Foldover-Free Mesh Warping for Constrained Texture Mapping

Yuewen Ma, Jianmin Zheng, and Jian Xie

Abstract—Mapping texture onto 3D meshes with positional constraints is a popular technique that can effectively enhance the visual realism of geometric models. Such a process usually requires constructing a valid mesh embedding satisfying a set of positional constraints, which is known to be a challenging problem. This paper presents a novel algorithm for computing a foldover-free piecewise linear mapping with exact positional constraints. The algorithm begins with an unconstrained planar embedding, followed by iterative constrained mesh transformations. At the heart of the algorithm are radial basis function (RBF)-based warping and the longest edge bisection (LEB)-based refinement. A delicate integration of the RBF-based warping and the LEB-based refinement provides a *provably* foldover-free, smooth constrained mesh warping, which can handle a large number of constraints and output a visually pleasing mapping result without extra smoothing optimization. The experiments demonstrate the effectiveness of the proposed algorithm.

Index Terms—Triangular meshes, texture, RBF-based interpolation, foldover-free mapping, adaptive refinement.

1 Introduction

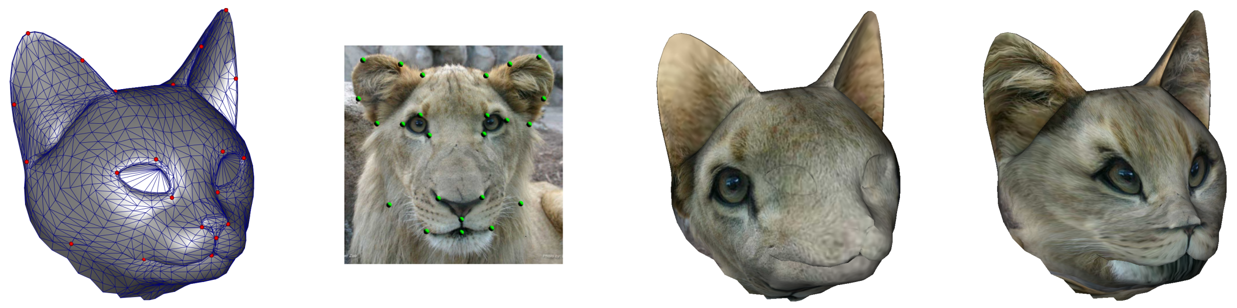
Texture mapping is a popular technique in computer graphics to add visual detail contained in a texture to the rendering of 3D geometric models. A common approach to painting a texture onto a 3D mesh model is to map the 3D mesh to a plane through a planar parameterization, enabling a correspondence between the mesh and the texture. However, simply using an unconstrained parameterization may produce undesired results (see Figure 1(c)). In practice, constrained texture mapping, which constrains some of the mesh vertices to correspond to specific points in the texture, is more preferred [1]–[6]. It ensures alignment of features in the 3D mesh and the texture for producing a better mapping result (see Figure 1(d)). The precise alignment of the vertices in the 3D mesh with the points in the texture domain is known as *hard constraints*. As pointed out in [3], [4], hard constraints are important for producing visually pleasing results. In particular, they are critical in hiding texture discontinuities along the seams in the parameterization. Hard constraints are also used in photogrammetric texture mapping [7].

Constrained texture mapping is however challenging technically due to basic requirements for a visually pleasing mapping. First, the mapping between the surface and the texture domain must prevent occurrence of foldover. Exact positional constraints for a given mesh connectivity may result in non-existence of a foldover-free mapping. Fortunately it is found that a foldover-free mapping always exists if a number of extra vertices called *Steiner vertices* are allowed to add. However, determining necessary Steiner vertices is very difficult. Second, for an arbitrary surface, isometric mapping in general does not exist. Efforts have to be taken to make the distortion as small as possible. Various metrics have been proposed to measure the distortion. Moreover, the mapping is often required to be smooth.

