BELTRAMI REPRESENTATION AND ITS APPLICATIONS TO TEXTURE MAP AND VIDEO COMPRESSION

LOK MING LUI, KA CHUN LAM, TSZ WAI WONG, 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 3D coordinates each vertex is mapped to, which often requires lots of storage 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 memory can be effectively reduced, which is useful for many practical problems in computer graphics and imaging. In this paper, we proposed to apply 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 improve the compression ratio.

Key words. Beltrami representation, registration, Linear Beltrami Solver, texture map compression, video compression, motion vector compression

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 [1, 2, 3]. While in computer graphics, surface parameterizations are needed for texture mapping [4, 5, 6]. There are many different approaches for surface registration and parameterization. A commonly used method is to find surface maps satisfying certain constraints, such as matching landmarks [7, 10, 17, 18] and minimizing distortions [10, 11, 12, 14]. Surface maps computed from these processes can be highly convoluted and are usually represented and stored as 3D coordinate functions in $\mathbb{R}^3$. A huge storage memory is therefore required, especially when a large set of fine surface meshes are to be processed. For instance, in computer graphics, mesh parameterization that maps each vertex of the surface to a 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 reduces the storage requirement.

In this work, we propose an effective algorithm for compressing surface homeomorphism 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 an 1-1 correspondence between the set of surface homeomorphisms and the set of Bel-
trami 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 to it. In this paper, we propose an algorithm called the Linear Beltrami Solver to reconstruct the surface map associated to 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 little constraints. The only constraint is that its supreme norm has to be strictly less than 1. It does not have any requirements of injectivity nor surjectivity. This allows us to further compress the Beltrami representation using Fourier approximations, without distorting the bijectivity of the map. The storage memory 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 4.2, Figure 4.3 and Figure 4.4).

Our proposed algorithm for surface map compression can be practically applied to problems in computer graphics and imaging. In this paper, we propose to apply 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 improve the compression ratio.

In short, the contribution of this paper is three-folded: 1. we propose a compression algorithm for surface homeomorphism 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 follow. 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. Applications of the proposed algorithm to texture map compression and video compression are described in Section 5. In Section 6, a conclusion is drawn.

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

Surface parameterization/registration. Surface parameterization and registration have been extensively studied, for which different representations of bijective surface maps have been proposed. Conformal registration, which minimizes angular distortion, have been widely used to obtain a smooth 1-1 correspondence between surfaces [9, 10, 12, 14, 11, 57]. For example, Hurdal et al. [14] proposed to compute the conformal parameterizations using circle packing and applied it to registration of human brains. Gu et al. [10, 12, 11] proposed to compute the conformal parameterizations of human brain surfaces for registration using harmonic energy minimization and holomorphic 1-forms. Conformal registration is advantageous to local geometry preservation. But it cannot match feature landmarks, such as sulcal landmarks on the human brains, consistently. To alleviate this issue, Wang et al. [15, 18] proposed to compute the optimized conformal parameterizations of brain surfaces by minimizing a compounded energy. All of the above algorithms represent surface maps with their 3D coordinate functions. Special attention is required to ensure the bijectivity
of the resulting registration. Besides, smooth vector field has also been proposed to represent surface maps. Lui et al. [17] proposed the use of vector fields to represent surface maps and reconstruct them through integral flow equations. They obtained shape-based landmark matching harmonic maps by looking for the best vector fields minimizing a shape energy. The use of vector fields to represent surface maps makes optimization easier, but they cannot describe all surface maps. Time dependent vector fields can be used to represent the set of all surface maps. For example, Joshi et al. [13] proposed the generation of large deformation diffeomorphisms for landmark point matching, where the registrations are generated as solutions to the transport equation of time dependent vector fields. The time dependent vector fields facilitate the optimization procedure, although they may not be a good representation of surface maps since they requires more storage memory.

**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. The problem of finding a suitable parameterization for texturing a polygonal mesh has been widely studied. Besides, the storage memory for texture properties contribute a significant portion of the total file size of a textured mesh. Texture mapping compression is therefore necessary. Recently, much attention has been focused on compressing texture images while preserving the quality of the textured 3D model. Balmelli et al. warped the texture and sub-sampled it to reduce the unnecessary image space. Yoon et al. proposed a wavelet-based image compression technique which is specialised for compressing texture properties of a polygonal mesh. Iourcha et al. proposed the S3TC texture compression method, which is the most common texture compression algorithm nowadays. S3TC is a lossy compression algorithm that adapts a block truncation coding. Recently, researches has also been carried out on the high dynamic range (HDR) texture compression, which encodes texture images at a constant 8 bits per pixel. Furthermore, to overcome the bandwidth problem in 3D rendering, mesh compression algorithms, which reduce the storage memory for mesh geometry and mesh connectivity, have been proposed. However, compression of texture coordinates has received less attention. Isenburg et al. proposed an algorithm in which texture coordinates are predicted from mesh vertices through parallelogram rule. However, such algorithm only reduces 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 last decades. Many techniques and algorithms for video compression have been proposed. The basic idea to achieve the compression is to remove temporal and spatial redundancies existing in video sequences. The first generation of video compression involved techniques for intra-frame coding and simple inter-frame coding. 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 motion vector estimation have been proposed. In this paper, we are interested in compressing the motion vectors using their Beltrami representations which can further improve the compression ratio of existing video compression algorithms.

**Surface map compression.** Compression of mappings has also been studied. Chai et al. proposed the depth map compression algorithm by encoding mappings
as a simplified triangular meshes. Lewis [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 et al. [34] developed an algorithm to compress normal maps by decomposing them in the frequency domain. Ioup [31] also proposed to compress vector map data in the frequency domain. Kolesnikov et al. [32] proposed an algorithm for distortion-constrained compression of vector maps, based on optimal polygonal approximations and dynamic quantizations of vector data. 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. [19]. In that work, Beltrami coefficient (BC) defined on every vertex of the mesh was proposed to represent bijective surface maps. BC 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 registration in [20]. In this paper, we propose to use Beltrami representation defined on each triangular faces to represent bijective surface maps. The Beltrami representation can exactly represent the surface map, regardless of the regularity of the triangulation. Also, Linear Beltrami Solver is proposed to efficiently reconstruct surface map from its Beltrami representation.

3. Mathematical Background. In this section, we describe some basic mathematical concepts related to our algorithms. For details, we refer the readers to [8][21][22].

A surface $S$ with a conformal structure is called a Riemann surface. Given two Riemann surfaces $M$ and $N$, a map $f: M \rightarrow N$ is conformal if it preserves the surface metric up to a multiplicative factor called the conformal factor. An immediate consequence is that every conformal map preserves angles. With the angle-preserving property, a conformal map effectively preserves the local geometry of the surface structure.

A generalization of conformal maps is the quasi-conformal maps, which are orientation preserving homeomorphisms between Riemann surfaces with bounded conformality distortion, in the sense that their first order approximations takes small circles to small ellipses of bounded eccentricity [8]. Thus, a conformal homeomorphism that maps a small circle to a small circle can also be regarded as quasi-conformal. Surface registrations and parameterizations can be considered as quasi-conformal maps. Mathematically, $f: \mathbb{C} \rightarrow \mathbb{C}$ is quasi-conformal provided that it satisfies the Beltrami equation:

$$\frac{\partial f}{\partial \bar{z}} = \mu(z) \frac{\partial f}{\partial z},$$

for some complex valued Lebesgue measurable $\mu$ satisfying $||\mu||_\infty < 1$. $\mu$ is called the Beltrami coefficient, which is a measure of non-conformality. In particular, the map $f$ is conformal around a small neighborhood of $p$ when $\mu(p) = 0$. Infinitesimally, around a point $p$, $f$ may be expressed with respect to its local parameter as follows:

$$f(z) = f(p) + f_z(p)z + f_{\bar{z}}(p)\bar{z}$$
$$= f(p) + f_z(p)(z + \mu(p)\bar{z}).$$

(3.2)
Beltrami representation and applications

Fig. 3.1. Illustration of how the Beltrami coefficient measures the conformality distortion of a quasi-conformal map.

