement for the texture mappings. Note that in the case where the texture image is of regular shape (e.g., 2D rectangle), storing the texture coordinates of the boundary vertices is unnecessary. Instead, texture coordinates of the four boundary vertices are sufficient.

The decoding algorithm is also straightforward. Given the Fourier coefficients $c_{j,k}^i$ and the texture coordinates of the boundary vertices, our goal is to reconstruct the texture map ρ_i. Using the inverse FFT applied to Fourier coefficients $c_{j,k}^i$ saved, the Beltrami representation $\mu_i : M_i \to \mathbb{C}$ can be restored. With the Beltrami representation, the texture map ρ_i can be reconstructed. More specifically, ρ_i can be computed by solving the two linear systems (4.4), subject to the boundary condition given by the texture coordinates of the boundary vertices. Texture coordinates of M_i can then be obtained.

Algorithm 4.1 (encoding of the texture coordinates).

Input : *Surface mesh M; Texture coordinate function $f : M \to D \subset [0,1] \times [0,1]$.*

Output : *Fourier coefficients $c_{j,k}$ in the truncated Fourier series; Target coordinates of the boundary vertices of $f(D)$.*

1. *Obtain the harmonic parametrization, $\phi : M \to [0,1] \times [0,1]$;*
2. *Compute the Beltrami representation μ_f of f as in section 4.1;*
3. *Compress the Beltrami representation μ_f using Fourier approximation described in section 4.3;*
4. *Store the Fourier coefficients $c_{j,k}$ in the truncated Fourier series and coordinates of the boundary vertices of $f(D)$.*

Algorithm 4.2 (decoding of the texture coordinates).

Input : *Surface mesh M; Fourier coefficients $c_{j,k}$ of the Beltrami representation; Coordinates of the boundary vertices.*

Output : *Reconstructed texture coordinates of the surface mesh M.*

1. *Apply the inverse FFT on $c_{j,k}$ to restore the Beltrami representation μ_f defined on triangular faces of M;*
2. *Parametrize the surface mesh M by the harmonic mapping ϕ to the unit square;*
3. *Reconstruct the texture coordinates by solving (4.4) subject to the boundary condition given by the texture coordinates of the boundary vertices.*

4.5. Video compression. With the development of video compression techniques, real time digital television and Internet streaming video become practical. However, resolution of

the video is limited by the channel bandwidth. Video compression is therefore an important field of research.

The basic idea of most existing algorithms is to remove the temporal and spatial redundancies existing in a video sequence. In motion compensation techniques, instead of storing every image frame in the video, the I-frame, P-frame, and B-frame are introduced. The I-frame is the reference frame, which is stored in the compressed image form. The P-frame is called the predictive frame. It is encoded as an MV field together with a residual. The MV field creates a prediction depicting how pixels in the previous frame move. This prediction is then subtracted from the original frame to obtain the residual image. The B-frame is called the bidirectional predictive frame, which is obtained from the interpolated MV fields from previous and future frames. For details, please refer to [20, 30].

On average, P-frames contribute 50% less storage than that of I-frames. However, the required storage for the MV field is still significant when considering HD videos. For example, if a maximum of a 4×4 block mode is used in a full HD $[1920 \times 1280]$ video, each P-frame requires 2.765×10^6 bits of memory to store the MV field. It thus calls for the need of compressing the MV field.

Every MV field V can be represented by the Beltrami representation μ. Using the scheme proposed in section 4.3, the MV field V can be compressed by performing the Fourier compression of μ. Storing the Fourier coefficients of the truncated Fourier series of μ_T requires much less storage than that of the MV field itself.

The reconstruction of the MV field from the Fourier coefficients $c_{j,k}$ is straightforward. We first carry out the inverse FFT to restore the BC μ. By the reconstruction procedure as described in section 4.2, the corresponding MV field V can be reconstructed.
The encoding and decoding of the compression algorithm can be summarized as follows.

Algorithm 4.3 (encoding of the P-frame).
Input : *Frame F_1 (Reference) and F_2 (P-frame)*
Output : *Coefficients $c_{j,k}$ in the truncated Fourier series*

1. *Obtain the MV field V from F_1 to F_2 in the regular grid;*
2. *Compute the Beltrami representation;*
3. *Use the algorithm described in section 4.3 to compress the Beltrami representation μ and store the coefficients $c_{j,k}$ in the truncated Fourier series.*

Algorithm 4.4 (decoding of the P-frame).
Input : *Frame F_1; Coefficients $c_{j,k}$ in the truncated Fourier series*
Output : *MV field V*

1. *Apply the inverse FFT to restore the Beltrami coefficient $\tilde{\mu}$;*
2. *Perform the reconstruction procedure to obtain V.*

5. Numerical implementation.

In this section, we describe in detail the numerical implementation of our proposed algorithms.

5.1. Computation of the Beltrami representation.

In practice, surfaces are represented by triangular meshes. Surface maps are usually approximated by piecewise linear homeomorphisms between meshes.

Suppose K_1 and K_2 are two surface meshes with the same topology (either genus-0 closed

surface meshes or simply connected open surface meshes). Since we are considering the representation of the bijective map between the two surfaces K_1 and K_2, there is a 1-1 correspondence between them. We can then assume that they have the same number of vertices and same connectivity information. Define the set of vertices of K_1 and K_2 by $V^1 = \{v_i^1\}_{i=1}^n$ and $V^2 = \{v_i^2\}_{i=1}^n$, respectively. Similarly, define the set of triangular faces of K_1 and K_2 by $F^1 = \{T_j^1\}_{j=1}^m$ and $F^2 = \{T_j^2\}_{j=1}^m$, respectively. Now, consider a piecewise linear homeomorphism $f : K_1 \to K_2$ between K_1 and K_2. We first parameterize K_1 and K_2 onto a 2D parameter domain D by harmonic maps [17]. Denote them by $\phi_1 : K_1 \to D$ and $\phi_2 : K_2 \to D$, respectively. We then have the composition mapping $\tilde{f} = \phi_2 \circ f \circ \phi_1^{-1}$.