The problem of our constrained texture mapping can be stated: given a 3D triangular mesh $\mathscr{U}$ and a set of constrained point pairs $\{(U_i, P_i) \mid i = 1, 2, \cdots, h\}$, where U_i are the vertices of mesh $\mathscr{U}$ and $P_i \in \mathbb{R}^2$ are points in the image domain, we want to find a foldover-free piecewise linear mapping $F : \mathscr{U} \to \mathbb{R}^2$ satisfying $F(U_i) = P_i$. In this paper we restrict our discussion to the case of a mesh topologically equivalent to a rectangular area. For a mesh model of arbitrary topology, extra processes such as cutting and maintaining of continuity may be needed. Considering that many robust, efficient and mature parameterization algorithms have been developed, we propose a two-phase approach similar to the one of [4], which begins with an unconstrained planar parameterization f of the mesh and then performs a constrained mesh transformation G^* that moves the constrained vertices to the required positions. The constrained texture mapping is achieved by the composition: $F = G^* \circ f$, as illustrated in Figure 2. Particularly, in the first phase we use ABF++ [8], a boundary-free conformal parameterization, to embed the 3D mesh into a 2D domain. This paper thus mainly focuses on the second phase: finding a foldover-free transformation G^* that maps $f(U_i)$ to P_i.

Inspired by the work of [1], [9], we propose to construct the constrained 2D mesh transformation by an iterative warping process. To realize each warping step, we first construct locally bijective smooth mapping using radial basis function (RBF)-based interpolation to achieve alignment of constrained vertices, and then induce the

- *Y. Ma is with School of Computer Engineering, Nanyang Technological University, Singapore, 639798.*
- *J. Zheng (corresponding author) is with School of Computer Engineering, Nanyang Technological University, Singapore, 639798. E-mail: asjmzheng@ntu.edu.sg*
- *J. Xie is with Department of Mathematics, Hangzhou Normal University, Hangzhou 310036, China.*



(a) 3D mesh with feature vertices marked in red (b) Texture with corresponding points marked in green (c) Mapping result with unconstrained parameterization (d) Mapping result with constrained parameterization

Fig. 1. Texture mapping and constrained texture mapping.

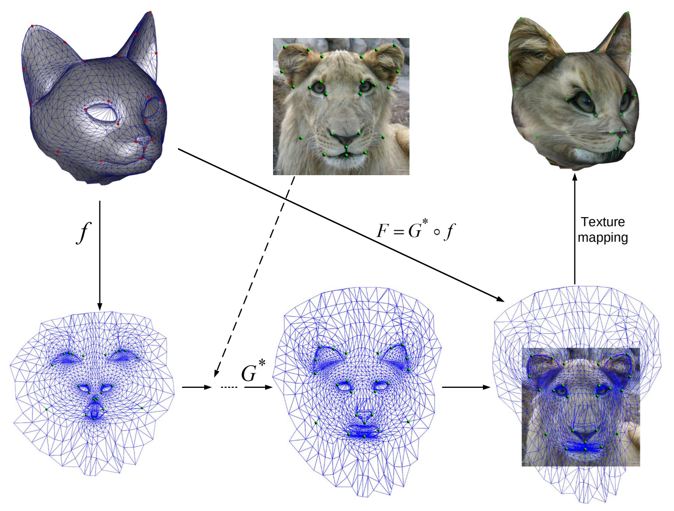


Fig. 2. The proposed method

piecewise affine transformation using the RBF function with barycentric coordinates within each triangle. While some previous work [6], [9] implicitly assumes that the foldover-free property of the induced transformation can be inherited from the local bijectivity of the smooth mapping, this is unfortunately not true. We improve the work of [9] by revealing the relation between the smooth mapping and its induced transformation and propose a refinement strategy based on the longest edge bisection to guarantee that the induced transformation is foldover free if the smooth mapping is locally bijective. To ensure the local bijectivity of the RBF-based warping, we propose to find non-intersecting trajectories to guide the warping and derive a bound for the warping stepsize. Furthermore, to make the warping process more effective, we derive a new stepsize from the necessary and sufficient condition for the induced transformation to be foldover free. Based on the new stepsize and the bound for the locally bijective smooth mapping, we determine whether we perform local mesh refinement or warping. Integrating all these technical components provides a constrained mesh warping algorithm that is effective, *provably* foldover free, able to handle a large number of constraints, and able to output a visually pleasing result without extra smoothing optimization. These merits are not simultaneously achieved by existing algorithms.