Obviously, \( f \) is not conformal if and only if \( \mu(p) \neq 0 \). Inside the local parameter domain, \( f \) may be considered as a map composed of a translation to \( f(p) \) together with a stretch map \( S(z) = z + \mu(p)z \), which is postcomposed by a multiplication of \( f_z(p) \), which is conformal. All the conformal distortion of \( S(z) \) is caused by \( \mu(p) \). \( S(z) \) is the map that causes \( f \) to map a small circle to a small ellipse. From \( \mu(p) \), we can determine the angles of the directions of maximal magnification and shrinking and the amount of them as well. Specifically, the angle of maximal magnification is \( \arg(\mu(p))/2 \) with magnifying factor \( 1 + |\mu(p)| \); The angle of maximal shrinking is the orthogonal angle \( (\arg(\mu(p)) - \pi)/2 \) with shrinking factor \( 1 - |\mu(p)| \). The distortion or dilation is given by:

\[
K = \frac{1 + |\mu(p)|}{1 - |\mu(p)|}.
\]

Thus, the Beltrami coefficient \( \mu \) gives us all the information about the properties of the map (See Figure 3.1).

Given a Beltrami coefficient \( \mu : \mathbb{C} \to \mathbb{C} \) with \( \|\mu\|_{\infty} < 1 \). There is always a quasiconformal mapping from \( \mathbb{C} \) onto itself which satisfies the Beltrami equation in the distribution sense [8]. More precisely,

**Theorem 3.1 (Measurable Riemann Mapping Theorem).** Suppose \( \mu : \mathbb{C} \to \mathbb{C} \) is Lebesgue measurable satisfying \( \|\mu\|_{\infty} < 1 \), then there is a quasiconformal homeomorphism \( \phi \) 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 \( \infty \), the associated quasiconformal homeomorphism \( \phi \) is uniquely determined.

By reflection, the above theorem can be further extended to Beltrami coefficients defined on the unit disk \( \mathbb{D} \).

**Theorem 3.2.** Suppose \( \mu : \mathbb{D} \to \mathbb{C} \) is Lebesgue measurable satisfying \( \|\mu\|_{\infty} < 1 \), then there is a quasiconformal homeomorphism \( \phi \) 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 quasiconformal homeomorphism \( \phi \) is uniquely determined.

Theorem 3.1 and Theorem 3.2 suggest that under suitable normalization, a homeomorphism from \( \mathbb{C} \) or \( \mathbb{D} \) onto itself can be uniquely determined by its associated Beltrami coefficient. These two theorems can be extended to homeomorphisms between Riemann surfaces [20].
Beltrami representation. We also describe how the quasi-conformal homeomorphism proposed in this paper to represent and compress surface homeomorphisms using the Beltrami coefficient, we can easily compress the surface maps. These two theorems play important roles for the main algorithms proposed in this paper.

\[ \mu \]

Suppose \( \{ p_1, p_2, p_3 \in S_1 \} \leftrightarrow \{ q_1, q_2, q_3 \in S_2 \} \). By Riemann mapping theorem, \( S_1 \) and \( S_2 \) can both be uniquely parameterized by \( \phi_1: S_1 \to S^2 \cong \mathbb{C} \) and \( \phi_2: S_2 \to S^2 \cong \mathbb{C} \) respectively, such that \( \phi_1(p_1) = 0, \phi_1(p_2) = 1, \phi_1(p_3) = \infty \) and \( \phi_2(q_1) = 0, \phi_2(q_2) = 1, \phi_2(q_3) = \infty \). Given any surface diffeomorphism \( f: S_1 \to S_2 \), the composition map \( \tilde{f} := \phi_2 \circ f \circ \phi_1^{-1}: S^2 \to S^2 \) is a diffeomorphism from \( S^2 \) to itself fixing 0, 1 and \( \infty \). By Theorem 3.3, \( \tilde{f} \) can be uniquely represented by a Beltrami coefficient \( \tilde{\mu} \) defined on \( S^2 \). Hence, \( f \) can be uniquely represented by a Beltrami coefficient \( \mu := \tilde{\mu} \circ \phi_1^{-1} \) defined on \( S_1 \). In other words, we have the following:

**Theorem 3.3.** Let \( S_1 \) and \( S_2 \) be two genus-0 closed surfaces. Suppose \( f: S_1 \to S_2 \) is a surface diffeomorphism. Given 3-point correspondence \( \{ p_1, p_2, p_3 \in S_1 \} \leftrightarrow \{ f(p_1), f(p_2), f(p_3) \in S_2 \} \), \( f \) can be represented by a unique Beltrami coefficient \( \mu : S_1 \to \mathbb{C} \).

Similarly, let \( M_1 \) and \( M_2 \) be two genus-0 open surfaces. Given 2-points correspondence \( \{ p_1 \in M_1, p_2 \in \partial M_1 \} \leftrightarrow \{ q_1 \in M_2, q_2 \in \partial M_2 \} \) between them, we can again uniquely parameterize \( M_1 \) and \( M_2 \) to map the corresponding points to 0 and 1. Denote them by \( \phi_1: M_1 \to \mathbb{D} \) and \( \phi_2: M_2 \to \mathbb{D} \). The composition map \( \tilde{f} := \phi_2 \circ f \circ \phi_1^{-1}: \mathbb{D} \to \mathbb{D} \) is a diffeomorphism of \( \mathbb{D} \) fixing 0 and 1. Again, \( \tilde{f} \) can be uniquely represented by a Beltrami coefficient \( \tilde{\mu} \) defined on \( \mathbb{D} \). Hence, \( f \) can be uniquely represented by a Beltrami coefficient \( \mu := \tilde{\mu} \circ \phi_1^{-1} \) defined on \( M_1 \). So, we have the following theorem:

**Theorem 3.4.** Let \( M_1 \) and \( M_2 \) be two genus 0 open surfaces with disk topology. Suppose \( f: M_1 \to M_2 \) is a surface diffeomorphism. Given 2-point correspondence \( \{ p_1 \in M_1, p_2 \in \partial M_1 \} \leftrightarrow \{ f(p_1) \in M_2, f(p_2) \in \partial M_2 \} \), \( f \) can be represented by a unique Beltrami coefficient \( \mu : M_1 \to \mathbb{C} \).

Theorem 3.3 and 3.4 allow us to represent a homeomorphism of genus-0 closed surfaces or open surfaces with disk topology using a Beltrami coefficient. The Beltrami coefficient uniquely determine the surface map with least constraint. Using the Beltrami coefficient, we can easily compress the surface maps. These two theorems 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 this paper to represent and compress surface homeomorphisms using the Beltrami representation. We also describe how the quasi-conformal homeomorphism...
can be reconstructed from its associated Beltrami representation using the proposed Linear Beltrami Solver.

4.1. Beltrami representation for bijective maps between meshes. 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 3D coordinate functions have to satisfy certain constraint on the Jacobian $J$ (namely, $J > 0$) 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.

Theorem 3.3 and 3.4 tell us that we can represent diffeomorphisms of smooth Riemann surfaces using Beltrami coefficients. In practice, surfaces are represented discretely by triangular meshes. Surface maps are usually approximated by piecewise linear homeomorphisms between meshes. Our goal is to represent piecewise linear maps between meshes by a simple and easy-to-manipulate representation.

Suppose $K_1$ and $K_2$ are two surface meshes with the same topology (either genus-0 closed surface mesh or simply-connected open surface mesh). Define the set of vertices on $K_1$ and $K_2$ by $V_1 = \{v_1^i\}_{i=1}^n$ and $V_2 = \{v_2^i\}_{i=1}^n$ respectively. Similarly, define the set of triangular faces on $K_1$ and $K_2$ by $F_1 = \{T_1^j\}_{j=1}^m$ and $F_2 = \{T_2^j\}_{j=1}^m$ respectively. Now, consider a piecewise linear homeomorphism $f : K_1 \rightarrow K_2$ between $K_1$ and $K_2$, which is orientation preserving. In other words, $f|_{T_1^j}$ is piecewise linear and orientation preserving. We aim to represent $f$ by a complex-valued function defined on each triangular face of $K_1$.