To compute $\mu_{\tilde{f}}$, we simply need to approximate the partial derivatives at each face T. We denote them by $D_x\tilde{f}(T)$ and $D_y\tilde{f}(T)$, respectively. Note that $\tilde{f}$ is piecewise linear. The restriction of $\tilde{f}$ on each triangular face T can be written as

$$\tilde{f}|_T(x,y) = \begin{pmatrix} a_T x + b_T y + r_T \\ c_T x + d_T y + q_T \end{pmatrix}. \tag{5.1}$$

Hence, $D_x\tilde{f}(T) = a_T + ic_T$ and $D_y\tilde{f}(T) = b_T + id_T$. Now, the gradient $\nabla_T\tilde{f} := (D_x\tilde{f}(T), D_y\tilde{f}(T))^t$ on each face T can be computed by solving

$$\begin{pmatrix} \vec{v}_1 - \vec{v}_0 \\ \vec{v}_2 - \vec{v}_0 \end{pmatrix} \nabla_T \tilde{f}_i = \begin{pmatrix} \tilde{f}_i(\vec{v}_1) - \tilde{f}_i(\vec{v}_0) \\ \tilde{f}_i(\vec{v}_2) - \tilde{f}_i(\vec{v}_0) \end{pmatrix}, \tag{5.2}$$

where $[\vec{v_0}, \vec{v_1}]$ and $[\vec{v_0}, \vec{v_2}]$ are two edges on T. By solving (5.2), a_T, b_T, c_T, and d_T can be obtained. The BC $\mu_{\tilde{f}}(T)$ of the triangular face T can then be computed from the Beltrami equation (3.1) by

$$\mu_{\tilde{f}}(T) = \frac{(a_T - d_T) + i(c_T + b_T)}{(a_T + d_T) + i(c_T - b_T)}. \tag{5.3}$$

5.2. Linear Beltrami solver. Given the BC $\mu_{\tilde{f}}$, we can reconstruct the corresponding quasi-conformal mapping $\tilde{f}$ by solving (4.4). In this subsection, we propose the linear Beltrami solver to solve (4.4).

Recall that the restriction of $\tilde{f}$ on each triangular face T is linear and can be written as

$$\tilde{f}|_T(x,y) = \begin{pmatrix} a_T x + b_T y + r_T \\ c_T x + d_T y + q_T \end{pmatrix}. \tag{5.4}$$

By (4.2) and (4.3), we have

$$\begin{aligned} -d_T &= \alpha_1^T a_T + \alpha_2^T b_T; \\ c_T &= \alpha_2^T a_T + \alpha_3^T b_T \end{aligned} \tag{5.5}$$

and

$$\begin{aligned} -b_T &= \alpha_1^T c_T + \alpha_2^T d_T; \\ a_T &= \alpha_2^T c_T + \alpha_3^T d_T. \end{aligned} \tag{5.6}$$

Let $T = [v_i, v_j, v_k]$ and $w_I = \tilde{f}(v_I)$, where $I = i, j$, or k. Suppose $v_I = g_I + ih_I$ and $w_I = s_I + it_I$ $(I = i, j, k)$. Using (5.2), a_T, b_T, c_T, and d_T can be written as follows:

$$\begin{aligned} a_T &= A_i^T s_i + A_j^T s_j + A_k^T s_k;\ b_T = B_i^T s_i + B_j^T s_j + B_k^T s_k; \\ c_T &= A_i^T t_i + A_j^T t_j + A_k^T t_k;\ d_T = B_i^T t_i + B_j^T t_j + B_k^T t_k, \end{aligned} \tag{5.7}$$

where

$$\begin{aligned} &A_i^T = (h_j - h_k)/2Area(T),\ A_j^T = (h_k - h_i)/2Area(T),\ A_k^T = (h_i - h_j)/2Area(T); \\ &B_i^T = (g_k - g_j)/2Area(T),\ B_j^T = (g_i - g_k)/2Area(T),\ B_k^T = (g_j - g_i)/2Area(T). \end{aligned} \tag{5.8}$$

For each vertex v_i, let N_i be the collection of neighborhood faces attached to v_i. By careful checking, one can observe that

$$\sum_{T \in N_i} A_i^T b_T = \sum_{T \in N_i} B_i^T a_T;\ \sum_{T \in N_i} A_i^T d_T = \sum_{T \in N_i} B_i^T c_T. \tag{5.9}$$

Thus, following from (5.5) and (5.6), we have

$$\sum_{T \in N_i} \left(A_i^T [\alpha_1^T a_T + \alpha_2^T b_T] + B_i^T [\alpha_2^T a_T + \alpha_3^T b_T] \right) = 0 \tag{5.10}$$

and

$$\sum_{T \in N_i} \left(A_i^T [\alpha_1^T c_T + \alpha_2^T d_T] + B_i^T [\alpha_2^T c_T + \alpha_3^T d_T] \right) = 0 \tag{5.11}$$

for all vertices $v_i \in D$. Note that a_T and b_T can be written as a linear combination of the x-coordinates of the desired quasi-conformal map $\tilde{f}$. Hence, (5.10) gives us the linear system to solve for the x-coordinate function of $\tilde{f}$. Similarly, c_T and d_T can also be written as a linear combination of the y-coordinates of the desired quasi-conformal map $\tilde{f}$. Therefore, (5.11) gives us the linear system to solve for the y-coordinate function of $\tilde{f}$.

6. Numerical experiments.

In this section, experimental results are presented to demonstrate the effectiveness of the proposed algorithms.

6.1. Reconstruction of surface homeomorphism using linear Beltrami solver.

Given a Beltrami representation, the corresponding surface homeomorphism can be exactly computed using a linear Beltrami solver. Experimental results show that the proposed linear Beltrami solver can effectively compute the surface homeomorphism associated with a given Beltrami representation. Figure 3(A) shows the homeomorphism of the unit square and the homeomorphism of the synthetic genus-zero closed surface. Figure 3(B) shows the reconstructed homeomorphisms from their corresponding BCs. The original homeomorphisms and the reconstructed ones have no observable difference. Figure 3(C) shows the norm of their BCs.

