An RBF-Based Reparameterization Method for Constrained Texture Mapping

Hongchuan Yu, *Member*, *IEEE*, Tong-Yee Lee, *Senior Member*, *IEEE*, I-Cheng Yeh, Xiaosong Yang, Wenxi Li, and Jian J. Zhang

Abstract—Texture mapping has long been used in computer graphics to enhance the realism of virtual scenes. However, to match the 3D model feature points with the corresponding pixels in a texture image, surface parameterization must satisfy specific positional constraints. However, despite numerous research efforts, the construction of a mathematically robust, foldover-free parameterization that is subject to positional constraints continues to be a challenge. In the present paper, this foldover problem is addressed by developing radial basis function (RBF)-based reparameterization. Given initial 2D embedding of a 3D surface, the proposed method can reparameterize 2D embedding into a foldover-free 2D mesh, satisfying a set of user-specified constraint points. In addition, this approach is mesh free. Therefore, generating smooth texture mapping results is possible without extra smoothing optimization.

Index Terms—Foldover, constrained texture mapping, reparameterization.

1 Introduction

Texture mapping is an effective technique to enhance visual realism in computer graphics. Existing research has largely concentrated on producing planar parameterization [1], [2], [3], [4], [5], [6], [7] to embed a 3D mesh into a 2D texture space. This embedding is computed to minimize distortion between the 3D mesh and the 2D mapping in order to obtain visually pleasing results. However, adding user-specified positional constraints into such embedding is a more challenging task. For example, to create a texture mapping of a human face and reduce texture distortion, the animator has to ensure that the important feature points and lines (i.e., the eyes, nose, eyebrows, and lips) on the 3D model match those on the texture image. Usually, the image content is greatly distorted when the 3D mesh becomes compressed or stretched during 2D mapping.

The current production in practice is nearly completely manual. Once a texture map is generated by the animation software, the animator has to painstakingly tweak the unwrapped mesh (the 2D mesh obtained from parameterizing the input 3D model) on the texture plane to align its corresponding features to the 3D model. The animator manually moves multiple vertices around each feature to reduce texture distortion and to avoid mesh foldovers. This is a time-consuming task.

Several attempts have been made to formulate this problem as a constrained optimization problem [3], [4], [5], [7], where a feature in the texture image is matched to its counterpart on the 3D surface. Despite varying degrees of success, no robust solution has been found to avoid mesh foldovers during parameterization with positional constraints. In other words, one-to-one mapping between the 2D and 3D domains during the unwrapping process is necessary.

Without consideration of any internal constraints, a 3D surface can be embedded into a 2D parameterization domain using existing approaches, such as harmonic mapping [16]. The internal constraints can then be added into the 2D parameterization domain. The current paper focuses on the reparameterization of a 2D mesh to satisfy a set of user-defined internal constraints. Specifically, a foldover-free reparameterization method using radial basis functions (RBF) is developed. The major contributions of the current paper are as follows:

- To the best of our knowledge, the proposed method is the first RBF-based approach that ensures user-specified constraints are satisfied and that foldovers are avoided. An explicit mathematical condition guarantees that no mesh foldover is generated during the RBF reparameterization. This is called the *foldover-free condition*.
- The RBF-based method is a mesh-free approach. Thus, generating smooth texture mapping is possible without an extra computationally expensive smoothing optimization, as required in [3] and [8].
- To the best of our knowledge, the proposed method presents the first implementation of satisfying positional constraints without predefined fixed boundaries. Furthermore, the proposed method can handle models with interior boundaries (Fig. 6) without additional treatment, such as cutting the model into several pieces, as required in [8].

The rest of the current paper is organized as follows: Section 2 describes the most relevant related work. Section 3 presents an overview of the foldover-free RBF-based reparameterization method. Section 4 describes the numerical foldover-free condition in detail. In Section 5, the strategy of triangle subdivision is explained and several extreme scenarios for an accurate alignment are provided.

- *H. Yu, X. Yang, W. Li, and J.J. Zhang are with the National Centre for Computer Animation, Bournemouth University, Poole BH125BB, United Kingdom. E-mail: {hyu, xyang, wli, jzhang}@bournemouth.ac.uk.*
- *T.-Y. Lee and I.-C. Yeh are with the Department of Computer Science and Information Engineering, National Cheng-Kung University, Tainan, Taiwan 701-01, R.O.C. E-mail: tonylee@mail.ncku.edu.tw, ichenyeh@gmail.com.*

Manuscript received 13 Oct. 2010; revised 25 May 2011; accepted 26 May 2011; published online 16 June 2011.
Recommended for acceptance by B. Guo.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2010-10-0254.
Digital Object Identifier no. 10.1109/TVCG.2011.117.

Experimental results and discussions are given in Section 6. Conclusions and future work are presented in Section 7.

2 Related Works

Many surface parameterization methods for embedding a 3D surface onto a 2D parametric domain have been proposed [2], [6], [9], [10], [11], [12], [13], [19], [20], [21], [22]. Floater and Hormann summarized a detailed survey of surface parameterization [18]. Floater [2] and Sander et al. [6] targeted the validity of the resulting parameterization (e.g., bijective mapping). Attempts have also been made to minimize distortion according to different metrics [6], [9], [10], [11], [12], [13], [19], [20]. In general, previous methods can be roughly classified into two categories: the first seeks to embed a 3D mesh onto a 2D region with a specified convex boundary [2], [6], [9], [13], whereas the second attempts embeddings without fixed boundary constraints [11], [12], [14], [21], [22] to reduce parameterization distortion. Our proposed RBF-based method does not require fixed boundaries during the iterative reparameterization procedures.

A more challenging problem is computing texture coordinates to satisfy user-specified correspondence between the 3D model and texture image. However, this has not been given much consideration in the literature. Few studies have been conducted on meeting soft constraints [13], [15] (i.e., to satisfy the positional constraints approximately). Levy [5] and Desbrun et al. [12] proposed a least-squares system and Lagrange multipliers as solutions, respectively. However, these two methods fail to guarantee a bijective embedding. Zhang et al. [30] focused on a special case (i.e., deforming a patch by stretching its boundary). Occurrence of foldovers when internal positional constraints are added in the original patch, and whether these can converge to expected positions, were not clearly stated.

In contrast, hard constraints were studied in [3], [4], and [8] because a perfect texture alignment is essential at certain delicate areas of a mesh. Eckstein et al. [4] proposed a constrained simplification to align constraints, adding Steiner vertices to avoid foldovers. Theoretically, although the above method can handle large sets of constraints, it is extremely complicated and not very robust [3]. In addition, only simple examples were shown in [4]. Thus, whether the above method can handle more complicated constraints is not clear. Kraevoy et al. [3] and Lee et al. [8] performed embedding by adding a fixed rectangular virtual boundary, after which the Delaunay method was applied to triangulate the region between true and virtual boundaries. After aligning user-specified hard constraints, the embedding is usually highly distorted. Therefore, a postsmoothing procedure is required to reduce the distortion, adding to computation costs. In contrast, our proposed method can avoid such expensive postsmoothing steps based on the continuity of RBF function. Kraevoy et al. [3] failed to completely remove foldovers, because the consistent neighboring ordering was not considered in finding matching triangulations. Fujimura and Makarov [24] presented an image-warping method. To satisfy positional constraints, the Delaunay triangulation and edge swaps were repeatedly used in their work to avoid foldovers. However, edge swaps can damage the geometric surface when used to texture map a 3D mesh, as discussed in [4].