We first parameterize $K_1$ and $K_2$. Suppose $K_1$ is a simply-connected open surface mesh. $K_1$ and $K_2$ can be parameterized onto a unit square $R = [0, 1] \times [0, 1]$ by harmonic maps: $\phi_1 : K_1 \rightarrow R$ and $\phi_2 : K_2 \rightarrow R$ respectively (See Figure 4.1(A)). The harmonic parameterizations $\phi_i$ ($i = 1, 2$) can be computed by solving a sparse linear system [57]:

$$
\sum_{[u,v] \in K_i} k_{u,v}(\phi_i(u) - \phi_i(v)) = 0 \quad \text{with} \quad (4.1)
$$

$$
\phi_i(p_0) = (0, 0); \ \phi_i(p_1) = (0, 1); \ \phi_i(p_2) = (1, 0); \ \phi_i(p_3) = (1, 1);
$$

where $p_0, p_1, p_2$ and $p_3$ are four fixed vertices on the boundary. Other boundary vertices are constrained to be uniformly distributed on the boundary of the unit square $R$. The weight $k_{uv}$ is given by the cotangent formula: $k_{uv} = \cot \alpha_{uv} + \cot \beta_{uv}$ where $\alpha_{uv}$ and $\beta_{uv}$ are the two adjacent angles of the edge $[u,v]$. 

**Fig. 3.3.** Reconstruction of surface homeomorphism from its Beltrami representation. (A) shows the original surface homeomorphism between two genus-0 closed surfaces. (B) shows the reconstructed surface map from its Beltrami representation. The reconstructed map closely resembles the original one. (C) shows the colormap of the norm of the Beltrami representation.
Fig. 4.1. 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.

In case $K_1$ is a genus-0 closed surface mesh, the surface mesh can be parameterized onto a big triangle on $\mathbb{R}^2$ by cutting away one triangular face (See Figure 4.1(B)). The harmonic parameterizations $\phi_i$ ($i = 1, 2$) from $K_i$ to a triangle on $\mathbb{R}^2$ can be computed as follows:

$$
\sum_{[u,v] \in K_i} k_{u,v}(\phi_i(u) - \phi_i(v)) = 0 \quad \text{with} \quad \phi_i(v_1) = p_1, \quad \phi_i(v_2) = p_2, \quad \phi_i(v_3) = p_3 \quad (4.2)
$$

where $T_i = [v_1^i, v_2^i, v_3^i]$ corresponds to the cut away triangular face on $K_i$ and $\{p_1, p_2, p_3\}$ are three fixed point on $\mathbb{R}^2$.

The bijective map $f : K_1 \rightarrow K_2$ can now be represented by the Beltrami coefficient $\tilde{\mu}_f$ of the composition map $\tilde{f} : \phi_2 \circ f \circ \phi_1^{-1} : D \rightarrow D$, where $D$ is a triangular mesh in $\mathbb{R}^2$. $D$ is either the unit square or a triangle in the complex plane (See Figure 4.1).

To compute $\mu_f$, 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:

$$
\tilde{f}|_T(x, y) = \left( \begin{array}{c} a_Tx + b_Ty + r_T \\ c_Tx + d_Ty + q_T \end{array} \right) \quad (4.3)
$$

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:

$$
\left( \begin{array}{c} \vec{v}_1 - \vec{v}_0 \\ \vec{v}_2 - \vec{v}_0 \end{array} \right) \nabla_T \tilde{f}_i = \left( \begin{array}{c} \tilde{f}_i(\vec{v}_1) - \tilde{f}_i(\vec{v}_0) \\ \tilde{f}_i(\vec{v}_2) - \tilde{f}_i(\vec{v}_0) \end{array} \right), \quad (4.4)
$$

where $[\vec{v}_0, \vec{v}_1]$ and $[\vec{v}_0, \vec{v}_2]$ are two edges on $T$. By solving equation $4.4$, $a_T, b_T, c_T$ and $d_T$ can be obtained. The Beltrami coefficient $\mu_f(T)$ of the triangular face $T$ can then
be computed from the Beltrami equation \(\mu f(T)\) by:

\[
\mu f(T) = \frac{(a_T - d_T) + \sqrt{-1}(c_T + b_T)}{(a_T + d_T) + \sqrt{-1}(c_T - b_T)}
\]  

(4.5)

The Beltrami coefficient \(\mu_f\) defined on each triangular face of \(D\) uniquely determine the surface map \(\bar{f}\). Hence, the complex-valued function \(\mu_f := \mu_f \circ \phi_1\) defined on each face of \(K_1\) uniquely determine the surface map \(f : K_1 \rightarrow K_2\). We call \(\mu_f\) the Beltrami representation of \(f\). Given a Beltrami representation, the corresponding surface homeomorphism can be exactly computed, which will be described in the next subsection (See Figure 3.2 and 3.3).

In summary, the Beltrami representation of a homeomorphism between surface meshes can be computed as follows.

**Algorithm 4.1: (Computation of Beltrami representation)**

**Input:** Surface meshes: \(K_1\) and \(K_2\); piecewise linear homeomorphism: \(f : K_1 \rightarrow K_2\); a fixed triangular face \(T\) if \(K_1\) is genus-0 or 4 fixed points on \(\partial K_1\) if \(K_1\) is a topological disk.

**Output:** Beltrami representation: \(\mu_f\)

1. Get harmonic parameterizations \(\phi_1 : K_1 \rightarrow D\) and \(\phi_2 : K_2 \rightarrow D\) by solving the linear system \([4.1]\) or \([4.2]\).
2. Compute the gradient of \(\bar{f} := \phi_2 \circ f \circ \phi_1^{-1} : D \rightarrow D\) for each triangular face of \(D\) by solving the linear system \([4.4]\).
3. Compute the Beltrami coefficient of \(\bar{f}\) for each triangular face of \(D\) by equation \([4.5]\).
4. Compute the Beltrami representation \(\mu_f\) of \(f\) by computing \(\mu_f := \mu_f \circ \phi_1\).

**4.2. Linear Beltrami solver.** Given a Beltrami representation, it is important to have an algorithm to reconstruct the associated quasi-conformal homeomorphism.

Suppose \(K_1\) and \(K_2\) are two surface meshes with the same topology (either topological disks or genus-0 closed surface meshes). Given a Beltrami representation \(\mu\), which is a complex-valued function defined on each triangular face of \(K_1\), our goal is to reconstruct the associated homeomorphism \(f : K_1 \rightarrow K_2\) between \(K_1\) and \(K_2\).

We first compute the harmonic parameterizations \(\phi_1 : K_1 \rightarrow D\) and \(\phi_2 : K_2 \rightarrow D\) using equation \([4.1]\) or \([4.2]\). We shall look for a piecewise linear homeomorphism \(f : D \rightarrow D\) with Beltrami coefficient \(\mu_f := \phi_1 \circ \mu\). The desired homeomorphism \(f\) can then be reconstructed by taking the composition: \(f = \phi_2 \circ \bar{f} \circ \phi_1^{-1}\). Again, the restriction of \(\bar{f}\) on each triangular face \(T\) is linear and thus it can be written as:

\[
\bar{f}|_T(x, y) = \begin{pmatrix}
a_T x + b_T y + r_T \\
c_T x + d_T y + q_T
\end{pmatrix}
\]  

(4.6)

Let \(\mu(T) = \rho_T + \sqrt{-1} \tau_T\). Equation \([4.5]\) gives

\[
\rho_T + \sqrt{-1} \tau_T = \frac{(a_T - d_T) + \sqrt{-1}(c_T + b_T)}{(a_T + d_T) + \sqrt{-1}(c_T - b_T)}
\]  

(4.7)

This is equivalent to

\[
-d_T = \alpha_1^T a_T + \alpha_2^T b_T \\
c_T = \alpha_2^T a_T + \alpha_3^T b_T
\]  

(4.8)
where: \( \alpha_1 = \frac{(\rho_1^2-1)^2+\tau_2^2}{1-\rho_1^2-\tau_2^2}; \) \( \alpha_2 = -\frac{2\tau_2-\rho_2^2+\tau_2^2}{1-\rho_1^2-\tau_2^2}; \) \( \alpha_3 = \frac{1+2\rho_2^2+\tau_2^2}{1-\rho_1^2-\tau_2^2}. \)