Compared to the BHF method introduced in [36, 37], our method can compute the associated surface homeomorphism more accurately and efficiently. Table 1 shows the comparison of the computational time and error under the LBS method and the BHF method. Experimental

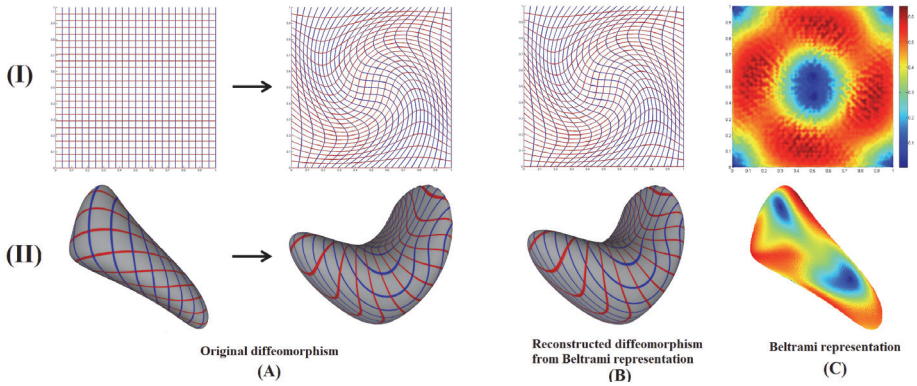

Figure 3. *Illustration of the proposed linear Beltrami solver.* (A) *shows the original homeomorphism of the unit square* (I) *and the homeomorphism between two synthetic genus-0 closed surface* (II). (B) *shows the reconstructed homeomorphisms from the Beltrami representations.* (C) *shows the norm of their Beltrami representations.*

Table 1
Comparison of the computational time and error under LBS and BHF.

	$\|\|\mu\|\|_\infty$	L1-error (LBS)	Time (LBS)	L1-error (BHF)	Time (BHF)
Map 1	0.6258	1.365×10^{-14}	0.032 s	0.0064	98.5 s
Map 2	0.8556	9.326×10^{-15}	0.028 s	0.0159	96.4 s
Map 3	0.9995	2.408×10^{-13}	0.033 s	0.0313	102.2 s

results show that the LBS method can compute the associated surface map much faster than the BHF method. Also, the accuracy is much better when using LBS. As shown in Table 1, the numerical error under LBS is much less than that under BHF. The computational efficiency of LBS allows us to apply our proposed algorithm in more practical applications that require real-time processing, such as video compression.

6.2. Compressing surface homeomorphism using Fourier approximation. Experiments have been carried out to compress the surface homeomorphisms, which are shown in Figures 3(A)(I) and 3(A)(II). Results show that the proposed compression algorithm is stable and effective in reducing the storage requirement of bijective surface maps. Figure 4(A)(left) shows the reconstructed homeomorphism from the compressed Beltrami representation. The reconstructed map closely resembles the original one (see Figure 3(A)(I)). Figure 4(A)(right) shows the reconstructed map from the compressed coordinate functions. Note that the bijectivity is completely disrupted.

The proposed compression scheme can also be applied to compressing surface homeomorphisms. Figure 4(B)(left) shows the reconstructed surface map from the compressed Beltrami representation, which closely resembles the original map (see Figure 3(A)(II)). Figure 4(B)(right) shows the the reconstructed surface map from the compressed coordinate functions. Again, the bijectivity of the surface map cannot be preserved after the compression.

6.3. Texture map compression. To examine the performance of the proposed texture map compression algorithm, experiments have been carried out on different 3D surface meshes.

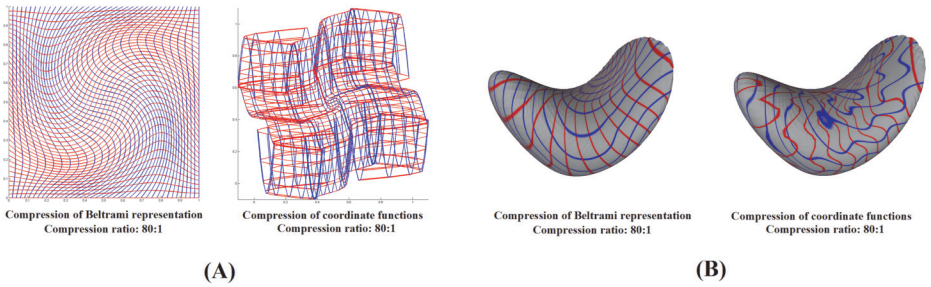

Figure 4. *Compression of the bijective map.* (A) *and* (B) *show the reconstructed map from the compressed Beltrami representation and the compressed coordinate functions of examples* (I) *and* (II) *in Figure* 3*, respectively. The bijectivity of the reconstructed maps from the compressed coordinate functions is completely disrupted.*

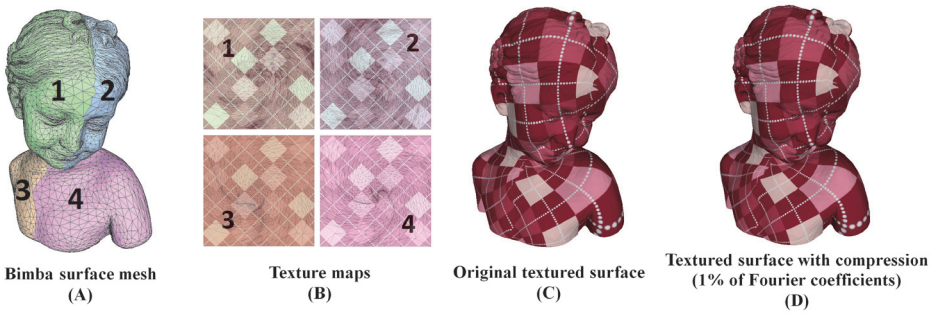

Figure 5. *Illustration of the proposed compression scheme for texture maps.*

Figure 5(A) shows the surface mesh of a bimba surface. Its texture maps and texture images are as shown in Figure 5(B). There are a total of four texture maps and texture images with regular shapes. The original textured surface is as shown in Figure 5(C). We apply the proposed compression algorithm to this example (with 1% of Fourier coefficients stored). Figure 5(D) shows the reconstructed textured surface after the compression, which closely resembles the original one.

Note that the preservation of the bijectivity of the texture map after compression is necessary. Figure 6(A) shows the original textured Buddha surface. Figure 6(B) shows the compressed textured surface using the Fourier compression of the Beltrami representation. The reconstructed textured surface closely resembles the original one. Figure 6(C) shows the compressed textured surface using the Fourier compression of the coordinate functions, with the same compression ratio as in Figure 6(B). Distortion of the texture can clearly be observed.