Similar to the current paper, Tang et al. [17] and Lee and Huang [25] proposed an RBF-based parameterization method. However, neither method guarantees a foldover-free parameterization, unlike the method presented in the current paper. Tiddeman et al. [27] applied the condition of positive Jacobian determinant [see (2) in Section 4] to remove foldovers in their image-warping application. This condition is well known in differential geometry to ensure one-to-one mapping [28]. The method starts from an initial dense mapping that is likely to contain foldovers. Foldovers are then removed by iteratively scaling the given mapping. However, dense mapping is difficult to establish beforehand. Moreover, the primary deficiency of this method is that the convergence cannot be guaranteed. In a given discrete setting, scaling a given dense mapping usually results in iterative step length toward zero quickly, as admitted by the authors. In a few extreme cases, the method cannot satisfy the specified positional constraints.

3 Algorithm Overview

3.1 Basic Idea and Motivation

The overview of the proposed algorithm is as follows. An input 3D surface is first embedded into a 2D convex domain with harmonic mapping [16]. A mathematical foldover-free condition (see Section 4) is derived, and incorporated into an RBF-based reparameterization algorithm. The algorithm then iteratively aligns user-specified positional constraints. The main idea is to first estimate the iterative step length (i.e., displacement) subject to the foldover-free condition, and then to successively approximate the desired positions through RBF-based deformation. In short, RBF is used to iteratively deform the 2D mesh to align user-specified constraints. With the foldover-free condition at each iterative step, the deformation is prevented from being overaggressive (i.e., to induce foldovers).

3.2 Iterative RBF-Based Reparameterization Procedure

For a given 2D mesh embedding S of R^2, a transformation T is a one-to-one mapping of points $X \in S$ onto another 2D parametric domain $U \in \Omega$ of R^2, with arbitrary m constraint point pairs $(X_i^* \leftrightarrow U_i^*)$

$$T : \begin{cases} X = (x, y)^T \in S \rightarrow U(X) = (u(X), v(X))^T \in \Omega \\ \text{subject to } U(X_i^*) = U_i^*, i = 1, \ldots, m. \end{cases}$$

The reparameterization algorithm is developed based on the RBF scheme. RBF ensures a smooth final parameterization due to its numerous excellent properties, such as being mesh-free and C^2 continuity. Moreover, the most important advantage is the suitability of RBF for implementation in a successive approximation. This can smoothly deform S to align user-specified constraints, as demonstrated later. The RBF-based method is reinforced with the proposed foldover-free condition to appropriately control the displacement of $X \in S$ at each iteration. The displacement of each point coordinate is computed with the RBF scheme to implement successive approximation

$$\Delta U = P(X) + \sum_{i}^{m} \boldsymbol{\lambda}_i \phi(\|X - C_i\|), \tag{1}$$

where the coefficient $\boldsymbol{\lambda}_i = (\lambda_u^i, \lambda_v^i)^T$ is a vector, $C_i = (c_x^i, c_y^i)^T$ denotes the constraint points, $\Delta U = (\Delta u, \Delta v)^T$, and $P(X)$ is an affine transformation $P(X) = \left(\begin{smallmatrix} a_1 & a_2 \\ b_1 & b_2 \end{smallmatrix}\right) X + \left(\begin{smallmatrix} a_3 \\ b_3 \end{smallmatrix}\right)$. Although various radial basis functions exist, thin plate spline (i.e., $\phi(r) = r^2 \ln r$) is adopted for its simplicity. The deformed U is

obtained by updating $U(X) = X + \Delta U$. For the next iteration, let $X \leftarrow U(X)$.

The following pseudocode gives an overview of the reparameterization algorithm. The algorithm is executed iteratively; the superscript j stands for the iteration index. Let the initial 2D mesh embedding be $S^{(0)}$, a set of user-specified constraint point pairs be $(C_i^{(0)}, C_i^{(*)}), i = 1, \ldots, m$, on S and U. C_i^* denotes the final constraint points whereas $C_i^{(j)}$ is the updated point per iteration.

Loop:

(1) Estimate the displacement bound δ by Eq.(6) (see Section 4) based on the configuration of the current m constraint points $C_i^{(j)}$;

(2) Compute the current m constraint point displacements by $\Delta C_i^{(j+1)} = \delta \frac{C_i^* - C_i^{(j)}}{\left\| C_i^* - C_i^{(j)} \right\|}$, such that $C_i^{(j+1)} = C_i^j + \Delta C_i^{(j+1)}$;

(3) If $\delta < \delta_{threshold}$, apply triangle subdivision (see Section 5) then go to Step (1); otherwise,

(4) Compute the displacements of the points on $S^{(j)}$ by Eq.(4) (see Section 4) based on all m updated $C_i^{(j+1)}$ and updating $S^{(j)} \rightarrow S^{(j+1)}$;

Repeat until $C_i^{(j)} = C_i^*$.

In this procedure, C_i^* denotes the desired positions. Initially, if C_i^* is applied to directly deform S in (1), the result is usually too aggressive and foldovers may occur. Therefore, in Step (1), a conservative displacement bound δ needs to be computed and used to ensure that $C_i^{(j+1)}$ is not overaggressive. Note that the iterative RBF-based reparameterization procedure can definitely change mesh S to a foldover-free state. However, the final positions may not align exactly with the constraints C_i^* in some extreme scenarios. This implies that the method is only able to handle soft constraints. To alleviate this problem (i.e., to approximate hard constraints as much as possible), the mesh in Step (3) is subdivided by adding extra Steiner vertices. For more details, see Section 5.

4 Foldover-Free Condition

4.1 Foldover-Free Condition

From a mathematical perspective, a "foldover-free" parameterization yields a "one-to-one" mapping between corresponding surfaces (or meshes) and parametric domains. In the present work, the initial 2D embedding of a 3D surface is given in advance. Focus is given on deforming this initial embedding with a set of internal constraint point pairs. This requires that the mapping T is globally univalent or "globally one-to-one" (i.e., the topology or the relationship between any pair of vertices in the mesh should remain unchanged before and after parameterization). Mathematically, this means that the determinant of the Jacobian matrix must always be positive [28]

$$\det(\nabla U) > 0. \tag{2}$$

According to the Gerschgorin circle theorem [26], a *sufficient* condition of satisfying (2) can be described as follows:

$$\begin{cases} \dfrac{\partial u}{\partial x} > \left| \dfrac{\partial u}{\partial y} \right| \\ \dfrac{\partial v}{\partial y} > \left| \dfrac{\partial v}{\partial x} \right| \end{cases}. \tag{3}$$

The geometric meaning of (3) is simply that the two vectors $\partial u/\partial(x, y), \partial v/\partial(x, y)$ are linearly independent of each other; thus, their included angle is less than π. The former is easy to understand. The latter implies that the right-hand rule in vector calculus is satisfied over the entire domain. Holding $\det(\nabla U) < 0$ at any point would result in left-handedness instead of right-handedness. This change would cause mesh foldover.

4.2 Iterative Step-Length Estimation

Our reparameterization algorithm employs an iterative framework and step length (i.e., displacement) size is estimated considering the condition of (3). Equation (1) must be rewritten to implement the procedure, such that the displacements of some points linearly depend on the constrained points. This implies that deformation of the mesh is achieved by adjusting the displacements of the constrained points in an iterative manner. A further expectation is that foldovers will be avoided by controlling the displacement of the constrained points in each iteration.