Similarly, one can deduce that

\[
-b_T = \alpha_1^T c_T + \alpha_2^T d_T \\
ad_T = \alpha_3^T c_T + \alpha_4^T d_T
\]

(4.9)

Let \( T = [v_i, v_j, v_k] \) and \( w_I = \tilde{f}(v_I) \) where \( I = i, j \) or \( k \). Suppose \( v_I = g_I + \sqrt{-1} h_I \) and \( w_I = s_I + \sqrt{-1} t_I \) \((I = i, j, k)\). Using equation 4.4, \( \alpha_T, b_T, c_T \) and \( d_T \) can be written as follows:

\[
\alpha_T = A_i^T s_i + A_j^T s_j + A_k^T s_k; \quad b_T = B_i^T s_i + B_j^T s_j + B_k^T s_k; \quad c_T = A_i^T t_i + A_j^T t_j + A_k^T t_k; \quad d_T = B_i^T t_i + B_j^T t_j + B_k^T t_k;
\]

(4.10)

where:

\[
A_i^T = \frac{(h_j - h_k)}{2\text{Area}(T)}; \quad A_j^T = \frac{(h_k - h_i)}{2\text{Area}(T)}; \quad A_k^T = \frac{(h_i - h_j)}{2\text{Area}(T)}; \quad B_i^T = \frac{(g_k - g_j)}{2\text{Area}(T)}; \quad B_j^T = \frac{(g_i - g_k)}{2\text{Area}(T)}; \quad B_k^T = \frac{(g_j - g_i)}{2\text{Area}(T)}.
\]

(4.11)

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

\[
\sum_{T \in N_i} A_i^T b_T \text{Area}(T) = \sum_{T \in N_i} B_i^T a_T \text{Area}(T); \quad \sum_{T \in N_i} A_i^T d_T \text{Area}(T) = \sum_{T \in N_i} B_i^T c_T \text{Area}(T).
\]

(4.12)

Thus, following from equation 4.8 and 4.9, we have

\[
\sum_{T \in N_i} \left( A_i^T \left[ \alpha_1^T a_T + \alpha_2^T b_T \right] + B_i^T \left[ \alpha_3^T c_T + \alpha_4^T d_T \right] \right) \text{Area}(T) = 0 \quad (4.13)
\]

\[
\sum_{T \in N_i} \left( A_i^T \left[ \alpha_1^T a_T + \alpha_2^T b_T \right] + B_i^T \left[ \alpha_3^T c_T + \alpha_4^T d_T \right] \right) \text{Area}(T) = 0 \quad (4.14)
\]

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, equation 4.13 gives us the linear systems 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, equation 4.14 gives us the linear systems to solve for the \( y \)-coordinate function of \( \tilde{f} \).

Besides, \( \tilde{f} \) has to satisfy certain constraints on the boundary. When \( K_1 \) is a genus-0 closed surface mesh, the parameter domain \( D \) is a triangle with boundary vertices \( p_0, p_1 \) and \( p_2 \). In this case, the desired quasi-conformal map \( \tilde{f} \) should satisfy

\[
\tilde{f}(p_0) = p_0; \quad \tilde{f}(p_1) = p_1 \quad \text{and} \quad \tilde{f}(p_2) = p_2 \quad (4.15)
\]

When \( K_1 \) is a topological disk, the parameter domain \( D \) is unit square. In this case, the desired quasi-conformal map should satisfy

\[
\tilde{f}(0) = 0; \quad \tilde{f}(1) = 1 \quad \tilde{f}(i) = i \quad \tilde{f}(1 + i) = 1 + i; \quad \text{Re}(\tilde{f}) = 0 \quad \text{on arc } [0, i]; \quad \text{Re}(\tilde{f}) = 1 \quad \text{on arc } [1, 1 + i]; \quad \text{Imag}(\tilde{f}) = 0 \quad \text{on arc } [0, 1]; \quad \text{Imag}(\tilde{f}) = 1 \quad \text{on arc } [i, 1 + i]
\]

(4.16)
Equations 4.13 and 4.14 together with the above boundary conditions give a non-singular linear system to solve for \( \tilde{f} \). The linear system is symmetric positive definite. Hence, it can be solved effectively by the conjugate gradient method. Once \( \tilde{f} \) is computed, the desired quasi-conformal map between \( K_1 \) and \( K_2 \) can be computed by taking the composition: \( f = \phi_2^{-1} \circ \tilde{f} \circ \phi_1 \).

We summarize the reconstruction scheme of the quasi-conformal map from its Beltrami representation as follows.

**Algorithm 4.2: (Linear Beltrami Solver)**

**Input:** Surface meshes: \( K_1 \) and \( K_2 \); Beltrami representation \( \mu \)

**Output:** Quasi-conformal map \( f : K_1 \rightarrow K_2 \) associated to \( \mu \)

1. Get harmonic parameterizations \( \phi_1 : K_1 \rightarrow D \) and \( \phi_2 : K_2 \rightarrow D \) by solving the linear system 4.1 or 4.2
2. Compute the Beltrami coefficient \( \mu_f := \mu \circ \phi_1^{-1} \)
3. Compute the quasi-conformal map \( \tilde{f} : D \rightarrow D \) associated to \( \mu_f \) by solving the linear system 4.13 and 4.14 with boundary conditions 4.15 or 4.16
4. Compute the quasi-conformal map \( f : K_1 \rightarrow K_2 \) by \( f = \phi_2^{-1} \circ \tilde{f} \circ \phi_1 \)

In some situation, the surface mesh may be parameterized onto an arbitrary domain \( \Omega \) in \( \mathbb{R}^2 \) (such as in texture mapping applications). In this case, the Beltrami representation together with the boundary map uniquely determines the parameterization. By solving Equation 4.13 and 4.14 together with the boundary condition given by the boundary map, the parameterization can be exactly reconstructed.

Experimental results show that the proposed Linear Beltrami Solver can effectively compute the surface homeomorphism associated to a given Beltrami representation. Compared to the Beltrami Holomorphic Flow (BHF) method introduced in [19, 20], our method can compute the associated surface homeomorphism more accurately and efficiently. Table 4.1 shows the comparison of the computational time and error under the LBS method and the BHF method. Experimental results show that 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 4.1, 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.

### 4.3. Compression of surface homeomorphisms.

The Beltrami representation can be further compressed using the Fourier approximation to reduce the storage requirement. 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 is easily lost (see Figure 4.2, 4.3 and 4.4). The Beltrami representation, however, is advantageous because it does not have any requirement for injectivity and surjectivity, making the Jacobian constraint unnecessary. The only requirement for the Beltrami representation $\mu$ is that it has to be a complex-valued function defined on the triangular faces with supreme norm less than 1. We can therefore compress $\mu$ using Fourier approximations without losing the bijectivity.

Let $K_1$ be a simply-connected open surface mesh. Suppose $\mu$ is the Beltrami representation of the surface map $f : K_1 \rightarrow K_2$. We first parameterize $K_1$ to the unit square $D$ by the parameterization $\phi_1 : K_1 \rightarrow D$. We can then obtain the Beltrami coefficient $\tilde{\mu}_f := \mu \circ \phi_1^{-1}$, which is defined on triangular faces of $D$. Let $D_{\text{reg}}$ be the $N \times N$ regular finite difference grid of the unit square. Here, $N = \sqrt{N_f}$, where $N_f$ is the number of triangular faces of $K_1$. We linearly interpolate $\mu_\tilde{f}$ defined on $D$ onto $D_{\text{reg}}$ to obtain a Beltrami coefficient $\mu_\tilde{f}^{\text{reg}}$ defined on $D_{\text{reg}}$.

$\mu_\tilde{f}^{\text{reg}}$ can be expressed by Fourier expansion as follow:

$$\mu(x, y) = \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} c_{j,k} e^{2\sqrt{-1}\pi j x/N} e^{2\sqrt{-1}\pi k y/N}, \quad (4.17)$$

where

$$c_{j,k} = \frac{1}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \mu(x, y) e^{-2\sqrt{-1}\pi j x/N} e^{-2\sqrt{-1}\pi k y/N}.$$

We can use fast Fourier transform to compute the coefficients $c_{j,k}$ efficiently. One can then take fewer Fourier coefficients to approximate the Beltrami coefficient, which can significantly reduce the storage requirement. With the Fourier coefficients of the truncated Fourier series, the compressed Beltrami coefficient defined on $D$ and hence the compressed Beltrami representation $\mu^c$ defined on $K_1$ can be obtained. $\mu^c$ accurately approximates $\mu$, and hence the quasi-conformal map associated to $\mu^c$ closely resembles to $f$.