Quantitative experiments have also been carried out. Experiments are performed on three surface meshes, namely, "Susan," "Buddha," and "Zebra" (see Figure 7). Susan is a simply connected open surface with 5161 vertices. Buddha and Zebra are genus-0 closed surfaces with 15138 vertices and 20157, vertices respectively, which are partitioned into several simply connected open surfaces. Since some partitions in Buddha and Zebra contain very few vertices, we applied the compression algorithm only on major parts with more vertices. We tested the

Figure 6. *Comparison between the results of compressing the Beltrami representation and the coordinate function.* (A) *shows the original textured Buddha surface.* (B) *shows the Beltrami representation compression result.* (C) *shows the coordinate function compression result.*

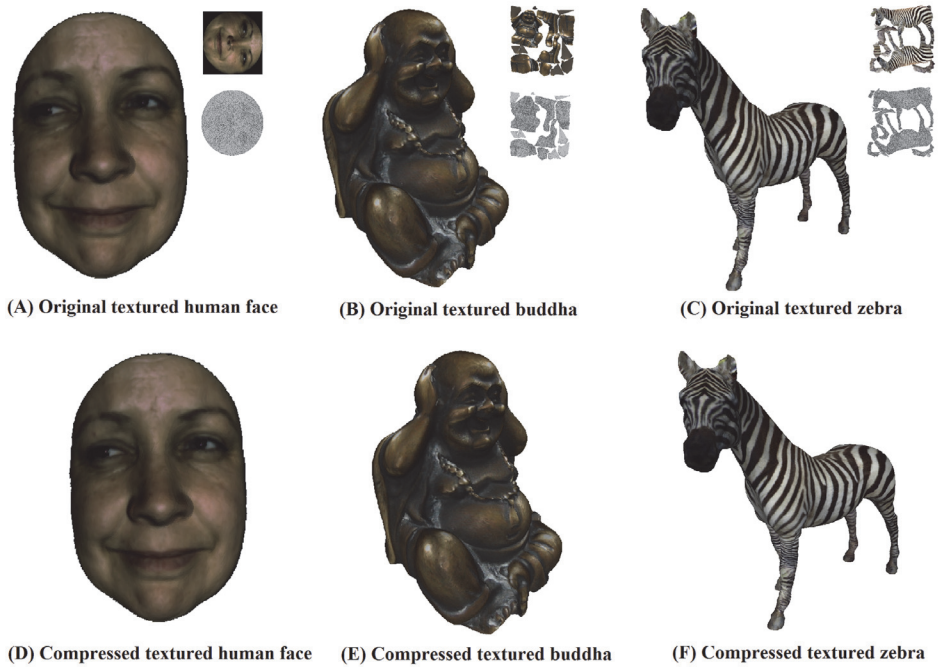

Figure 7. *Summary of compression results using* 1% *Fourier coefficients. The texture map examples are freely available from* http://www.kunzhou.net/tex-models.htm.

proposed algorithm with 1% and 3% of the Fourier coefficients.

To measure the accuracy of the compression scheme, we compute the root mean square error (RMSE) between the original and the reconstructed texture maps, respectively. We also compute the compression ratio (CR) to quantify the compression efficiency of the algorithm. The compression results for each partition of the surface meshes are shown in Table 2. Results show that the reconstructed textured surface after compression is very close to the original data, with very small RMSE. Even in the case when only 1% of Fourier coefficients are saved, the RMSE still remains in the order of 10^{-3}. As expected, the RMSE is smaller when 3% of

Table 2
Root mean square error.

	Vertices	Faces	RMSE(1%)	CR(1%)	RMSE(3%)	CR(3%)
Susan	5161	9999	$5.103e^{-4}$	12.29:1	$2.304e^{-4}$	8.32:1
Buddha-1	6523	12779	$3.009e^{-3}$	16.64:1	$1.731e^{-3}$	10.07:1
Buddha-2	1921	3609	$4.298e^{-3}$	7.19:1	$2.418e^{-3}$	5.67:1
Buddha-3	2385	4563	$3.551e^{-3}$	9.54:1	$1.870e^{-3}$	6.99:1
Buddha-4	2001	3842	$2.194e^{-3}$	10.21:1	$1.021e^{-3}$	7.33:1
Buddha-5	1985	3808	$2.766e^{-3}$	10.03:1	$1.551e^{-3}$	7.24:1
Zebra-1	7638	14683	7.042^{-3}	10.36:1	$4.898e^{-3}$	7.41:1
Zebra-2	1045	1924	$3.217e^{-3}$	5.71:1	$2.030e^{-3}$	4.73:1
Zebra-3	1287	2375	$3.914e^{-3}$	5.85:1	$2.311e^{-3}$	4.80:1
Zebra-4	7600	14622	$7.576e^{-3}$	10.53:1	$4.801e^{-3}$	7.50:1
Zebra-5	767	1414	$1.395e^{-3}$	5.82:1	$1.116e^{-3}$	4.79:1

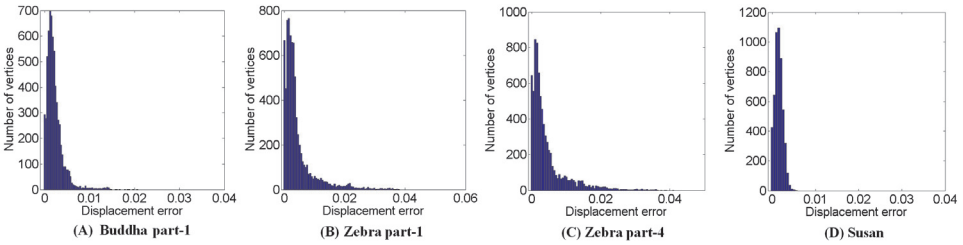

Figure 8. *Statistics of the displacement error of each vertex.* (A) *Buddha part*-1. (B) *Zebra part*-1. (C) *Zebra part*-4. (D) *Susan. The majority of the reconstructed texture coordinates have less than* 1 *relative percentage error.*

Fourier coefficients are saved (see column 5). The CRs are shown in columns 4 and 6, which illustrate that our proposed algorithm can significantly reduce the memory for texture maps. Note also that the table shows the out-performance of CR in Buddha-1, Zebra-1, Zebra-4, and Susan. This is expected, as texture coordinates of the boundary vertices contribute a relatively small amount of total storage of the fine meshes. Hence, our compression algorithm, which compresses the texture coordinates in the interior, gives better compression results. Figure 8 presents the displacement error of each reconstructed texture coordinate measured in the supreme norm. It is observed that the majority of the texture coordinates can be reconstructed almost losslessly.