The RBF coefficients $(\boldsymbol{\lambda}_u, \boldsymbol{\lambda}_v, \mathbf{a}, \mathbf{b})$ of (1) are first computed, where $\mathbf{a} = (a_1, a_2, a_3)^T$ and $\mathbf{b} = (b_1, b_2, b_3)^T$. For a given set of constrained points and their displacements, this can be achieved by solving the following linear system:

$$K \begin{pmatrix} \boldsymbol{\lambda}_u \\ \mathbf{a} \end{pmatrix} = \begin{pmatrix} \Delta \mathbf{x}_c \\ 0 \end{pmatrix} \text{and } K \begin{pmatrix} \boldsymbol{\lambda}_v \\ \mathbf{b} \end{pmatrix} = \begin{pmatrix} \Delta \mathbf{y}_c \\ 0 \end{pmatrix},$$

where $K = \begin{pmatrix} \phi & P \\ P^T & 0 \end{pmatrix}$, $\phi_{ij} = \phi(\|C_i - C_j\|)$, and P contains the constrained points coordinates (i.e., $c_x, c_y, 1$) and the vectors $\Delta \mathbf{x}_c, \Delta \mathbf{y}_c$ hold the displacements of the constraint points as $\Delta \mathbf{x}_c = (\Delta c_x^1, \ldots, \Delta c_x^m)^T, \Delta \mathbf{y}_c = (\Delta c_y^1, \ldots, \Delta c_y^m)^T$. This can be expressed as follows:

$$\begin{pmatrix} \boldsymbol{\lambda}_u & \boldsymbol{\lambda}_v \\ \mathbf{a} & \mathbf{b} \end{pmatrix} = K^{-1} \begin{pmatrix} \Delta \mathbf{x}_c & \Delta \mathbf{y}_c \\ 0 & 0 \end{pmatrix}.$$

(For a detailed RBF computation, refer to [29].)

Substituting $(\boldsymbol{\lambda}_u, \boldsymbol{\lambda}_v, \mathbf{a}, \mathbf{b})$ into (1), the new expression is as follows:

$$\begin{cases} \Delta u = \mathbf{M}(X) K^{-1} \begin{pmatrix} \Delta \mathbf{x}_c \\ 0 \end{pmatrix} \\ \Delta v = \mathbf{M}(X) K^{-1} \begin{pmatrix} \Delta \mathbf{y}_c \\ 0 \end{pmatrix} \end{cases} \tag{4}$$

$$\mathbf{M}(X) = (\phi(\|X - C_1\|), \ldots, \phi(\|X - C_m\|), x, y, 1).$$

Note that (4) describes a linear system of the displacement of any X (i.e., $\Delta u, \Delta v$) and of the constraint points (i.e., $\Delta \mathbf{x}_c, \Delta \mathbf{y}_c$). Whether or not the resulting mesh satisfies the condition of (3) should depend on the configuration of the current constraint points (i.e., $\mathbf{M}(X)$ and K^{-1}), rather than their displacements, $\Delta \mathbf{x}_c, \Delta \mathbf{y}_c$. Moreover, during iterations, $\mathbf{M}(X)$ and K^{-1} are unfixed and depend on the configuration of the current constraint points. Hence, $\mathbf{M}(X)$ and K^{-1} are given focus. The derivatives of $\partial(u, v)/\partial(u, v)$ are computed as follows:

$$\begin{cases} \dfrac{\partial u}{\partial x} = 1 + \mathbf{M}_x K^{-1} \begin{pmatrix} \Delta \mathbf{x}_c \\ 0 \end{pmatrix} \\ \dfrac{\partial u}{\partial y} = \mathbf{M}_y K^{-1} \begin{pmatrix} \Delta \mathbf{x}_c \\ 0 \end{pmatrix} \\ \dfrac{\partial v}{\partial x} = \mathbf{M}_x K^{-1} \begin{pmatrix} \Delta \mathbf{y}_c \\ 0 \end{pmatrix} \\ \dfrac{\partial v}{\partial y} = 1 + \mathbf{M}_y K^{-1} \begin{pmatrix} \Delta \mathbf{y}_c \\ 0 \end{pmatrix}, \end{cases}$$

where $\mathbf{M}_x (\text{or} \mathbf{M}_y)$ denotes the partial derivatives of M(X).

Substituting the above derivatives into (3) yields

$$\begin{cases} 1 + \mathbf{M}_x K^{-1} \begin{pmatrix} \Delta \mathbf{x}_c \\ 0 \end{pmatrix} > \left| \mathbf{M}_y K^{-1} \begin{pmatrix} \Delta \mathbf{x}_c \\ 0 \end{pmatrix} \right| \\ 1 + \mathbf{M}_y K^{-1} \begin{pmatrix} \Delta \mathbf{y}_c \\ 0 \end{pmatrix} > \left| \mathbf{M}_x K^{-1} \begin{pmatrix} \Delta \mathbf{y}_c \\ 0 \end{pmatrix} \right|. \end{cases}$$

The regions defined by the above inequalities can be further described as follows:

$$\begin{cases} \Omega(\delta_u) = \left\{ (1+\alpha, \beta) : |\alpha| \le \left| \mathbf{M}_x K^{-1} \begin{pmatrix} \delta_u \\ \mathbf{0} \end{pmatrix} \right|, \right. \\ \quad \left. |\beta| \le \left| \mathbf{M}_y K^{-1} \begin{pmatrix} \delta_u \\ \mathbf{0} \end{pmatrix} \right|, 1 + \alpha > |\beta| \right\} \\ \Omega(\delta_v) = \left\{ (\alpha, 1+\beta) : |\alpha| \le \left| \mathbf{M}_x K^{-1} \begin{pmatrix} \delta_v \\ \mathbf{0} \end{pmatrix} \right|, \right. \\ \quad \left. |\beta| \le \left| \mathbf{M}_y K^{-1} \begin{pmatrix} \delta_v \\ \mathbf{0} \end{pmatrix} \right|, 1 + \beta > |\alpha| \right\}, \end{cases} \tag{5}$$

where (α, β) denotes a point in Ω, vector $\delta_u(, \delta_v)$ consists of the same value as $\delta_u = \max_i^m(|\Delta c_x^i|)(, \delta_v = \max_i^m(|\Delta c_y^i|))$, and m denotes the constraint points number. Fig. 2 illustrates the regions $\Omega(\delta_u), \Omega(\delta_v)$. The δ_u, δ_v are not constants, and depend on the displacements of the constraint points. Thus, the dashed line is used to highlight these undetermined boundaries.

The (3) condition implies that vectors $\partial u/\partial(x,y)$, $\partial v/\partial(x,y)$ should be linearly independent of each other. Fig. 2 intuitively illustrates this concept by the four tangent lines: l_1, l_2, l_3, l_4. For example, α and β should be outside the regions of sweeping l_1 to l_2 and l_3 to l_4. For simplicity, let $\delta_u = \delta_v = \delta$. The regions $\Omega(\delta_u), \Omega(\delta_v)$ would then have the same size in terms of (5). This will lead to the overlap of straight lines l_1 and l_2 (l_3 and l_4) and form two new dividing lines, which are $\alpha + \beta = 0$ and $\alpha - \beta = 0$ in Fig. 2. Line $\alpha + \beta = 0$ guarantees the included angle is less than π, and line $\alpha - \beta = 0$ guarantees the linear independence. Consequently, the condition of (3) can be reexpressed as

$$\left(\mathbf{M}_x K^{-1} \begin{pmatrix} \delta \\ \mathbf{0} \end{pmatrix} \right)^2 + \left(\mathbf{M}_y K^{-1} \begin{pmatrix} \delta \\ \mathbf{0} \end{pmatrix} \right)^2 \le \frac{1}{2}.$$

Notice that the possible values of $\partial u/\partial(x,y)(\text{or } \partial v/\partial(x,y))$ are assumed to be evenly distributed around the center of (1, 0) and (0, 1). This is because of various possible configurations of the constraint points (e.g., $\mathbf{M}_x, \mathbf{M}_y, K^{-1}$). Hence, circles are employed to estimate the domains of $\partial u/\partial(x,y)$ and $\partial v/\partial(x,y)$.

To satisfy the above inequality, let $|\mathbf{M}_x K^{-1}(\begin{smallmatrix} \delta \\ \mathbf{0} \end{smallmatrix})| + |\mathbf{M}_y K^{-1}(\begin{smallmatrix} \delta \\ \mathbf{0} \end{smallmatrix})| \le \frac{1}{\sqrt{2}}$.