The main contributions of the paper include:

- We comprehensively analyze the relation between a C^2 continuous 2D mapping and its induced piecewise linear transformation, and propose a strategy to refine the mesh such that if the C^2 mapping is locally bijective, its induced transformation on the refined mesh keeps the orientation of each triangle of the refined mesh.
- An iterative RBF-based warping scheme is proposed and an explicit stepsize of the warping is derived, which can assure the corresponding RBF-based mapping to be locally bijective.
- An effective, provably foldover-free algorithm for smooth mesh warping with hard constraints is presented, which consists of preprocess, construction of non-intersecting warping trajectories, iterative RBF-based mapping and foldover-free induced transformation.

It is worth pointing out that though this paper focuses on texture mapping, foldover-free and constrained mesh warping itself is a difficult and interesting problem. It has many other applications such as in finite element simulation and shape interpolation [10]. Thus the solution proposed in this paper can find applications in other areas as well.

The rest of the paper is organized as follows. Section 2 reviews some prior work. Section 3 elaborates on the connection between a 2D mapping and its induced transformation and then proposes a refinement algorithm based on the longest edge bisection. Section 4 presents in detail the algorithm for 2D mesh transformation with positional constraints. Section 5 provides experimental results to demonstrate the proposed algorithms. Section 6 discusses some limitations of the proposed algorithm and Section 7 concludes the paper.

2 Prior Work

Texture mapping is often implemented using mesh parameterization. Extensive research has been done for embedding a 3D surface onto a 2D parametric domain during last two decades [8], [11]–[15]. Most existing work aims to construct a bijective mapping between 3D geometry and 2D domain, which minimizes the distortion occurring during the mapping in terms of some metrics such as angle and stretch. A good review can be found in [16], [17]. While some methods parameterize the surface with a specifed boundary [11]–[13], [18], the others free this requirement and treat the boundary as part of the minimization problem [8], [14], [15]. In general, boundary-free approaches such as ABF++ produce lower levels of distortion. Recently Lipman [19] introduced a method for constructing triangular mesh mappings into the plane with a bounded amount of conformal distortion and bijectivity. However, in extreme cases the method may fail to find a bijective solution even if one exists. Schneider et al. [20] proposed composite mean value mappings for designing bijective mappings between polygons and the bijectivity can be assured in the convex setting. Weber and Zorin [21] presented an algorithm for parameterizing a triangular with arbitrary fixed boundaries. The method guarantees a locally injective mapping as long as the boundary is a self-overlapping polygon.

To enforce positional matching on the vertices of the planar embedding, constraints are introduced into the minimization process for parameterization. Lévy [2] introduced a quadratic term into the formulation of parameterization. Desbrun et al. [22] and Gingold et al. [23] applied the classical Lagrange multiplier method to add the constraints to the objective function. Hence the solutions of these approaches satisfy the constraints approximately or "softly". These approaches may fail in the presence of large sets of constraints [4].

However, hard constraints are required in many applications. There are not many works published so far to provide a solution to the problem of texture mapping with hard constraints. Eckstein et al. [3] presented such a solution that starts with an unconstrained embedding and achieves hard constraints through constrained simplification and multi-resolution reconstruction. The method may add a number of Steiner vertices and is foldover free. As pointed out in [4], nevertheless, the method is very complicated and how it handles complicated constraints is not obvious. Kraevoy et al. [4] proposed an effective Matchmaker algorithm for adding positional constraints to texture maps. As observed in [1], [24], the Matchmaker algorithm does not consider consistent neighbor ordering and may fail to generate a foldover-free mapping in handling challenging constraints. Lee et al. [1] presented a foldover-free mesh warping scheme for hard constrained texture mapping. The feature alignment is achieved in a coarse mesh. Due to the large distortion after the alignment, a postembedding smoothing procedure is performed, which is generally complicated and time-consuming.