In summary, the Fourier compression scheme can be described as follows:
Beltrami representation and applications

Fig. 4.3. Compression of bijective map with different compression ratios. The top row shows the reconstructed map from the compressed Beltrami representation with different compression ratios. The bottom row shows the reconstructed map from the compressed coordinate functions with the same compression ratios as in the top row.

Algorithm 4.3: (Compression of Beltrami representation)

Input: Beltrami representation \( \mu \) and compression percentage \( \epsilon \)

Output: \( \epsilon \% \) Fourier coefficients \( c_{j,k} \)'s

1. Get harmonic parameterization \( \phi_1 : K_1 \rightarrow D \) by solving the linear system \( 4.2 \)
2. Compute the Beltrami coefficient \( \mu_f := \phi_1 \circ \mu \)
3. Compute \( \mu_{reg}^f \) defined on the regular grid \( D_{reg} \) by linear interpolation
4. Compute the Fourier coefficients of \( \mu_{reg}^f \) using equation \( 4.17 \)
5. Store \( \epsilon \% \) Fourier coefficients \( c_{j,k} \)'s

Experimental results show that the proposed compression algorithm is stable and effective in reducing the storage requirement of bijective surface maps. Figure 4.2(A) shows a homeomorphism from a regular grid. (B) shows the reconstructed homeomorphism from the compressed Beltrami representation. The reconstructed map closely resembles the original one. (C) shows the reconstructed map from the compressed coordinate functions. Note that the bijectivity is completely disrupted. In comparison, Beltrami compression gives accurate results with just a small number of coefficients. Figure 4.3 shows the result of Beltrami compression with different compression ratios. The reconstructed maps again closely resemble to the original data. In the bottom row of Figure 4.3, we show the reconstructed map from the compressed coordinate functions with the same compression ratios. Again, the bijectivity of the reconstructed maps is completely disrupted.

The proposed compression scheme can also be applied to compressing surface homeomorphisms. Figure 4.4(A) shows the original surface homeomorphism between
Fig. 4.4. Compression of bijective surface homeomorphism between two genus-0 closed surfaces. (A) shows the original surface homeomorphism between two closed surfaces. (B) shows the reconstructed surface map from the compressed Beltrami representation. (C) shows the reconstructed surface map from the compressed coordinate functions.

5. Applications. In this section, we apply our proposed compression algorithm to texture map and video compression. A flow chart summarizing the idea of the two applications is shown in Figure 5.1.

5.1. 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. As the demand for higher level of realism is rising, texture mapping of higher resolution is also required for producing more detailed models, which in turn increases the required storage capacity and transmission bandwidth for texture coordinates. The need for a compact representation of the texture mapping is therefore crucial. In this section, we propose to develop an effective algorithm for texture map compression using the Beltrami representation.

Mathematically, a texture map is a parameterization of a surface onto a 2D image. This parameterization is commonly represented by its coordinate functions \( \rho = (\rho_1, \rho_2) \), which maps each vertex of the mesh to a 2D position, given by

\[
\rho : M \rightarrow [0, 1] \times [0, 1]
\]

where \( M \) is the surface mesh. Conventionally, the target range is set to be \([0, 1] \times [0, 1]\), which can be obtained by normalizing the texture image. In other words, a texture mapping can be considered as a piecewise linear homeomorphism between the surface mesh \( M \) and a 2D mesh in \([0, 1] \times [0, 1]\). Once every vertices are assigned to a 2D position, the texture image can be mapped onto the 3D surface mesh.

Figure 5.2 illustrates the idea of texture mapping. The left shows a 2D texture image. Vertices \( a, b \) and \( c \) have texture coordinates \( \rho(a), \rho(b) \) and \( \rho(c) \) respectively. With this per-vertex assignment, texels can be assigned to the interior of the triangle by linear interpolation. In real application, a 3D model is usually represented by fine
meshes composed of many small triangular faces. With the texture mapping, texels can be mapped to the interior of each triangular faces, and hence the whole textured surface mesh can be obtained.

Texture images are not necessarily of regular shapes. In order to minimize the distortion of the texture mapping, surface meshes are often partitioned into patches and parameterized onto texture images with irregular shapes. Figure 5.3 shows a typical example of texture mapping. (A) shows the surface mesh of a zebra, which is partitioned into patches. Each patch is parameterized onto a texture image with irregular shapes, as shown in (B) and (C). In other words, the surface mesh consists of several texture mappings corresponding to different patches. With the texture mappings, texture images can be mapped onto the surface mesh. (D) shows the textured surface mesh of the zebra.

In practice, a textured surface mesh is commonly represented by its mesh geometry, mesh connectivity, texture mapping and texture image. Usually, the storage memories for texture properties contribute a significant portion of the total file size of the textured mesh. This motivates us to look for an effective compression scheme for texture mapping.

Here, we propose an effective algorithm for texture mapping compression by the Beltrami representation of the texture coordinates. Consider a surface mesh $M$ with texture mappings $\{\rho_i : M_i \to [0,1] \times [0,1]\}_{i=1}^n$, where $M_i$’s are patches of $M$. Our goal is to compress the texture mappings $\rho_i$’s. We first parameterize $M_i$ onto a unit square $D$ by harmonic parameterizations $\phi_i : M_i \to D$. It can be done by solving Equation 4.1. The Beltrami coefficient $\tilde{\mu}_i$ 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 determine $\rho_i$. Using the Fourier compression scheme as described in Algorithm 4.2, the Beltrami representation $\mu_i$ can be effectively compressed. Hence, the texture mapping $\rho_i$ 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.

**Fig. 5.1. Summary of the steps of encoding and decoding of texture mapping/video compression.**
Fig. 5.2. Illustration of a typical texture mapping in a triangular mesh. Vertices \(a, b\) and \(c\) are mapped to \(\rho(a), \rho(b)\) and \(\rho(c)\) respectively. Texels in the interior of the triangle \(\Delta \rho(a)\rho(b)\rho(c)\) can then be assigned to the interior of \(\Delta abc\). Note that interior angles are changed after the mapping, indicating the quasiconformality of the mapping.

Fig. 5.3. Texture images with arbitrary shapes are common in texture mapping techniques. With the texture coordinates, texels of the texture can be assigned to the corresponding position in the mesh. (A)Mesh of Zebra. (B)The texture maps of Zebra. (C)The texture images of all parts of Zebra. (D)Texture mapping result of Zebra.

This significantly reduces the storage requirement for the texture mappings. Note that in the case the texture image is of regular shape (e.g. 2D rectangle), storing the texture coordinates of the boundary vertices is not necessary. Instead, texture coordinates of 4 boundary vertices are sufficient.

The decoding algorithm is also straightforward. Given the Fourier coefficients \(c_{j,k}\) and the texture coordinates of the boundary vertices. Our goal is to reconstruct the texture mapping \(\rho_i\). Using the inverse Fast Fourier transform applied to Fourier coefficients \(c_{j,k}\) saved, the Beltrami representation \(\mu_i : M_i \rightarrow \mathbb{C}\) can be restored. With the Beltrami representation, the texture mapping \(\rho_i\) can be reconstructed using the Linear Beltrami Solver in Algorithm 4.2. More specifically, \(\rho_i\) can be computed by solving the linear systems \([4.13]\) and \([4.14]\), subject to the boundary condition given by the texture coordinates of the boundary vertices. Texture coordinates of \(M_i\) can then be obtained.