Therefore, δ_u, δ_v are bounded by

$$\delta = \min_{X \in S} \delta(X), \tag{6}$$

where

$$\delta(X) = \frac{1}{\sqrt{2} \left| (\mathbf{M}_x + \mathbf{M}_y) K^{-1} \begin{pmatrix} \mathbf{1} \\ \mathbf{0} \end{pmatrix} \right|}$$

for all vertices X of S, and $\mathbf{1}$ denotes a $m \times 1$ vector filled with ones as entries. Equation (6) is called the *foldover-free condition*. Note that the bound δ of (6) only depends on the current constraint points configuration (i.e., $\mathbf{M}_x, \mathbf{M}_y, K^{-1}$). This implies that bound δ is independent of displacements of the constraint points. For any constraint point C_i, its displacement must satisfy $|\Delta c_x^i|, |\Delta c_y^i| \le \delta$. However, the displacements of the initial given constraint point pairs $\Delta C_i = C_i^* - C_i^{(0)}$ always exceed the bound δ in practice. Thus, the method has to be implemented in an iterative manner to reach the desired positions of the constraint points C_i^* without mesh foldover. A trajectory for each constraint point may be defined to enable each point to reach its individual desired position (i.e., $C_i^{(0)}, \ldots, C_i^{(n)} = C_i^*, i = 1, \ldots, m$). The displacements of the successive $C_i^{(j)}$ and $C_i^{(j+1)}, (|\Delta c_x^i|, |\Delta c_y^i|)$, can further be viewed as an iterative step length.

The iterative scheme for constrained texture mapping has been outlined. The iterative step length is adaptively estimated by the current constraint point configuration. Before proceeding further, the iterative step length δ of (6) is taken as an estimate of the lower bound for our purpose of foldover-free reparameterization. The condition of (6) is a sufficient condition [i.e., there may be an iterative step length δ beyond the estimate of (6) to yield a foldover-free solution]. Note that the presented foldover-free condition in (6) only eliminates all probable foldover cases in order to guarantee that the mesh topology is continually preserved. The goal of (6) is to guarantee that the domain is completely foldover free. Thus, (6) only provides an estimate of the lower bound.

Remark. A number of existing approaches [3], [8] have also been used to achieve a foldover-free solution by adding Steiner vertices and using edge-swap operations [8]. These are unlike our proposed method, which utilizes successive approximation. Compared to the previous approaches, our proposed method can generate a smooth solution without the need for postprocessing. In addition, because of the continuity of the RBF function, it leads to smaller distortion during reparameterization. These advantages over other methods are further illustrated in the experiment section.

5 Triangle Subdivision

In general, the proposed RBF-based reparameterization can effectively generate a continuous deformation to match positional constraints exactly. However, for extreme scenarios with large deformation, (1), together with the foldover-free condition [(6) in Section 4], may not always converge the mesh to the most ideal position. Looking at Fig. 3 for example, two constraint points are to be swapped while the other two points are fixed. Without triangle subdivision, although the scheme of (1) and (6) ensure that the mesh will converge to a foldover-free state (see Fig. 3b), the position is not ideal. This is a deficiency of our proposed scheme in (1) and (6).

New vertices should be added by subdividing the triangles to circumvent this issue. This step is similar to that presented in [3] and [8], in which extra Steiner vertices

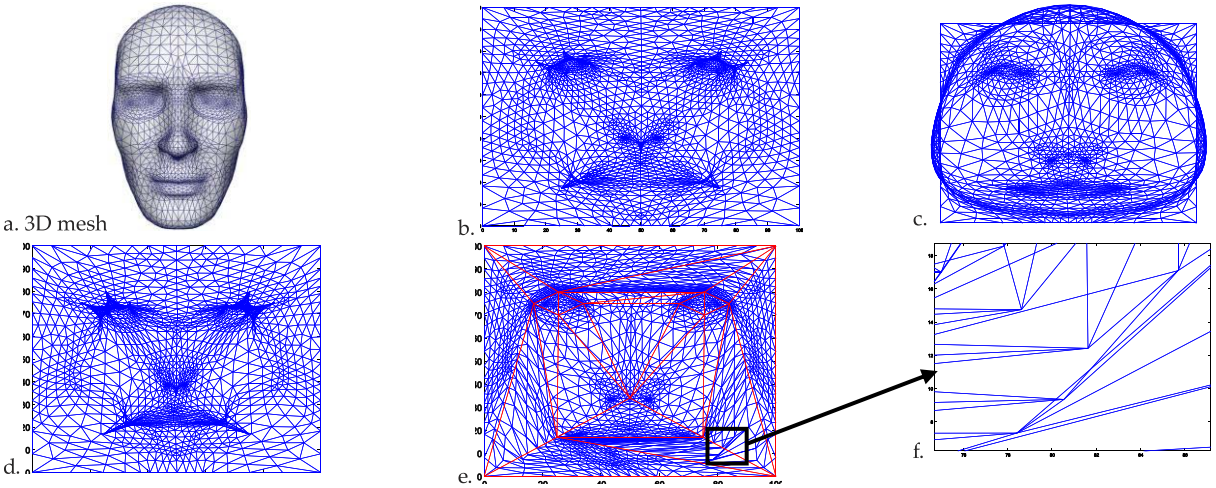

Fig. 1. Illustration of the foldover results using four recent parameterization methods, and considering internal constraints (Note: the mesh details can be seen more clearly by zooming in on the document): (a) a 3D mesh; (b) least squares meshes [23]; (c) RBF-based embedding [17]; (d) harmonic mapping [16], [18]; (e) Delauney triangulation-based mapping [3] (the red lines mark the boundaries of the triangle patches within which there is no foldover; however, foldover triangles can be observed around the red lines.); and f) inset showing the details of distortion around the red line.

are added. The basic idea of the subdivision strategy in the present study is to first determine the potential folding vertices, and then to identify the edges that the vertices will most likely cross. Thus, the triangles sharing these edges can be subdivided by adding new vertices around the potential folding vertices. The underlying idea is very simple: to approximate the continuous implicit function (i.e., RBF) by local upsampling. More sampling points provide more freedom and the higher the probability that foldovers could be avoided. The iterative step lengths $\delta(X)$ is estimated with (6) for all vertices, to determine the potential folding vertices when their $\delta(X)$s are below an empirically selected threshold $\delta_{threshold}$. The approach is summarized as follows: Assume N selected folding vertices:

Determining the Most Probable Edges

DO $i = 1, N$,

(1) Extract the 1-ring of the selected vertex v_i, then compute the probable location v_i' of v_i by Eq.(4) (Section 4)◊ with 2–3 times the threshold $\delta_{threshold}$ (i.e., setting the elements of vectors $\Delta\mathbf{x}_c$ and $\Delta\mathbf{y}_c$ with the same value as $2\delta_{threshold}$ or $3\delta_{threshold}$;

(2) Determine the 1-ring edge of v_i that intersects with line of $\overline{v_i v_i'}$. This edge is called the most probable edge for v_i;

(3) Bisect the selected edge. The midpoint is then added to the mesh as a new vertex.

END DO

◊Equation 4 is another expression of Eq.(1) because Eq.(4) offers a linear expression about the displacement of the current constraint points.

6 Experiments and Discussions

In this section, the proposed method is applied to a number of examples to evaluate its validity, efficiency, and robustness. For simplicity, the 2D meshes are normalized in $[0,1]\times[0,1]$ domain, and the texture images are similarly normalized, regardless of the aspect ratio. Based on this normalization, the threshold used in the algorithm of Section 3 can be preset without further tuning.