Yu et al. [9] proposed to avoid the expensive postprocessing by using RBF-based warping due to the smoothness of radial basis functions. The RBF-based warping iteratively reparameterizes the 2D embedding. However, as mentioned by the authors [9], the method may fail if too many constrained points crowd together and whether the algorithm converges is not clear. The RBF-based interpolation was also used to solve constrained problems in [5] and [25], but both of them cannot avoid foldover. Another smooth foldover-free warping scheme for constrained texture mapping was proposed in [6], which allows to constrain both position and orientation. The smooth warping is computed by constructing C^1 continuous vector fields. Tiddeman et al. [26] proposed to construct one-to-one image warping functions by interpolation and scaling. While the methods in [6], [9], [26] are devoted to constructing locally bijective smooth 2D mappings, unfortunately these smooth mappings cannot assure that the transformed 2D embedding is foldover free. Fujimura and Makarov [27] proposed a method based on a time-varying triangulation that provides a foldover-free mapping, but the method needs the process of edge swap, which changes the connectivity of the triangular mesh. It is thus not suitable for our application. In contrast, our proposed method uses the RBF-based warping to achieve the smoothness and guarantee the foldover-free property of 2D embedding.

3 Fundamental Theory

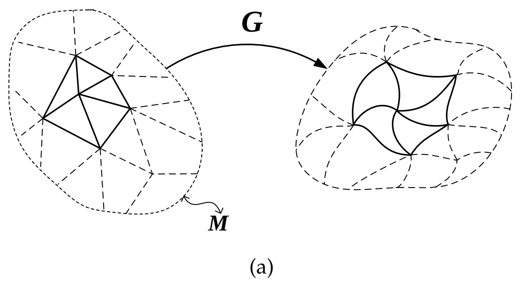
Let $M = \{\mathscr{V}, \mathscr{K}\}$ be a 2D triangular mesh, where $\mathscr{V} = \{V_i \mid V_i \in \mathbb{R}^2, i = 1, 2, \cdots\}$ is the list of vertices V_i, $\mathscr{K} = \{\triangle V_i V_j V_k \mid V_i, V_j, V_k \in \mathscr{V}, i \neq j, j \neq k, k \neq i\}$ is the list of triangles encoding how the vertices are connected, and $\Omega \subset \mathbb{R}^2$ is a bounded domain containing all the vertices. We assume that mesh M is conforming. That is, for any two intersecting triangles in $\mathscr{K}$, their intersection is either a vertex or a shared edge. A conforming mesh $\widetilde{M} = \{\widetilde{\mathscr{V}}, \widetilde{\mathscr{K}}\}$ is called a refinement of M if $\mathscr{V} \subset \widetilde{\mathscr{V}}$ and for any $\widetilde{\triangle} \in \widetilde{\mathscr{K}}$ there exists a unique $\triangle \in \mathscr{K}$ such that $\widetilde{\triangle}$ is within $\triangle$.

Given a C^2 continuous mapping $G : \Omega \to \mathbb{R}^2$, it maps M to a triangular mesh with curved edges (see Figure 3(a)). When the Jacobian of G does not vanish, G is *locally* one-to-one or bijective.

From G, we can induce a piecewise linear mapping $G^*_M : M \to \mathbb{R}^2$ by which the vertices V_i of M are mapped to $G(V_i)$ and any other point V lying in triangle $\triangle V_i V_j V_k$ is mapped to $G^*_M(V)$, a linear combination of $G(V_i), G(V_j)$ and $G(V_k)$:

$$G^*_M(V) = \lambda_i G(V_i) + \lambda_j G(V_j) + \lambda_k G(V_k) \qquad (1)$$

with barycentric coordinates $(\lambda_i, \lambda_j, \lambda_k)$ of V with respect to $\triangle V_i V_j V_k$. G^*_M is a triangular mesh transformation induced by G since it maps the edges of M to line segments (see Figure 3(b)). G^*_M is called *foldover free* if