In Table 3, we study the compression performance of the whole mesh for the three surfaces. For each surface, 1% of Fourier coefficients are stored. Note that the RMSEs are of order 10^{-3} in all three cases. Therefore, there is no noticeable difference between the compressed textured surfaces and the original ones.

The actual memory required is also listed. Note that after the compression, the memories of the textured surfaces are much less than those of the original data. The CRs are at least 12:1. This demonstrates the efficacy of our proposed method.

Furthermore, surface parameterization can also be represented by a triangular mesh of the parameter domain. Under this framework, compression of the texture maps can be done through mesh compression techniques. We have carried out experiments and compared our

Table 3
Summary of the texture map compression result.

	Susan	Buddha	Zebra
Number of vertices	5161	15138	20157
RMSE	$5.103e^{-4}$	$3.176e^{-3}$	$6.796e^{-3}$
CR	12.29:1	17.27:1	13.68:1
Data saving	91.86%	94.21%	92.69%
Memory required	1.6406 kB	3.3691 kB	5.6289 kB
Original memory required	20.1548 kB	58.1883 kB	77.0027 kB

Table 4
Comparison for texture map compression with an error tolerance of 10^{-3}*.*
(P = parallelogram rule, S = spectral compression, B = our method).

	Mesh1	Mesh2	Mesh3	Mesh4	Mesh5	Mesh6
Vertices Faces	7638 13443	6258 12203	2179 4124	1687 3112	1090 1971	879 1582
Overlap numbers	38 (P) 20 (S) 0 (B)	190 (P) 102 (S) 0 (B)	21 (P) 36 (S) 0 (B)	10 (P) 21 (S) 0 (B)	1 (P) 7 (S) 0 (B)	1 (P) 16 (S) 0 (B)
Bit per vertex	6.45 (P) 14.39 (S) 4.18 (B)	6.65 (P) 16.51 (S) 1.92 (B)	7.09 (P) 20.99 (S) 1.49 (B)	7.92 (P) 19.26 (S) 1.66 (B)	8.55 (P) 16.38 (S) 4.04 (B)	10.02 (P) 14.20 (S) 3.64 (B)

algorithm for texture map compression with existing state-of-the art mesh compression techniques, namely, the spectral mesh compression (S) by Kami and Gotsman [27] and the vertex position prediction algorithm using the parallelogram rule (P) introduced by Touma and Gotsman [43]. For fairness, we take into account only the memory required for the geometric information (the texture coordinates in this case) and ignore the memory needed for connectivity information. Table 4 shows the compression results for the three methods with an error tolerance of 10^{-3}. Note that the overlapping numbers of the proposed algorithm are zero for all six texture maps, while the parallelogram rule algorithm (P) and the spectral method (S) both produce some flips in the texture maps. The last row shows the bit per vertex needed for the corresponding texture coordinates. Note that our proposed methods have the least bit per vertex needed for all cases. For meshes with larger vertex numbers, the proposed algorithm provides an even better compression ratio for the texture maps.

Finally, every smooth BC corresponds to a smooth bijective quasi-conformal map. As a result, no abnormal texture (e.g., zaggy textures) would appear on the textured mesh (e.g., zero overlap numbers in Table 4). The diffeomorphic property is also a major reason why there is no noticeable difference between the original and the reconstructed texture mapping even with higher RMSE (see Figure 9).

6.4. Video compression. To study the performance of our proposed algorithm for video compression, experiments have been carried out on real videos. In our experiments, all B-frame in the Group of Picture are taken away for simplicity. Instead, we consider the following frame set: $I_1P_1P_2P_3P_4$. MV fields in each P-frame are obtained from the previous I-frame or P-frame, which are then compressed by our proposed algorithm. In order to study the

Figure 9. *A close look at Zebra part-*1. (A) *Uncompressed texture map.* (B) *Compressed texture map with* 4.9×10^{-3} *RMSE.*

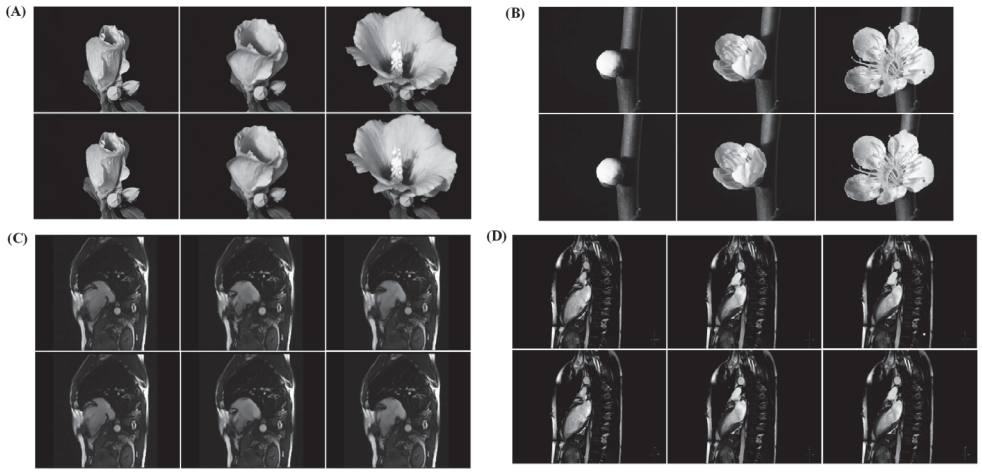

Figure 10. *Comparison between the uncompressed and the compressed video using* 0.5% *of Fourier coefficients:* (A) *Flower* 1. (B) *Flower* 2. (C) *Heart* 1. (D) *Heart* 2. *Upper rows are some original fourth P-frames in each frame set. The lower rows show the corresponding compressed P-frames.*

performance of our algorithm without adding any residual, a per-pixel MV field is used. It should be noted that the proposed algorithm can also be applied to block-based MV fields in exactly the same way. However, residual has to be added. In the decoding process, errors may be introduced to the MV fields after the compression. As a result, the reconstructed P-frame $\widetilde{P}_i$ might not be identical to the original frame P_i. Meanwhile, the decoding of P_{i+1} is based on the reconstructed P-frame $\widetilde{P}_i$ rather than the original P_i. The reconstruction error would be accumulated. However, experimental results show that the accumulated errors are small enough that the effect to the overall results would be negligible.