6.1 Foldover

Fig. 1 shows the results produced using several established methods [3], [17], [18], [23]. As shown in the figure, the methods are incapable of completely circumventing foldovers during the reparameterization process. The first experiment in the present study is to test the proposed method on the same head model as shown in Fig. 1a for comparison purpose. The initial 2D mesh (i.e., embedding or parameterization) obtained by conventional harmonic mapping is shown in Fig. 4a. Fig. 4 shows the results with different iterations of the RBF-based reparameterization. In Fig. 4a, red stars mark the constraints that need to move to the points circled in white. No foldover triangles occur

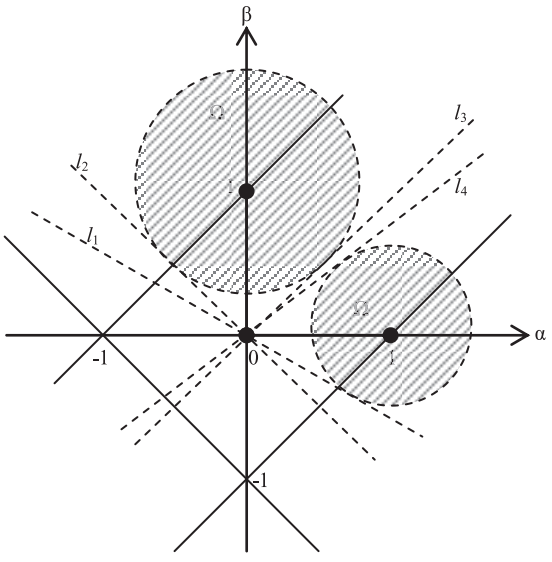

Fig. 2. Illustration of the condition of (3). The dashed lines denote the undetermined boundaries.

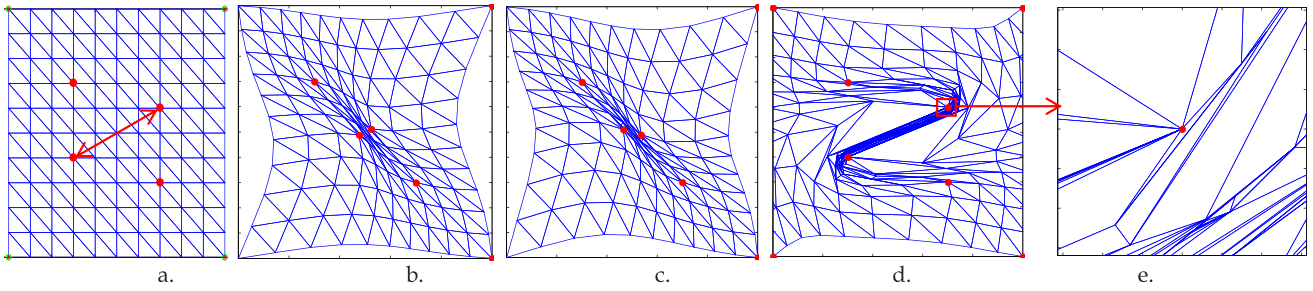

Fig. 3. Illustration of convergence in an extreme case: (a) the initial configuration of constraint points; (b) the result without triangle subdivision; (c) the intermediate result of rotating displacement vectors; (d) the final result using triangle subdivision and displacement rotation; and (e) the zoomed-in image corresponding to the selected region.

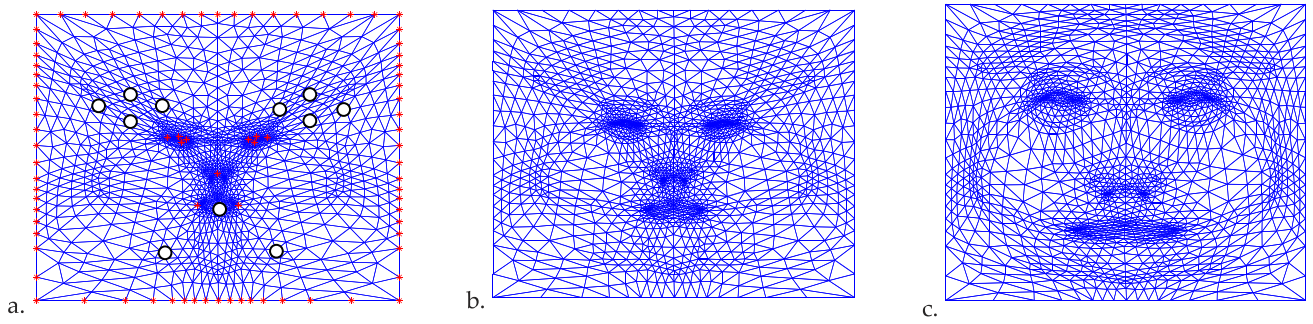

Fig. 4. Illustration of the iterative results of the proposed method: (a) initial mesh with constraint point pairs; (b) three iterations; and (c) five iterations.

during the iterations; thus, the internal constraints are satisfied. Note that during the iterative reparameterization process, the boundary of the 2D parameterized mesh does not have to be fixed on the initial predefined convex domain. Readers are referred to the accompanying video (at http://nccastaff.bournemouth.ac.uk/jzhang/projects.htm). This distinct advantage offers more freedom than the other methods to reduce mesh distortion.

6.2 Comparison of Experimental Results

The techniques proposed in [3] and [8] represent state-of-the-art methods in texture mapping subject to hard constraints. Lee et al. [8] experimentally showed that their method is able to handle challenging examples and generate satisfactory results. Therefore, in the current paper, the proposed algorithm is compared with the work of Lee et al. [8], tested on the same set of models. A further test was performed with a chessboard texture for smoothness comparison. A visual comparison shows that the smoothness of using the proposed approach is much better than that in [8] (Fig. 5, fourth column). In particular, the areas of the constraint points are smoother with the proposed method. This is because the previous method [8] cannot ensure smoothness of deformation in such areas. As a result, no further postprocessing for smoothing is necessary in the implementation of the proposed algorithm. In [3] and [8], this required postprocessing time usually takes much longer than that of the feature matching process, and becomes the bottleneck of the entire algorithm. However, their results are not very satisfactory without such postprocessing (Fig. 5, second column).

Moreover, to quantitatively study the distortion of reparameterization, the stretch metrics defined in [6] are used. The L-2 norm is used to measure the overall stretch of the parameterization, whereas the L-Inf measures the greatest stretch. Good parameterization is expected to have very small L-2 and L-Inf. These two metrics are used to measure distortion of all the examples in Fig. 5 (see Table 1). The proposed approach performs significantly better in most cases than that of [8], even with their smoothing process.

Furthermore, the proposed approach is capable of handling special models that have more than one border. Fig. 6 shows an example of texture mapping the photograph of an orangutan onto a 3D human head model with three boundaries. This figure shows that the proposed method produces a very smooth parameterization while keeping the two interior boundaries (i.e., the eyes). Applying previous methods [8] to this example would usually require extra treatment, such as cutting it into several pieces to ensure that each piece has no interior boundaries. The proposed approach is essentially a mesh-free method and does not need new any additional treatment.

6.3 Complexity Analysis

The core advantage of the proposed RBF-based reparameterization is that the RBF coefficients are updated at every iteration. The main computation cost, $O(2M^3)$, is to determine the inverse of a real symmetric matrix, where M is the number of the constraint points. The time complexity can be estimated as $O(2KM^3)$, where K denotes the number of iterations. Furthermore, considering the triangle subdivision procedure, computing the estimated iterative step lengths using (6) at every iteration is necessary. This will cost $O(N)$ each time, where N denotes the number of vertices on the mesh. The time for triangle subdivision is nearly fixed for each selected folding vertex. At each iteration, the running time of the triangle subdivision depends on the number of selected folding vertices m, which is generally much fewer than N. Therefore, the total time cost is $O(K(2M^3 + N + m))$. The majority of the time spent is on the computation of matrix inverse when there are a number of constrained points. The time spent for triangle subdivision is not an issue.