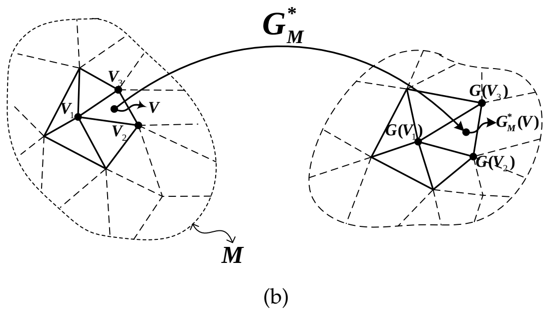


Fig. 3. (a) A 2D mapping G and (b) its induced transformation G_M^*

each triangle $\triangle G(V_i)G(V_j)G(V_k)$ keeps the orientation of triangle $\triangle V_iV_jV_k$.

3.1 Basic observations

While G and G_M^* are closely related, G_M^* relies only on triangle set $\mathscr{K}$ and the set $\{G(V_i) \mid V_i \in \mathscr{V}\}$. Some previous work implicitly assumes that the local bijectivity of G guarantees G_M^* to be foldover free. For example, [9] used the positive Jacobian of G as the condition for G_M^* to be foldover free to estimate the step bound. [6] used the local bijectivity of the vector field based warping for triangular mesh transformation. However, this is unfortunately not true. In fact, we have the following observations, which show the complexity of the relation between G and G_M^*.

Observation 1: *G is locally bijective, but G_M^* may fold over.* In Figure 4, G maps vertex A_i to vertex B_i and the red curves represent the image of edges of the original triangle. It can be seen that G is locally bijective, but when the transformed vertices are connected to form the induced transformation, foldover occurs.

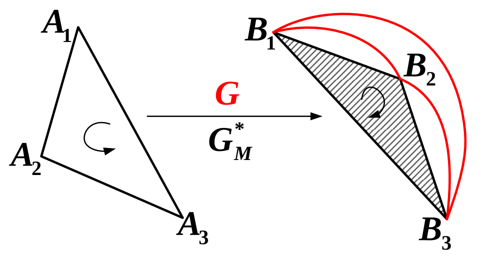


Fig. 4. G is locally bijective but G_M^* folds over.

Observation 2: *G is not locally bijective, but G_M^* could be foldover-free.* This is because the transformed vertices are connected using straight line segments rather than curves, as illustrated in Figure 5

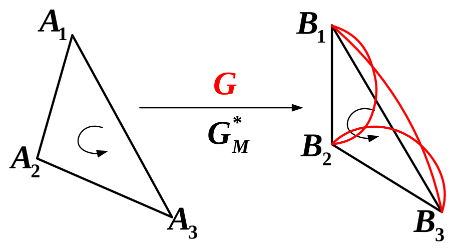


Fig. 5. G is not locally bijective, but G_M^* is foldover free.

Observation 3: *When G is bijective but G_M^* is folding over, if we refine M to $\widetilde{M}$, $G_{\widetilde{M}}^*$ might be foldover free.* For triangle $\triangle A_1A_2A_3$ in Figure 4, we refine it by inserting a point C to edge A_1A_3 (see Figure 6). While G is the same over $\triangle A_1A_2A_3$ and its refinement $\widetilde{\triangle A_1A_2A_3}$, in general $G_{\widetilde{\triangle A_1A_2A_3}}^* \neq G_{\triangle A_1A_2A_3}^*$. It can be easily checked that $G_{\widetilde{\triangle A_1A_2A_3}}^*$ in Figure 6 is foldover free.

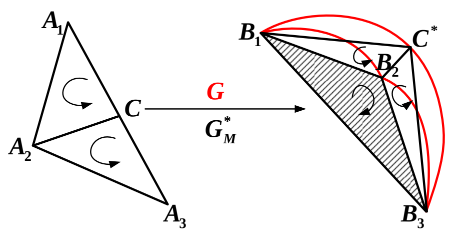


Fig. 6. The insertion of a Steiner point makes the mesh transformation foldover free.