Figure 5.4 and 5.5 give a simple illustrations of the idea. Figure 5.4(A) show the surface mesh of a human face. Its texture mapping and texture image are shown in (B). Note that there is only one texture mapping in this case and that the texture image is of regular shape. The original textured surface is shown in (C). The proposed compression algorithm is applied, and the reconstructed textured surface after compression is shown in (D). Note that the compressed textured surface closely resembles to the original textured surface. Figure 5.5(A) shows the surface mesh of

\[1\] The texture mapping example is freely available on [http://www.kunzhou.net/tex-models.htm](http://www.kunzhou.net/tex-models.htm)
Beltrami representation and applications

Fig. 5.4. Illustration of the proposed compression scheme. No visible difference can be found after the reconstruction. (A) Original mesh of a man face. (B) The corresponding texture map and the texture. (C) The original texture mapping result of the man face. (D) The compressed texture mapping result of the man face.

Fig. 5.5. Illustration of the proposed compression scheme. The Bimba is first partitioned into four parts. Texture maps are then defined independently. (A) Mesh of the Bimba. (B) The texture map overlapped with the texture for part-1, 2, 3 and 4 of the Bimba. (C) The un-compressed texture mapping result. (D) The compressed texture mapping result.

the bimba surface. Its texture mappings and texture images are as shown in (B). There are totally four texture mappings and texture images with regular shapes. The original textured surface is as shown in (C). We apply the proposed compression algorithm to this example. The reconstructed textured surface after the compression is as shown in (D). In both example, only 1% of Fourier coefficients are saved. Again, the reconstructed textured surface closely resembles to the original one.

Figure 5.6 shows a more complicated example whose texture image is of irregular shape, which is common in real situation. (A) shows one part of the Buddha surface mesh. It is being mapped to a texture image of irregular shape. The texture mapping is shown in (B). The compressed texture mapping using our proposed algorithm is shown in (C). (D) shows the original textured surface. (E) shows the textured surfaces reconstructed from the compressed texture mapping in (C). Note that there is no visual difference between the compressed textured surface and the original textured surface.

Note that compression of texture mappings with their coordinate functions generally does not work. Figure 5.7(A) shows the original textured buddha surface. (B) shows the compressed textured surface using the Fourier compression of the Beltrami representation. The reconstructed textured surface closely resemble to the original one. (C) shows the compressed texture surface using the Fourier compression of the
Fig. 5.6. Texture map compression for one part of Buddha. (A) shows the mesh of Buddha part 1. (B) shows the original texture map of part 1. (C) shows the compressed texture map of part 1 encoded by algorithm 5.1. (D) and (E) show the original textured surface and the one reconstructed from the compressed texture map in (C).  

coordinate functions, with the same compression ratio as in (B). Distortion of the texture can clearly be observed.

Summary of the encoding and the decoding part of our proposed compression scheme are described as follows:

**Algorithm 5.1:** (Encoding of the texture coordinates)

**Input:** Surface mesh $M$; Texture coordinate function $f : M \rightarrow [0, 1] \times [0, 1]$

**Output:** Fourier coefficients $c_{j,k}$ in the truncated Fourier series; Coordinates of the boundary vertices of $T$.

1. Obtain the harmonic parametrization, $\phi : M \rightarrow D$, by solving Equation 4.2
2. Compute the Beltrami representation $\mu_f$ of $f$ using Algorithm 4.1
3. Compress the Beltrami representation $\mu_f$ using Algorithm 4.3
4. Store the Fourier coefficients $c_{j,k}$ in the truncated Fourier series and coordinates of the boundary vertices of $T$.

**Algorithm 5.2:** (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 $M$.

---

2. The texture mapping example is freely available on [http://www.kunzhou.net/tex-models.htm](http://www.kunzhou.net/tex-models.htm)
Beltrami representation and applications

Fig. 5.7. Comparison between the results of compressing 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.

|        | Vertices | Faces | RMSE(1%)   | CF(1%)   | RMSE(3%)   | CF(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 | 7.042e-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   |

1. Apply inverse fast Fourier transform on \( c_{j,k} \) to restore the Beltrami representation \( \mu_f \) defined on triangular faces of \( M \)
2. Parametrize the surface mesh \( M \) by the harmonic mapping \( \phi \) to the unit square through solving Equation 4.2.
3. Reconstruct the texture coordinates by using the Linear Beltrami solver (Equation 4.13 and 4.14) subject to the boundary condition given by the texture coordinates of the boundary vertices.

Quantitative experiments have been carried out to demonstrate the performance of our proposed algorithm. Experiments are performed on three surface meshes, namely, 'Susan', 'Buddha' and 'Zebra' (See Figure 5.9). 'Susan' is a simply-connected open surface with 5161 vertices. 'Buddha' and 'Zebra' are genus-0 closed surface 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 proposed algorithm with 1% and 3% of the Fourier coefficients (FC).

To measure the accuracy of the compression scheme, we compute the root mean
Fig. 5.8. Statistics of the displacement error of each vertices. (A) Buddha part-1. (B) Zebra part-1. (C) Zebra part-4. (D) Susan. The majority of the reconstructed texture coordinates has less than 1 relative percentage error.

### Table 5.2
Summary of the texture mapping compression result.

|                | Susan  | Buddha | Zebra  |
|----------------|--------|--------|--------|
| Number of vertices | 5161   | 15138  | 20157  |
| RMSE            | $5.103e^{-4}$ | $1.974e^{-3}$ | $6.44e^{-3}$ |
| Compression ratio | 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 |

The square error (RMSE), defined by

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} || f(v_i) - \tilde{f}(v_i) ||_1},$$

(5.2)

where $n$ stands for the number of vertices, $f$ and $\tilde{f}$ are the original and the reconstructed texture map 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 5.2. Results shows 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 Fourier coefficients are saved (see column 5). The compression ratios are shown in column 4 and 6, which illustrate our proposed algorithm can significantly reduce the storage memory for texture mappings. 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 5.8 presents the displacement error of the each reconstructed texture coordinates measured in the supreme-norm. It is observed that majority of the texture coordinates can be reconstructed almost losslessly.

In Table 5.2, we study the compression performance of the whole mesh for the three surfaces. For each surface, 1\% of Fourier coefficients are stored. The average RMSE is computed which is defined as:

$$\text{RMSE}_{\text{avg}} = \sum_{k=1}^{P} \frac{n_k}{N} \text{RMSE}(k),$$

where $P$ is the number of parts in surface mesh, $N$ is the total number of vertices, $n_k$ stands for the number of vertices in part $k$ of the partitioned mesh and $\text{RMSE}(k)$ is the corresponding $\text{RMSE}$ calculated by formula (5.2). Note that the average RMSE is small in all 3 cases. Therefore, there is no noticeable difference between the com-
pressed textured surface and the original one.

The actual storage memories needed are also listed. Note that after the compression, the storage memories of the texture surfaces are much less than that of the original data. The compression ratios are at least 12:1. This demonstrates the efficacy of our proposed method.

Finally, every smooth Beltrami coefficient corresponds to a smooth bijective quasi-conformal map. As a result, no abnormal texture (e.g. zaggy textures) would appear on the textured mesh. 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 5.10).

5.2. Video Compression. With the recent 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.

Different video compression algorithms have been proposed [42, 44, 43]. The basic idea of these algorithms is to remove temporal and spatial redundancies existing in a video sequence. For instance, the two images shown in Figure 5.11(A) and (B) are the 1\textsuperscript{st} and the 31\textsuperscript{st} frame of a video respectively. (C) shows the intensity difference between (A) and (B). The black regions correspond to the unchanged background. The unchanged background is an example of the temporary redundant information existing in the uncompressed video. In motion compensation techniques, instead of storing every image frames in the video, I-frame, P-frame and B-frame are introduced. Commonly, video frames are partitioned into frame sets called the Group of Picture (GOP):

\[ I_1B_1B_2P_1B_3B_4P_2B_5B_6P_3B_7B_8P_4B_9B_{10} \]

I-frame is the reference frame, which is stored in the compressed image form. P-frame is called the predictive frame. It is encoded as a motion vector field (MV field) together with a residual. The MV field creates a prediction depicting how pixels in previous frame move. This prediction is then subtracted from the original frame to obtain the residual image. If the prediction is successful, the residual image can be represented with fewer bits than that of the original frame. B-frame is called the bi-directional predictive frame. It is encoded as the residual of the predicted image, which is obtained from the interpolated MV fields from previous and future frames.