All the experiments were conducted with Matlab on an Intel Pentium 4 3.2 GHz PC with 1 GB of RAM. Table 2 shows the running time of all the examples in Fig. 5 using the

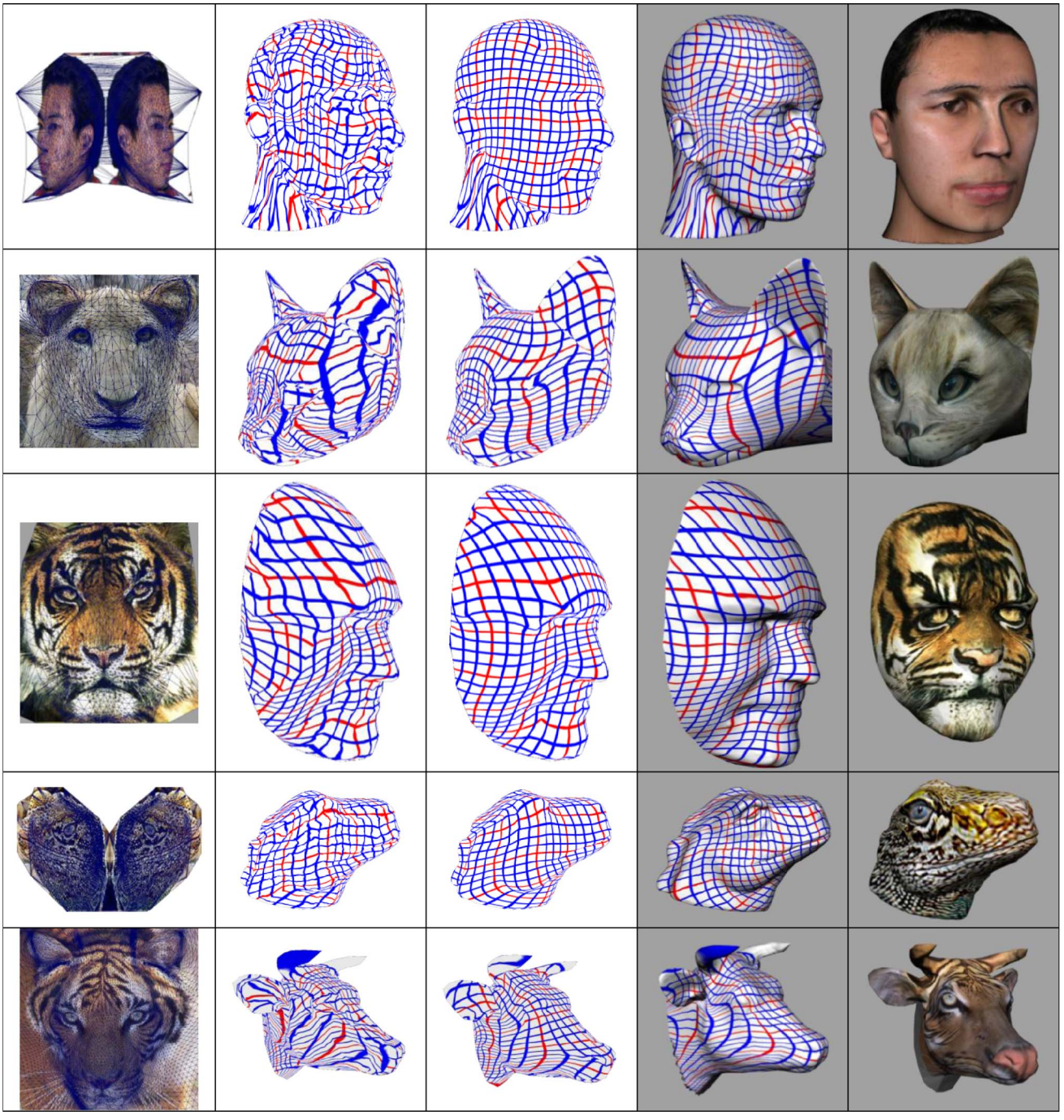

Fig. 5. Smoothness comparison. The first column shows the texture using the proposed approach. The second column shows the results without the postprocessing procedure [8]. The third column shows the results with the postprocessing procedure [8]. The fourth column shows the results using our proposed approach. The fifth column shows the final texture mapping results using the proposed approach.

proposed approach, which usually converges around 5-8 iterations. Each example usually takes only several seconds to compute with the proposed method because postprocessing for mesh smoothing is unnecessary. In [8], the postprocessing takes more than 1 minute to obtain the result.

6.4 Limitations

The proposed method deals with soft constraints. In Section 5, a subdivision approach is proposed to increase the chances of exactly matching the desired positional constraints. However, this simple approach has its limitations. For example, in Fig. 3a, two constrained points can move close to each other, but not reach the desired positions, even when the triangle subdivision strategy is applied. The displacement vectors can be rotated to circumvent this issue. The distance between any two constrained points is taken, and the displacement vectors of the two selected constrained points are rotated 90 degree clockwise. Fig. 3c shows the intermediate result of rotating the displacement vectors. Figs. 3d and3e show the convergence result with triangle subdivision and displacement rotation, where the desired deformation is achieved without triangle foldover. Thus, the above issue is successfully addressed. However, rotating the displacement vectors might fail if too many constrained points crowd together.

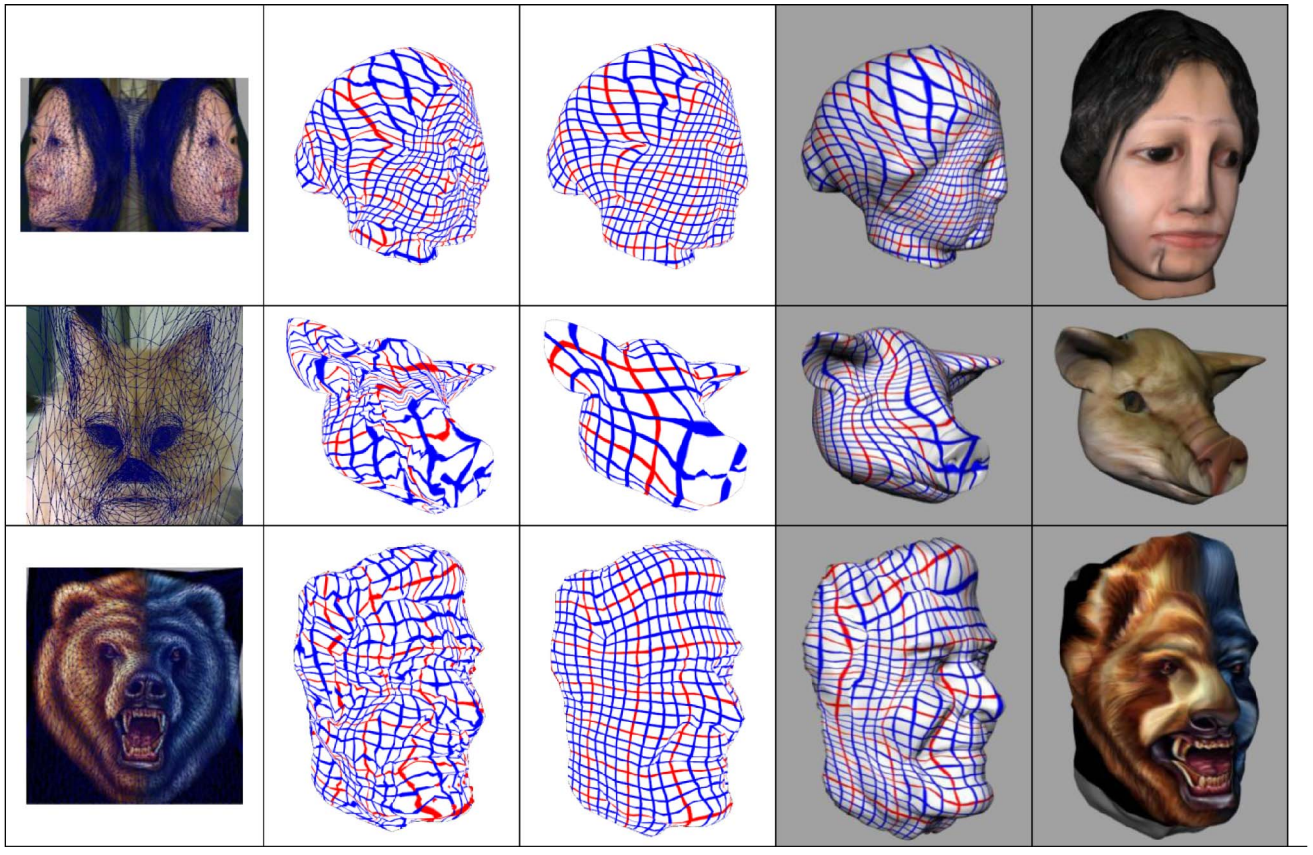

Fig. 5. (Continued).