Observation 4: *Foldover may propagate through subdivision.* When we split one triangle, an adjacent triangle should be subdivided simultaneously to conform the mesh. The new generated triangles may fold over even if their original parent triangle is foldover free. Figure 7 shows such a situation. Although triangle $\triangle B_1B_3B_4$ has the same orientation as triangle $\triangle A_1A_3A_4$, after splitting the orientation of $\triangle B_1C^*B_4$ is different from that of $\triangle A_1CA_4$ where $C^* = G(C)$.

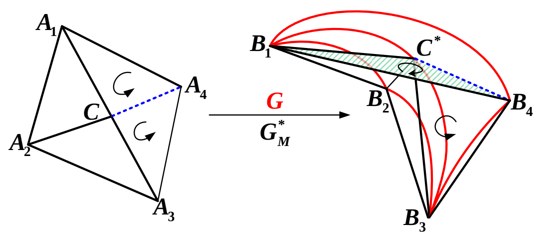


Fig. 7. The Propagation of foldover through one splitting.

3.2 Longest edge bisection

Observation 3 in Section 3.1 motivates us to raise a question: *Given a C^2 mapping $G : \Omega \subset \mathbb{R}^2 \to \mathbb{R}^2$ that* *is locally bijective and a triangular mesh M contained in Ω, does there always exist a conforming refinement $\widetilde{M}$ of M such that $G^*_{\widetilde{M}}$ is foldover free?* The answer is not trivial due to the propagation of foldover, as shown in Observation 4. In the following we present a constructive procedure to give a positive answer.

Our refinement is based on the *longest edge bisection* (LEB). See Figure 8 for an illustration. Suppose t_0 is a triangle that folds over. The refinement proceeds in the following steps:

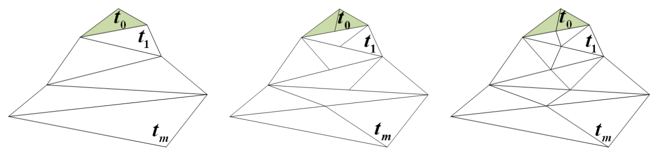
Step 1 Initialization: Let $i = 0$.

Step 2 Propagation: Repeat the following processes.

- For triangle t_i, find the longest edge e_i and flag it.
- Find the neighboring triangle t_{i+1} that shares e_i with t_i.
- If the neighboring triangle t_{i+1} is t_{i-1} or empty, break.
- Else let $i \leftarrow i + 1$.

In this way, we obtain an ordered sequence of triangles $\{t_0, t_1, \cdots, t_m\}$ such that triangle t_{j+1} and triangle t_j share the longest edge of t_j and t_m's adjacent triangle containing the longest edge of t_m is either t_{m-1} or empty. We call such a sequence the *longest edge propagation sequence* (LEPS) and denote it by LEPS(t_0).

Step 3 Bisection and conforming: for each triangle t_i in LEPS(t_0), connect the middle point of the longest edge e_i to its opposite vertex (see Figure 8(b) for bisection) and meanwhile for each pair of adjacent triangles t_{i-1} and t_i in LEPS(t_0), connect the middle point of the longest edge of t_{i-1} with that of t_i (see Figure 8(c) for conforming). The midpoints are the Steiner points and are added into the vertex list. The triangles are split by the new added edges and thus the triangular mesh is refined.



(a) Propagation from t_0 (b) Bisection (c) Conforming

Fig. 8. The LEB process that starts from triangle t_0.

The algorithm for constructing a refinement $\widetilde{M}$ of M such that the induced transformation on $\widetilde{M}$ is foldover-free is now straightforward. Referring to Algorithm 1, we start with $M^0 = M$ and $n = 0$, and apply the LEB-based refinement to each folding-over triangle in the folding-over triangle list τ^n, which generates a refined triangular mesh M^{n+1}. We repeat this process until there is no folding-over triangle.