Experiments have been carried out on four video clips, namely, "Flower 1," "Flower 2," "Heart 1," and "Heart 2." Their resolutions are $[360 \times 262]$, $[512 \times 384]$, $[600 \times 480]$, and $[1280 \times 720]$, respectively. Different percentages of Fourier coefficients have been used. Figure 10 shows some original P-frames of the four videos and their corresponding compressed P-frames using our proposed algorithm. Here, 0.5% of Fourier coefficients are used. As shown in the figure, no visible differences can be observed after compression. It demonstrates the effectiveness and accuracy of our proposed algorithm in compressing the MV field.

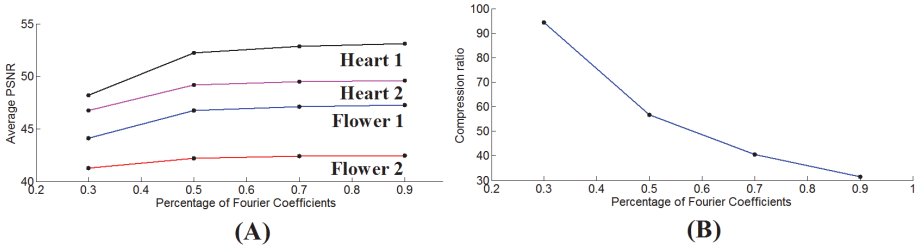

Figure 11. (A) *The average PSNR between the uncompressed and the compressed P-frames for the four videos with different percentages of Fourier coefficients saved.* (B) *The corresponding compression ratio obtained.*

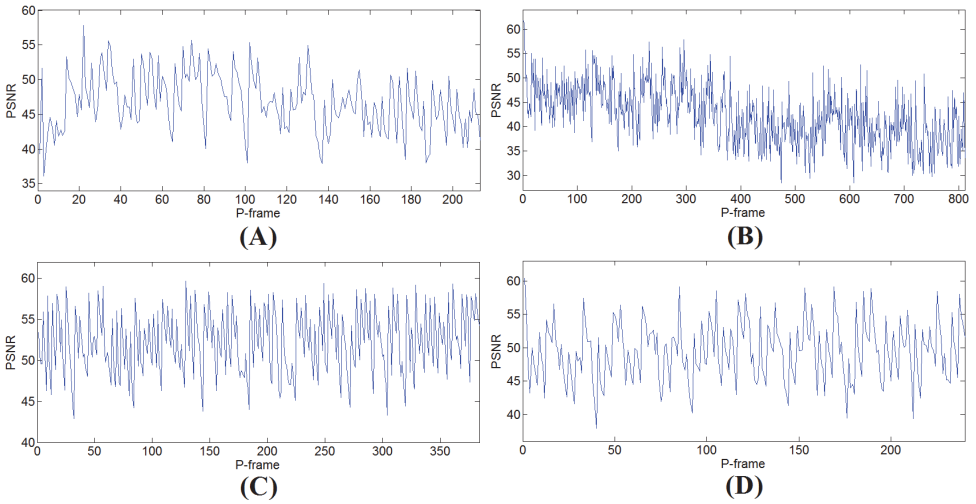

Figure 12. *PSNR between each pair of uncompressed and compressed P-frames.* (A) *Flower* 1. (B) *Flower* 2. (C) *Heart* 1. (D) *Heart* 2.

We also examine the performance of our algorithm quantitatively. Figure 11 shows the peak signal-to-noise ratio (PSNR) between the reconstructed P-frames and the original P-frames. As shown in Figure 11(A), the average PSNR in all four cases is higher than 42 when 0.5% of Fourier coefficients are used. In other words, only a negligible increase in residual will be induced by the error of the reconstructed MV field. As a result, the required memory for the residual image will not be affected much after compression with our algorithm.

Figure 11(B) shows the corresponding compression ratio for different percentages of Fourier coefficients stored. When 0.5% of coefficients are saved, the CR of the MV field can be as high as 56:1. The high CR of the MV field can effectively improve the CR of the existing video compression algorithm, such as MPEG and H.264. After the compression, the memory of the MV field is almost negligible compared with the total file size. For example, suppose the MV fields contribute 1/3 total storage of the compressed video; using our algorithm to compress the MV field, the storage requirement of the compressed video can be reduced by around 32%.

Figure 12 shows the PSNR between each pair of compressed and original P-frames. It

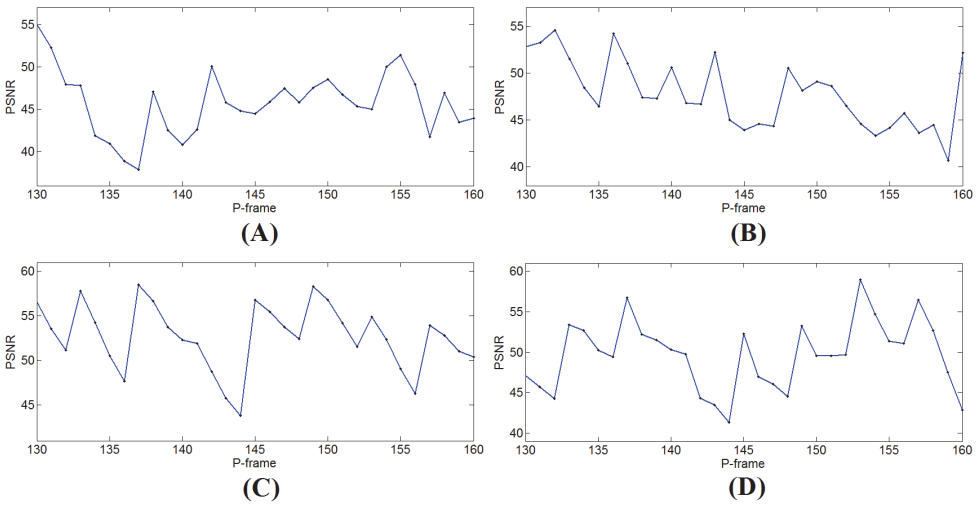

Figure 13. *A closer look at the plots in Figure* 12*. Patterns of decreasing PSNR with a period of four frames can be observed. It is the consequence of the accumulation of errors from previous P-frame reconstruction.* (A) *Flower* 1. (B) *Flower* 2. (C) *Heart* 1. (D) *Heart* 2.