On average, P-frames contributes 50\% less storage than that of I-frames. However, the required storage memory for the MV field is still significant when considering HD
Fig. 5.11. Illustration of the redundant information appears in a video. (A) The 1st frame (B) The 31st frame. Area in the boxes are examples of duplicated information appear in the video. (C) The intensity difference between (A) and (B).

videos. For example, if a maximum of $4 \times 4$ block mode is used in a full HD $[1920 \times 1280]$ video, each P-frame requires $2.765 \times 10^6$ bits of memory to store the MV field. It thus calls for the need of compressing the MV field.

Suppose $F_1$ and $F_2$ are two image frames in a video sequence. The MV field $V : F_1 \rightarrow \mathbb{R}^2$ is a vector field defined on $F_1$. It describes how pixels in $F_1$ should move to get the estimation of the image frame $F_2$. In particular, the MV field $V$ can be considered as a mapping $T : F_1 \rightarrow F_2$ between $F_1$ and $F_2$ given by:

$$T(x, y) = (x, y) + V(x, y)$$

By introducing a Delaunay triangulation to $F_1$, $T$ can be viewed as a piecewise linear homeomorphism between triangular meshes. Hence, every MV field $V$ can be represented by the Beltrami representation $\mu_T$ of $T$. Note that the harmonic parameterization of the meshes are not required in this case, since the triangular mesh is already a 2D rectangle embedded in $\mathbb{R}^2$. Using Algorithm 4.3, the MV field $V$ can be compressed by performing the Fourier compression of $\mu_T$. Storing the Fourier coefficients of the truncated Fourier series of $\mu_T$ requires much less storage memories than that of the MV field itself.

The reconstruction of the MV field from the Fourier coefficients $c_{j,k}$ is straightforward. Following the idea of Section 4.3, we first carry out the inverse fast Fourier transform to restore the Beltrami coefficient $\mu_T$. Using the Linear Beltrami solver as described in Algorithm 4.2, $T$ and hence the corresponding MV field $V$ can be reconstructed efficiently and accurately.

The encoding and decoding of the compression algorithm can be summarized as follows.

**Algorithm 5.3** : (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 motion vector field in regular grid
2. Compute the Beltrami representation by Algorithm 4.1
3. Use Algorithm 4.3 to compress the Beltrami representation $\mu$ and store the coefficients $c_{j,k}$ in the truncated Fourier series.

**Algorithm 5.4** : (Decoding of the P-frame)
**Input** : Frame $F_1$; Coefficients $c_{j,k}$ in the truncated Fourier series
Fig. 5.12. 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 4th P-frames in each frame set. The lower rows show the corresponding compressed P-frames.3
Fig. 5.13. (A) The average PSNR between the uncompressed and the compressed P-frames for the four videos with difference percentage of Fourier coefficients saved. (B) The corresponding compression ratio obtained.

Fig. 5.14. PSNR between each uncompressed and compressed P-frames. (A) Flower 1. (B) Flower 2. (C) Heart 1. (D) Heart 2.

Output: Motion vector field $V$
1. Apply inverse fast Fourier transform to restore the Beltrami coefficient $\tilde{\mu}$
2. Perform Linear Beltrami solver as described in Algorithm 4.2 to obtain $(\tilde{x}, \tilde{y})$
3. Compute $V$ by $V(x, y) = (\tilde{x}, \tilde{y}) - (x, y)$

To study the performance of our proposed algorithm, experiments have been carried out on real videos. In our experiments, all B-frames in the GOP are taken away for simplicity. Instead, we consider the following frame set:

$I_1P_1P_2P_3P_4$

MV fields in each P-frames are obtained from the previous I-frame or P-frame, which are then compressed by our proposed algorithm. In order to study the performance of our algorithm without adding any residual, per-pixel MV field is used. It should be noted that the proposed algorithm can also be applied to block-based MV fields in
A closer look to the plots in Figure 5.14. Patterns of decreasing PSNR with a period of four frames can be observed. This is the consequence of the accumulation of errors from previous P-frame reconstruction. (A) Flower1. (B) Flower2. (C) Heart1. (D) Heart2.

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 5.12 shows some original P-frames of the 4 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 means, it demonstrates the effectiveness and accuracy of our proposed algorithm in compressing MV field.

We also examine the performance of our algorithm quantitatively. Figure 5.14 shows the peak signal-to-noise ratio (PSNR) between the reconstructed P-frames and the original P-frames. The PSNR is defined by

\[
PSNR = 10 \log_{10} \frac{255^2 NM}{\sqrt{MSE}},
\]

with

\[
MSE = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \left[ \tilde{P}(i,j) - P(i,j) \right]^2,
\]

where the size of the frames is \(M \times N\). As shown in Figure 5.13(A), the average PSNR
in all four cases are higher than 90 when 0.5\% Fourier coefficients are captured. 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 storage memory for the residual image will not be much affected after compression with our algorithm.

Figure 5.13(B) presents the corresponding compression ratio for different percentage of Fourier coefficients stored. When 0.5\% coefficients are saved, compression ratio of the MV field can be as high as 56 : 1. The high compression ratio of the MV field can effectively improve the compression ratio of the existing video compression algorithm such has MPEG and H.264. After the compression, the storage memory of the MV field is almost negligible compare 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 5.14 shows the PSNR between each pair of the compressed and original P-frame. It is observed that PSNR stays close to about 90. It means the compressed P-frames closely resemble to the original ones. Figure 5.15 shows the zoom-in of the plots in Figure 5.14. 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 to the overall result is negligible.

6. 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 storage 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 bijective surface map without distorting the bijectivity of the map. This significantly reduces the storage requirement for surface maps. In this paper, we proposed to apply 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 motion vector fields for video compression. After compressing the motion vector field, the compression ratio of the state-of-the-art video compression algorithms can be significantly improved. Experiments on real textured surfaces and videos demonstrate the effectiveness and efficacy of our proposed algorithms.

Acknowledgments. Lok Ming Lui is supported by RGC GRF (Project ID: 2130271) and CUHK Direct Grant (Project ID: 2060413).

REFERENCES
[1] Y. Keller and A. Averbuch, Fast Gradient Methods Based on Global Motion Estimation for Video Compression, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, No. 4, April 2003 pp. 117–134.

[2] S. Fukunaga, MPEG-4 video verification model version 16.0, ISO/IEC document JTC1/SC29/WG11 N3312, Mar. 2000.

[3] A. Tekalp, Digital Video Processing, Englewood Cliffs, NJ: Prentice-Hall, 1995.

[4] B. Lévy, S. Petitjean, N. Ray and J. Maillot, Least Squares Conformal Maps for Automatic Texture Atlas Generation, ACM SIGGRAPH conference proceedings, 2002.

[5] E. Zhang, K. Mischeleckow and G. Turk, Feature-based surface parameterization and texture mapping, ACM Transactions on Graphics (TOG), Volume 24 Issue 1, January 2005.

[6] S. Haker, S. Angenent, A. Tannenbaum, R. Kikinis, G. Sapiro, and M. Halle. Conformal surface parameterization for texture mapping, IEEE Transaction of Visualization and Computer Graphics, 6:181–189, 2000.

[7] S.C. Joshi and M.I. Miller, Landmark Matching via Large Deformation Diffeomorphisms, IEEE Transaction on Image Processing, Vol. 9, No. 8, August 2000, pp. 1357–1370.

[8] F. Gardiner and N. Lakic. Quasiconformal Teichmüller Theory. American Mathematics Society, 2000.

[9] B. Fischl, M. Sereno, R. Tootell, and A. Dale. High-resolution intersubject averaging and a coordinate system for the cortical surface. Human Brain Mapping, 8:272–284, 1999.

[10] 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 Transactions on Medical Imaging, 23(8):949–958, 2004.

[11] Y. Wang, L. M. Lui, X. Gu, K. M. Hayashi, T. F. Chan, A. W. Toga, P. M. Thompson, and S.-T. Yau. Brain surface conformal parameterization using riemann surface structure. IEEE Transactions on Medical Imaging, 26(6):853–865, 2007.

[12] X. Gu and S. Yau. Computing conformal structures of surfaces, Communication in Information System, 2(2):121–146, 2002.