TABLE 1
Distortion Metrics of Texture Mapping Examples in Fig. 5

Examples	Row 1	Row 2	Row 3	Row 4	Row 5	Row 6	Row 7	Row 8
L-2	2.707	830.1	1.317	1.411	1523.1	1.799	2.781	2.206
L-Inf	32.901	74714.16	4.406	7.079	13083071.1	9.785	163.521	38.311
L-2 [8]	2.148	4773.1	631.2	631.2	51804.12	1.623	17.413	1.105
L-Inf [8]	193.357	65477071.2	24732.4	24731.4	8751806.2	22.334	1436.4	4.016

We highlight the cases that the performance of our proposed approach is worse than that of [8] by shading.

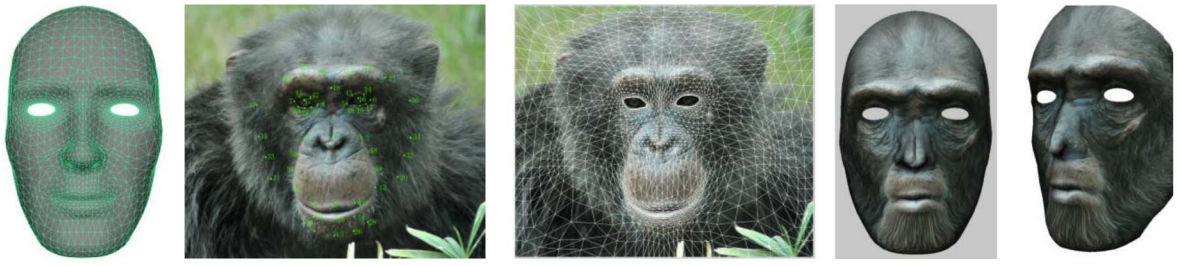

Fig. 6. Illustration of texture mapping with two interior boundaries.

Fortunately, such extreme cases are rarely seen in texture mapping applications. Another limitation is that the convergence of the proposed iterative algorithm has not been proven. Although this is not an empirical issue, ideally, mathematical comprehensive proof should still be given. This will be studied in future work.

Table 2 shows the results of the triangle subdivision of all the examples in Fig. 5. In Row 5, the number of the added vertices in the proposed approach is greater than that in [8]. This implies that the simple subdivision approach adds multiple redundant vertices. Looking at Fig. 6, the number of added vertices may depend on the level of smoothness and distortion because additional vertices are necessary for smoothness and low distortion. Nevertheless, the newly added vertices only increase the vertex number N on the mesh, rather than the constraint vertex number M or the selected folding vertex number m. Therefore, this does not result in a visible increase of total running time.

Further extension of the triangle foldover issue will incur another research focus: global self-intersection, that the boundary intersects itself. This still remains challenging. We

TABLE 2
Statistics of Texture Mapping Examples in Fig. 5

Examples	Row 1	Row 2	Row 3	Row 4	Row 5	Row 6	Row 7	Row 8
#Vertices	5184	1770	1808	10017	10736	4149	1772	1657
#Triangles	10354	3526	3602	20008	21404	8284	3450	3300
#Features	83	24	25	54	32	71	21	27
#Added Vertices	0	60	0	0	1556	0	2	0

TABLE 2 (Continued).

#Added Vertices [8]	183/38	77/7	128/10	94/34	127/8	122/25	383/21	69/2
Time (s)	8.94	5.12	3.86	13.71	16.93	7.21	4.08	3.45

Row 6 shows the numbers of the added points before and (/) after mesh optimization [8].

think, Jacobian constraint of (2) cannot sufficiently prevent the global self-intersection.

7 Conclusion and Future Work

An RBF-based reparameterization method for texture mapping with positional constraints is presented in the current paper. The proposed method can also avoid mesh foldovers. A mathematical condition has been formulated, called the foldover-free condition, that guarantees that the connection relationship of a mesh is maintained during the RBF-based reparameterization process. Basically, the meshes are iteratively moved to satisfy positional constraints. The foldover-free condition at each iteration provides the estimate of the iterative step length.

The proposed method does not require predefined restrictions in the implementation, such as fixing the boundaries of the mesh. This adds greater freedom for reducing distortion than do other methods [3], [8], and the resulting 2D parameterization has smaller distortion. In addition, the method is a mesh-free approach, allowing direct treatment of multiborder topology. Complexity analysis suggests a low computation cost, which depends mainly on the number of the vertices.

However, the need for triangle subdivision for some extreme cases is a limitation of the algorithm. This leads to the addition of redundant vertices. New methods to reduce the redundant vertices with a minimum loss of smoothness will be developed in the future work. In addition, the analysis of the time complexity indicates that running time could rapidly increase when adding a number of constraint vertices. The primary performance cost is to compute the inverse of the large symmetric matrix. In future, a GPU-based algorithm will be developed to speed up the computation.

Acknowledgments

This work is supported in part by National Science Council (contracts NSC-100-2628-E-006-031-MY3, and NSC-100-2221-E-006-188-MY3), Taiwan.

References

[1] M. Desbrun, M. Meyer, and P. Alliez, "Intrinsic Parameterizations of Surface Meshes," *Computer Graphics Forum,* vol. 21, no. 3, pp. 209-218, 2002.

[2] M.S. Floater, "Parameterization and Smooth Approximation of Surface Triangulations," *Computer Aided Geometric Design,* vol. 14, no. 3, pp. 231-250, 1997.

[3] V. Kraevoy, A. Sheffer, and C. Gotsman, "Matchmaker: Constructing Constrained Texture Maps," *ACM Trans. Graphics,* vol. 22, no. 3, pp. 326-333, 2003.

[4] I. Eckstein, V. Surazhsky, and C. Gotsman, "Texture Mapping with Hard Constraints," *Computer Graphics Forum,* vol. 20, no. 3, pp. 95-104, 2001.

[5] B. Lévy, "Constrained Texture Mapping for Polygonal Meshes," *Proc. ACM SIGGRAPH '01,* pp. 417-424, 2001.

[6] P.V. Sander, J. Snyder, S.J. Gortler, and H. Hoppe, "Texture Mapping Progressive Meshes," *Proc. ACM SIGGRAPH '01,* pp. 409-416, 2001.

[7] K. Zhou, X. Wang, Y. Tong, M. Desbrun, B. Guo, and H.-Y. Shum, "TextureMontage: Seamless Texturing of Arbitrary Surfaces from Multiple Images," *ACM Trans. Graphics,* vol. 24, no. 3, pp. 1148-1155, 2005.

[8] T. Lee, S. Yen, and I. Yeh, "Texture Mapping with Hard Constraints Using Warping Scheme," *IEEE Trans. Visualization and Computer Graphics,* vol. 14, no. 2, pp. 382-395, Mar./Apr. 2008.

[9] M. Eck, T. Derose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution Analysis of Arbitrary Meshes," *Proc. ACM SIGGRAPH '95,* pp. 173-182, 1995.