It is worth pointing out that there exist other strategies for refining triangles. We choose the LEB-based refinement because it has several nice properties and

Algorithm 1 LEB-based Refinement

```
Initialization:
 1: $n \leftarrow 0$, $M^n \leftarrow M$.
 2: Find the folding-over triangle list $\tau^n$ in $M^n$.
Iterative Process:
 3: while $\tau^n$.size() $\neq 0$ do
 4:     while $\tau^n$.size() $\neq 0$ do
 5:         $t_0 = \tau^n$.front();
 6:         Apply LEB to $t_0$
 7:         Update $\tau^n \leftarrow \tau^n - \tau^n \bigcap$ LEPS($t_0$).
 8:     end while
 9:     $M^{n+1} \leftarrow M^n$, $n \leftarrow n + 1$.
10:     Find folding-over triangle list $\tau^n$ in $M^n$.
11: end while
Output: $M^n$.
```

the algorithm is proven to stop after a finite number of iterations of refinement.

Property 1. The propagation in the LEB process is guaranteed to stop in a finite number of bisections.

In fact, for any triangle t that folds over, its LEPS will not form a loop and thus consists of a finite number of triangles. That is, its elements t_i and t_j are different for any $i \neq j$. This is because the length of the longest edge of t_i is increasing with i.

Property 2. The angles of the refined triangles are greater than a positive constant.

Let M^k be the refined mesh of M obtained in Algorithm 1. Denote the smallest angles of triangles in M and of triangles in M^k by θ_0 and θ_0^k, respectively. It can be proven that $\theta_0^k \geq \frac{\theta_0}{2}$ (see [28]).

Property 3. The area of a sub-triangle is not greater than half of the area of its parent triangle.

This is obvious since the parent triangle is at least bisectioned.

What remains now is to prove that Algorithm 1 will terminate after a finite number of iterations. With Properties 1 and 2, we can easily obtain

Lemma 3.1: If Algorithm 1 proceeds in an infinite loop, there exists a triangle sequence $\{\triangle A_n B_n C_n \in \tau^n\}_{n=0}^{\infty}$ such that

$$\lim_{n\to\infty} \max\{\|A_n B_n\|, \|B_n C_n\|, \|A_n C_n\|\} = 0. \quad (2)$$

Lemma 3.2: Let $G = (G_1, G_2)^T : \Omega \subset \mathbb{R}^2 \to \mathbb{R}^2$ be a C^2 continuous mapping where G_i are scalar functions. If the Jacobian of G satisfies $|\nabla G| > 0$ in Ω where $\nabla G = \begin{pmatrix} \frac{\partial G_1}{\partial x} & \frac{\partial G_1}{\partial y} \\ \frac{\partial G_2}{\partial x} & \frac{\partial G_2}{\partial y} \end{pmatrix}$, then for any $\epsilon > 0$, there exists an $\eta > 0$ such that for any triangle $\triangle ABC \subset \Omega$ satisfying

$$\begin{cases} \max(\sin\angle A, \sin\angle B, \sin\angle C) & > & \epsilon \\ \max(\|AB\|, \|BC\|, \|AC\|) & < & \eta \end{cases} \quad (3)$$

the transformed triangle $\triangle G(A)G(B)G(C)$ keeps the orientation of triangle $\triangle ABC$.

Proof: For any $\epsilon > 0$, we consider triangle $\triangle ABC$ that satisfies $\max(\sin\angle A, \sin\angle B, \sin\angle C) > \epsilon$. Without loss of generality, we assume that vertices A, B and C are ordered counter-clockwise and $\sin\angle A = \max(\sin\angle A, \sin\angle B, \sin\angle C)$. We introduce operator $\otimes$ for 2D vectors $V = (a, b)$ and $W = (c, d)$ such that $V \otimes W = ad - bc$. Then

$$(B-A)\otimes(C-A) = \|AB\|\|AC\|\sin\angle A > \|AB\|\|AC\|\epsilon > 0. \tag{4}$$