[13] S. Joshi and M. Miller. Landmark matching via large deformation diffeomorphisms. IEEE Transactions on Image Processing, 9(8):1357–1370, 2000.

[14] M. K. Hurdal and K. Stephenson. Discrete conformal methods for cortical brain flattening, Neuroimage, 45:86–98, 2009.

[15] Y. Wang, L. Lui, T. Chan, and P. Thompson. Optimization of brain conformal mapping with landmarks. Proceeding in Medical Image Computing and Computer-Assisted Intervention - MICCAI 2005, pages 675–683, 2005.

[16] L. Lui, S. Thiruvenkadam, Y. Wang, T. Chan, and P. Thompson. Optimized conformal parameterization of cortical surfaces using shape based matching of landmark curves, Proceeding in Medical Image Computing and Computer-Assisted Intervention - MICCAI 2005, pages 494–502, 2008.

[17] L. Lui, S. Thiruvenkadam, Y. Wang, P. Thompson, and T. Chan. Optimized conformal surface registration with shape-based landmark matching, SIAM Journal of Imaging Sciences, 3(1):52–78, 2010.

[18] L. Lui, Y. Wang, T. Chan, and P. Thompson. Landmark constrained genus zero surface conformal mapping and its application to brain mapping research, Applied Numerical Mathematics, 57:847–858, 2007.

[19] L.M. Lui, T.W. Wong, X.F. Gu, T.F. Chan and S.T. Yau. Compression of Surface Diffeomorphism using Beltrami coefficient, IEEE Computer Vision and Pattern Recognition(CVPR), 2839-2846 (2010)

[20] 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, Journal of Scientific Computing, DOI 10.1007/s10915-011-9506-2

[21] O. Lehto and K. Virtanen. Quasiconformal Mappings in the Plane. Springer-Verlag, New York, 1973.

[22] R. Schoen and S.-T. Yau. Lectures on Differential Geometry. International Press of Boston, 1994.

[23] J. Foley, A. van Dam, S. Feiner and J. Hughes. Computer Graphics, Addison-Wesley Publishing, Massachusetts, 1992.

[24] P. S. Heckbert. Survey of texture mapping, IEEE Computer Graphics and Applications 6, No. 11, pages 5667, November 1986.

[25] L. Balmelli, G. Taubin and F. Bernardini. Space-optimised texture maps, Computer Graphics Forum, 2002, 21, (3), pp. 411420.

[26] C. Bennis, J.M. Vézien and G. Iglesias. Piecewise surface flattening for nondistorted texture mapping, Computer Graphics (SIGGRAPH95 Proceedings) 25, pp 237246, 1991.
[27] S.D. Ma and H. Lin. Optimal texture mapping. EUROGRAPHICS88, pp 421-428, September 1988.

[28] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery and W. Stuetzle. Multiresolution analysis of arbitrary meshes. Computer Graphics (SIGGRAPH95 Proceedings) 29, pp 173-182, 1995.

[29] A. W. F. Lee, W. Sweldens, P. Schroder, L. Cowsar, and D. Dobkin. Maps: multiresolution adaptive parameterization of surfaces. Computer Graphics (SIGGRAPH 98 Proceedings) 32, pp 951-964, 1998.

[30] B.-B. Chai, S. Sethuraman, H. S. Sawhney, and P. Hatrack. Depth map compression for real-time view-based rendering. Pattern Recognition Letters, 25:755-766, 2004.

[31] J. W. Ioup, M. L. Gendron, and M. C. Lohrenz. Vector map data compression with wavelets. Journal of Navigation, 53:437-449, 2000.

[32] A. Kolesnikov and A. Akimov. Distortion-constrained compression of vector maps. Proceedings of the 2007 ACM symposium on Applied computing, pages 8–12, 2007.

[33] J. Lewis and M. English. Compression of body surface potential maps using image compression techniques. Proceeding of Computers in Cardiology, pages 401–404, 1995.

[34] J. Stachera and P. Rokita. Normal map compression based on bct and wavelet coding. Proceeding of Real-Time Image Processing, 6811:68110S–68110S–8, 2008.

[35] M. Deering. Geometry Compression. SIGGRAPH’95, pp. 13-20, ACM, 1995.

[36] G. Taubin and J. Rossinac. Geometric compression through topological surgery. ACM Transactions on Graphics, 17(2):84-115, 1998.

[37] K. Iourcha, K. Nayak and Z. Hong. System and Method for Fixed-Rate Block-based Image Compression with Inferred Pixels Values. In US Patent 5,956,431 (1999).

[38] B. Levy and J.L. Mallet. Non-Distorted Texture Mapping for Sheared Triangulated Meshes. Computer Graphics (SIGGRAPH 98 Proceedings), 1998.

[39] B. Levy. Constrained Texture Mapping for Polygonal Meshes. Computer Graphics (SIGGRAPH 98 Proceedings), 28, 2001.

[40] S. Haker, S. Angenent, A. Tannenbaum, G. Sapiro, and M. Halle. Conformal surface parameterization for texture mapping. IEEE Transactions on Visualization and Computer Graphics, 6(2), April-June 2000.

[41] M.-H. Yoon, D.-O. Kim, R.-H. Park and S.W. Lee. Geometry-dependent texture map compression. Electron. Lett., Volume 46, Issue 1, p.4344, doi:10.1049/el.2010.2427.

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

[43] Y. Wang, J. Ostermann and Y.Q. Zhang, Video Processing and Communications. Prentice Hall, Upper Saddle River, 2002.

[44] D. LeGall. MPEG: a video compression standard for multimedia applications. Communication ACM, vol. 34, pp. 4658, Apr. 1991.

[45] M. T. Orchard. A comparison of techniques for estimating block motion in image sequence coding. Proceeding of SPIE Visual Communications and Image Processing 89, vol. 1199, pp. 248258.

[46] M. Ghanbari. The cross-search algorithm for motion estimation. IEEE Transaction in Communication, vol. 38, pp. 9505, July 1999.

[47] M. Ghanbari. Video Coding - an Introduction to Standard Codecs. The Institution of Electrical Engineers, Stevenage 1999.

[48] B. Girod. The Efficiency of Motion-Compensating Prediction for Hybrid Coding of Video Sequences. IEEE Journal on Selected Areas in Communications, Vol. 5, Issue 7, pp. 1140-1154, August, 1987.

[49] J. R. Jain and A. K. Jain. Displacement measurement and its application in interframe image coding. IEEE Transaction in Communication, vol. COM-29, pp. 17991808, Dec. 1981.

[50] T. Koga, K. Iimura, A. Hirano, Y. Iijima, and T. Ishiguro. Motion compensated interframe coding for video conferencing. Proceeding National Telecommunication Conference, pp. G5.3.15.3.5, Nov. 29Dec. 3, 1981.

[51] S. Zafar, Y.-Q. Zhang and J. S. Baras. Predictive block-matching motion estimation for TV coding Part I: Inter-block prediction, IEEE Transaction on Broadcasting, vol. 37, pp. 97101, Sept. 1991.

[52] Y.-Q. Zhang and S. Zafar. Predictive block-matching motion estimation for TV coding Part II: Inter-frame prediction, IEEE Transaction on Broadcasting, vol. 37, pp. 97101, Sept. 1991.

[53] B. Liu and A. Zaccarin. New fast algorithms for the estimation of block motion vectors, IEEE Transaction on Circuits System and Video Technology, vol. 3, pp. 148157, Apr. 1993.

[54] J. Chalidabhongse and C.-C. Jay Kuo. Fast Motion Vector Estimation Using Multiresolution-Spatio-Temporal Correlations. IEEE Transactions on Circuit and Systems for Video Tech-
[55] J. Munkberg, P. Clarberg, J. Hasselgren and T. Akenine-Möller. *High Dynamic Range Texture Compression*, AMC Trans. Graphics, vol. 25, no. 3, pp. 698-706, 2006.

[56] M. Isenburg and J. Snoeyink. *Compressing texture coordinates with selective linear prediction*, Proceedings of Computer Graphics International’03, pp.126-131, 2003.

[57] X. Gu and S. Yau. *Computational conformal geometry*, Advanced Lectures in Mathematics(ALM) vol. 3, International Press, Somerville, MA, 2008.