[10] S. Haker, S. Angenent, A. Tannenbaum, R. Kikinis, G. Sapiro, and M. Halle, "Conformal Surface Parameterization for Texture Mapping," *IEEE Trans. Visualization and Computer Graphics,* vol. 6, no. 2, pp. 181-189, Apr.-June 2000.

[11] B. Levy, S. Petitjean, N. Ray, and J. Maillot, "Least Squares Conformal Maps for Automatic Texture Atlas Generation," *ACM Trans. Graphics,* vol. 21, no. 3, pp. 362-371, 2002.

[12] M. Desbrun, M. Meyer, and P. Alliez, "Intrinsic Parameterizations of Surface Meshes," *Computer Graphics Forum,* vol. 21, no. 3, pp. 210-218, 2002.

[13] B. Levy and J.L. Mallet, "Non-Distorted Texture Mapping for Sheared Triangulated Meshes," *Proc. ACM SIGGRAPH '98,* pp. 343-352, 1998.

[14] P. Sander, S. Gortler, J. Snyder, and H. Hoppe, "Signal Specialized Parameterization," *Proc. Eurographics Workshop Rendering,* 2002.

[15] M. Cabral, S. Lefebvre, and C. Dachsbacher, "Structure-Preserving Reshape for Textured Architectural Scenes," *Proc. EUROGRAPHICS Conf.,* pp. 469-480, 2009.

[16] Y. Guo, J. Wang, H. Sun, X. Cui, and Q. Peng, "A Novel Constrained Texture Mapping Method Based on Harmonic Map," *Computer and Graphics* vol. 29, no. 6, pp. 972-979, 2005.

[17] Y. Tang, J. Wang, H.J. Bao, and Q.S. Peng, "RBF-Based Constrained Texture Mapping," *Computers and Graphics,* vol. 27, no. 3, pp. 415-422, 2003.

[18] M.S. Floater and K. Hormann, "Surface parameterization: A Tutorial and Survey," *Advances in Multiresolution for Geometric Modeling,* N.A. Dodgson, M.S. Floater, and M.A. Sabin, eds., pp. 157-186, Springer, 2005.

[19] S. Yoshizawa, A. Belyaev, and H.P. Seidel, "A Fast and Simple Stretch-Minimizing Mesh Parameterization," *Proc. Int'l Conf. Shape Modeling and Applications (SMI '04),* 2004.

[20] D. Piponi and G. Borshukov, "Seamless Texture Mapping of Subdivision Surfaces by Model Pelting and Texture Blending," *Proc. SIGGRAPH '00,* pp. 471-478, 2000.

[21] A. Sheffer and E. Sturler, "Parameterization of Faceted Surfaces for Meshing Using Angle Based Flattening," *Eng. with Computers,* vol. 17, no. 3, pp. 326-337, 2001.
[22] A. Sheffer, B. Levy, M. Mogilnitsky, and A. Bogomyakov, "ABF++: Fast and Robust Angle Based Flattening," *ACM Trans. Graphics,* vol. 24, no. 2, pp. 311-330, 2005.
[23] O. Sorkine and D. Cohen-Or, "Least-Squares Meshes," *Proc. Int'l Conf. Shape Modeling and Applications,* 2004.
[24] K. Fujimura and M. Makarov, "Foldover-Free Image Warping," *Graphical Models and Image Processing,* vol. 60, no. 2, pp. 100-111, 1998.
[25] T.-Y. Lee and P.H. Huang, "Fast and Institutive Polyhedral Morphing Using SMCC Mesh Merging Scheme," *IEEE Trans. Visualization and Computer Graphics,* vol. 9, no. 1, pp. 85-98, Jan.-Mar. 2003.
[26] G.H. Golub and C.F. Van Loan, *Matrix Computations,* p. 320. Johns Hopkins Univ. Press, 1996.
[27] B. Tiddeman, N. Duffy, and G. Rabey, "A General Method for Overlap Control in Image Warping," *Computer and Graphics,* vol. 25, no. 1, pp. 59-66, 2001.
[28] G.H. Meisters and C. Olech, "Locally One-to-One Mappings and a Classical Theorem on Schlicht Functions," *Duke Math. J.* vol. 30, no. 1, pp. 63-80, 1963.
[29] J.C. Carr, R.K. Beatson, J.B. Cherrie, T.J. Mitchell, W.R. Fright, B.C. McCallum, and T.R. Evans, "Reconstruction and Representation of 3D Objects with Radial Basis Functions," *Proc. ACM SIGGRAPH '01,* pp. 67-76, Aug. 2001.
[30] E. Zhang, K. Mischaikow, and G. Turk, "Feature-Based Surface Parameterization and Texture Mapping," *ACM Trans. Graphics,* vol. 24, no. 1, pp. 1-27, Jan. 2005.

Hongchuan Yu received the PhD degree in computer vision from the Institute of Intelligent Machine, Chinese Academy of Sciences, Beijing, PRC, in 2000. He is currently a lecturer at the National Centre for Computer Animation, Bournemouth University, Poole, United Kingdom. His research interests include geometry modeling and rendering, face recognition and expression synthesis, and image processing. He is a member of the IEEE.

Tong-Yee Lee received the PhD degree in computer engineering from the Washington State University, Pullman, in May 1995. He is currently a distinguished professor in the Department of Computer Science and Information Engineering, National Cheng-Kung University, Tainan, Taiwan, R.O.C. He leads the Computer Graphics Group, Visual System Laboratory, National Cheng-Kung University (http://graphics.csie.ncku.edu.tw/). His current research interests include computer graphics, nonphotorealistic rendering, medical visualization, virtual reality, and media resizing. He also serves on the editorial boards of the *IEEE Transactions on Information Technology in Biomedicine, the Visual Computer, and the Computers and Graphics* journal. He served as a member of the international program committees of several conferences including the IEEE Visualization, the Pacific Graphics, the IEEE Pacific Visualization Symposium, the IEEE Virtual Reality, the IEEE-EMBS International Conference on Information Technology and Applications in Biomedicine, and the International Conference on Artificial Reality and Telexistence. He is a senior member of the IEEE and the IEEE Computer Society and a member of the ACM.

I-Cheng Yeh received the BS degree from the Department of Computer and Information Science, National Chiao Tung University, Taiwan, in 2004. He is currently working toward the PhD degree in the Department of Computer Science and Information Engineering, National Cheng-Kung University. His research interests are computer graphics, including mesh parameterization, shape manipulation, and camera planning.

Xiaosong Yang received the BS and MS degrees in computer science from the Zhejiang University, PRC, in 1993 and 1996, respectively, and the PhD degree in computing mechanics from Dalian University of Technology, PRC, in 2000. He is a lecturer at the National Centre for Computer Animation, Bournemouth Media School, Bournemouth University, United Kingdom. He worked as a postdoc (2000-2002) in the Department of Computer Science and Technology of Tsinghua University for two years, and as a research assistant (2001-2002) at the "Virtual Reality, Visualization and Imaging Research Centre" of the Chinese University of Hong Kong. His research interests include 3D modeling, animation, real-time rendering, virtual reality, virtual surgery simulation, and computer-aided design.

Wenxi Li received the MA degree from the Animation School of Communication University of China in 2007. He is currently working toward the PhD degree at the National Center for Computer Animation in Bournemouth University. His research interests are computer graphics, animation, virtual surgery, and virtual clothing.

Jian J. Zhang is currently a professor of computer graphics at the National Centre for Computer Animation, Bournemouth University, United Kingdom, where leads the Computer Animation Research Centre. He is also a cofounder of the UK's Centre for Digital Entertainment, which received an initial funding of over six million GBP from the Engineering and Physical Sciences Research Council. His research focuses on a number of topics relating to 3D virtual human modeling, animation and simulation, including geometric modeling, rigging and skinning, motion synthesis, deformation, and physics-based simulation.