is observed that PSNR stays close to about 45. It means the compressed P-frames closely resemble the original ones. Figure 13 shows the zoom-in of the plots in Figure 12. Decreases in the PSNR can be observed in each frame set. This is expected since the errors in MV fields are accumulated in each cycle. However, the decreases are very small. It means the effect of the accumulated errors in the MV fields on the overall result is negligible.

7. Conclusion. We address the problem of compressing surface homeomorphisms, which has important applications in computer graphics and imaging. Surface homeomorphisms are usually represented and stored by their 3D coordinate functions. It often requires lots of memory, which causes inconvenience in data transmission and data storage. In this paper, we propose an effective algorithm for compressing piecewise linear bijective surface maps between meshes using their Beltrami representations. The Beltrami representation is a complex-valued function defined on triangular faces of the surface mesh with supreme norm strictly less than 1. Under suitable normalization, there is a 1-1 correspondence between the set of surface homeomorphisms and the set of Beltrami representations. Given a Beltrami representation, the associated bijective surface map can be exactly reconstructed using the linear Beltrami solver introduced in this paper. Since the Beltrami representation has very few constraints, it can be easily combined with the Fourier approximation to compress the bijective surface map without distorting the bijectivity of the map. This significantly reduces the storage requirement for surface maps. In this paper, we proposed applying the algorithm to texture map compression and video compression. With our algorithm, the storage requirement for textured surfaces can be significantly reduced, while well preserving the quality of the original data. Our algorithm can also be applied to compressing MV fields for video compression. After compressing the MV field, the CR of the state-of-the-art video compression algorithms can be significantly improved. Experimental results on real textured surfaces and videos demonstrate the effectiveness and efficacy of our proposed algorithms.

REFERENCES

[1] V. Arsigny, O. Commowick, X. Pennec, and N. Ayache, *A log-Euclidean framework for statistics on diffeomorphism*, in Medical Image Computing and Computer-Assisted Intervention–MICCAI 2006, Lecture Notes in Comput. Sci. 4190, Springer, Berlin, 2006, pp. 924–931.

[2] J. Ashburner, *A fast diffeomorphic image registration algorithm*, NeuroImage, 38 (2007), pp. 95–113.

[3] L. Balmelli, G. Taubin, and F. Bernardini, *Space-optimised texture maps*, Comput. Graph. Forum, 21 (2002), pp. 411–420.

[4] M. F. Beg, M. I. Miller, A. Trouvé, and L. Younes, *Computing large deformation metric mappings via geodesic flows of diffeomorphisms*, Int. J. Comput. Vis., 61 (2005), pp. 139–157.

[5] C. Bennis, J.M. Vézien, and G. Iglésias, *Piecewise surface flattening for non-distorted texture mapping*, Comput. Graphics, 25 (1991), pp. 237–246.

[6] F. L. Bookstein, *Principal warps: Thin-plate splines and the decomposition of deformations*, IEEE Trans. Pattern Anal. Mach. Intell., 11 (1989), pp. 567–585.

[7] R. J. Campbell and P. J. Flym, *A survey of free-form object representation and recognition techniques*, Image Vis. Comput., 81 (2001), pp. 166–210.

[8] B. B. Chai, S. Sethuraman, H. S. Sawhney, and P. Hatrack, *Depth map compression for real-time view-based rendering*, Pattern Recognition Lett., 25 (2004), pp. 755–766.

[9] G. E. Christensen and H. J. Johnson, *Consistent image registration*, IEEE Trans. Med. Imag., 20 (2001), pp. 568–582.

[10] M. Deering, *Geometry compression*, Comput. Graphics, 29 (1995), pp. 13–20.

[11] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, *Multiresolution analysis of arbitrary meshes*, Comput. Graphics, 29 (1995), pp. 173–182.

[12] B. Fischl, M. Sereno, R. Tootell, and A. Dale, *High-resolution intersubject averaging and a coordinate system for the cortical surface*, Hum. Brain Mapp., 8 (1999), pp. 272–284.

[13] J. Foley, A. van Dam, S. Feiner, and J. Hughes, *Computer Graphics*, Addison–Wesley, Reading, MA, 1992.

[14] F. Gardiner and N. Lakic, *Quasiconformal Teichmüller Theory*, American Mathematical Society, Providence, RI, 2000.

[15] M. Ghanbari, *Video Coding—an Introduction to Standard Codecs*, The Institution of Electrical Engineers, Stevenage, UK, 1999.

[16] X. Gu, Y. Wang, T. F. Chan, P. M. Thompson, and S.-T. Yau, *Genus zero surface conformal mapping and its application to brain surface mapping*, IEEE Trans. Med. Imag., 23 (2004), pp. 949–958.

[17] X. Gu and S. Yau, *Computational Conformal Geometry*, Adv. Lect. Math. (ALM) 3, International Press, Somerville, MA, 2008.

[18] S. Haker, S. Angenent, A. Tannenbaum, R. Kikinis, G. Sapiro, and M. Halle, *Conformal surface parameterization for texture mapping*, IEEE Trans. Visual. Comput. Graph., 6 (2000), pp. 181–189.

[19] P. S. Heckbert, *Survey of texture mapping*, IEEE Comput. Graph. Appl., 6 (1986), pp. 56–67.

[20] D. T. Hoang and J. S. Vitter, *Efficient Algorithms for MPEG Video Compression*, John Wiley & Sons, New York, 2002.

[21] M. K. Hurdal and K. Stephenson, *Discrete conformal methods for cortical brain flattening*, NeuroImage, 45 (2009), pp. 86–98.

[22] J. W. Ioup, M. L. Gendron, and M. C. Lohrenz, *Vector map data compression with wavelets*, J. Navigation, 53 (2000), pp. 437–449.

[23] K. Iourcha, K. Nayak, and Z. Hong, *System and Method for Fixed-Rate Block-based Image Compression with Inferred Pixels Values*, US Patent 5956431, 1999.

[24] M. Isenburg and J. Snoeyink, *Compressing texture coordinates with selective linear prediction*, in Proceedings of Computer Graphics International, 2003, pp. 126–131.

[25] X. Jing and L. Chau, *An efficient three-step search algorithm for block motion estimation*, IEEE Trans. Multimedia, 6 (2004), pp. 435–438.

[26] S. C. Joshi and M. I. Miller, *Landmark matching via large deformation diffeomorphisms*, IEEE Trans. Image Process., 9 (2000), pp. 1357–1370.

[27] Z. Kami and C. Gotsman, *Spectral compression of mesh geometry*, in Proceedings of ACM SIGGRAPH,

2000, pp. 279–286.
[28] A. Kolesnikov and A. Akimov, *Distortion-constrained compression of vector maps*, in Proceedings of the 2007 ACM Symposium on Applied Computing, 2007, pp. 8–12.
[29] V. Kraevoy, A. Sheffer, and C. Gotsman, *Matchmaker: Constructing constrained texture maps*, ACM Trans. Graphics, 22 (2003), pp. 326–333.
[30] D. LeGall, *MPEG: A video compression standard for multimedia applications*, Comm. ACM, 34 (1991), pp. 46–58.
[31] B. Lévy, *Constrained texture mapping for polygonal meshes*, in Proceedings of ACM SIGGRAPH, 2001, pp. 417–424.
[32] B. Lévy, S. Petitjean, N. Ray, and J. Maillot, *Least squares conformal maps for automatic texture atlas generation*, in Proceedings of ACM SIGGRAPH, 2002, pp. 362–371.
[33] J. Lewis and M. English, *Compression of body surface potential maps using image compression techniques*, in Proceeding of Computers in Cardiology, 1995, pp. 401–404.
[34] T. Lin, E. F. Lee, I. Dinov, C. Le Guyader, P. Thompson, A. W. Toga, and L. A. Vese, *Gene to mouse atlas registration using a landmark-based nonlinear elasticity smoother*, in Proceedings of SPEI Medical Imaging: Image Processing, 2009, pp. 1–16.
[35] L. M. Lui, S. Thiruvenkadam, Y. Wang, P. M. Thompson, and T. Chan, *Optimized conformal surface registration with shape-based landmark matching*, SIAM J. Imaging Sci., 3 (2010), pp. 52–78.
[36] L. M. Lui, T. W. Wong, X. F. Gu, T. F. Chan, and S. T. Yau, *Compression of surface diffeomorphism using Beltrami coefficient*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 2839–2846.
[37] L. M. Lui, T. W. Wong, W. Zeng, X. F. Gu, P. M. Thompson, T. F. Chan, and S. T. Yau, *Optimization of surface registrations using Beltrami holomorphic flow*, J. Sci. Comput., 50 (2010), pp. 557–585.
[38] S. D. Ma and H. Lin, *Optimal texture mapping*, in Proceedings of EUROGRAPHICS, 1988, pp. 421–428.
[39] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr, *Discrete differential-geometry operators for triangulated 2-manifolds*, in Proceedings of Visual Mathematics, 2002.
[40] J. Stachera and P. Rokita, *Normal map compression based on BTC and wavelet coding*, in Proceeding of Real-Time Image Processing, 2008.
[41] G. Taubin and J. Rossinac, *Geometric compression through topological surgery*, ACM Trans. Graphics, 17 (1998), pp. 84–115.
[42] A. Tekalp, *Digital Video Processing*, Prentice–Hall, Englewood Cliffs, NJ, 1995.
[43] C. Touma and C. Gotsman, *Triangle mesh compression*, in Proceedings of Graphics Interface, 1998, pp. 26–34.
[44] M. Vaillant and J. Glaunés, *Surface matching via currents*, Inf. Process. Med. Imag., 19 (2005), pp. 381–392.
[45] M. Vaillant, A. Qiu, J. Glaunés, and M. I. Miller, *Diffeomorphic metric surface mapping in superior temporal gyrus*, NeuroImage, 34 (2007), pp. 1149–1159.
[46] T. Vercauteren, X. Pennec, A. Perchant, and N. Ayache, *Diffeomorphic demons: Efficient non-parametric image registration*, NeuroImage, 45 (2009), pp. S61–S72.
[47] Y. Wang, J. Ostermann, and Y. Q. Zhang, *Video Processing and Communications*, Prentice–Hall, Upper Saddle River, NJ, 2002.
[48] B. T. T. Yeo, M. R. Sabuncu, T. Vercauteren, N. Ayache, B. Fishl, and P. Golland, *Spherical demons: Fast diffeomorphic landmark-free surface registration*, IEEE Trans. Med. Imag., 29 (2010), pp. 1–20.
[49] Y. Yin, E. A. Hoffman, K. Ding, J. M. Reinhardt, and C. L. Lin, *A cubic B-spline-based hybrid registration of lung CT images for a dynamic airway geometric model with large deformation*, Med. Phys., 36 (2009), pp. 4213–4222.
[50] M. H. Yoon, D. O. Kim, R. H. Park, and S. W. Lee, *Geometry-dependent texture map compression*, Electron. Lett., 46 (2010), pp. 43–44.
[51] E. Zhang, K. Mischaikow, and G. Turk, *Feature-based surface parameterization and texture mapping*, ACM Trans. Graphics, 24 (2005), pp. 1–27.

[52] C. Zhu, X. Lin, and L. P. Chau, *Hexagon-based search pattern for fast block motion estimation*, IEEE Trans. Circuits Syst. Video Technol., 12 (2002), pp. 349–355.
[53] S. Zhu and K. K. Ma, *A new diamond search algorithm for fast block-matching motion estimation*, IEEE Trans. Image Process., 9 (2000), pp. 287–290.
[54] B. Zitová and J. Flusser, *Image registration methods: A survey*, Image Vis. Comput., 21 (2003), pp. 977–1000.