For any $V \in \triangle ABC$, we have

$$G_i(V) = G_i(A) + \left(\frac{\partial G_i(A)}{\partial x}, \frac{\partial G_i(A)}{\partial y}\right)(V - A) + O_{i,A,V} \tag{5}$$

where

$$O_{i,A,V} = (V-A)^T \left[\begin{array}{cc} \frac{\partial^2 G_i(V_i^*)}{\partial x^2} & \frac{\partial^2 G_i(V_i^*)}{\partial x \partial y} \\ \frac{\partial^2 G_i(V_i^*)}{\partial x \partial y} & \frac{\partial^2 G_i(V_i^*)}{\partial y^2} \end{array}\right](V - A)$$

with a point V_i^* on the line connecting V and A. Let $\mathbf{O}_{A,V} = (O_{1,A,V}, O_{2,A,V})^T$. Since Ω is bounded, G is C^2 continuous and $|\nabla G| > 0$, there exist positive numbers μ, ν and L such that $\|\mathbf{O}_{A,V}\| \le \mu\|V - A\|^2$ and $L < \|\nabla G\|_F < \nu$ for $V \in \Omega$ where $\|\nabla G\|_F$ represents the Frobenius norm of the matrix G.

Substituting $V = B$ and $V = C$ into Eq.(4) gives

$$\begin{aligned} G(B) &= G(A) + \nabla G(B-A) + \mathbf{O}_{A,B}, \\ G(C) &= G(A) + \nabla G(C-A) + \mathbf{O}_{A,C}. \end{aligned}$$

After calculation, we have

$$(G(B)-G(A))\otimes(G(C)-G(A)) = |\nabla G|(B-A)\otimes(C-A)+\psi$$

with $\psi = \nabla G(B-A) \otimes \mathbf{O}_{A,C} - \nabla G(C-A) \otimes \mathbf{O}_{A,B} + \mathbf{O}_{A,B} \otimes \mathbf{O}_{A,C}$. We choose $\eta = \min\left(1, \frac{L\epsilon}{2\mu\nu + \mu^2}\right)$. If $\max(\|AB\|, \|BC\|, \|AC\|) < \eta$, then

$$\begin{aligned} \|\psi\| &\le \nu\mu\|B-A\|\|C-A\|^2 + \nu\mu\|C-A\|\|B-A\|^2 \\ &\quad + \mu^2\|A-B\|^2\|A-C\|^2 < \|AB\|\|AC\|(2\mu\nu + \mu^2)\eta \\ &\le \|AB\|\|AC\|L\epsilon \le |\nabla G|(B-A)\otimes(C-A). \end{aligned}$$

Therefore $(G(B) - G(A)) \otimes (G(C) - G(A)) > 0$. That is, the transformed triangle $\triangle G(A)G(B)G(C)$ keeps the orientation of triangle $\triangle ABC$. ■

The geometric meaning of Lemma 3.2 is that a C^2 continuous mapping G with positive Jacobian can ensure a triangle transformed by the induced mapping of G to maintain its orientation as long as the triangle is sufficiently small.

Theorem 3.3: If $G : \Omega \to \mathbb{R}^2$ is a C^2 continuous mapping with $|\nabla G| > 0$ in Ω, Algorithm 1 will terminate after a finite number of iterations.

Proof: If Algorithm 1 cannot terminate in a finite number of iterations, according to Lemma 3.1, we can find a folding-over triangle sequence $\{\Delta A_i B_i C_i\}_{i=1}^{\infty}$ that satisfies

$$\lim_{i\to\infty} \max(\|A_iB_i\|, \|B_iC_i\|, \|A_iC_i\|) = 0.$$

Property 2 ensures that

$$\max(\sin\angle A_n, \sin\angle B_n, \sin\angle C_n) \ge \sin\left(\frac{\theta_0}{2}\right)$$

where θ_0 is the smallest angle of the triangles in M. We let $\epsilon = \sin\left(\frac{\theta_0}{2}\right)$. By Lemma 3.2, there exists an η such that when the diameter of the triangle is less than η, its transformed counterpart keeps the orientation, which contradicts the fact that $\triangle A_n B_n C_n$ are folding-over triangles. ■

4 Algorithm for 2D Mesh Transformation with Positional Constraints

The observations and theory in preceding section suggest an approach to constructing a foldover-free 2D mesh transformation, via some C^2 continuous mapping together with the LEB-based refinement if necessary. This section presents such an algorithm.

Similar to some previous work such as [4], the input to our algorithm is composed of a 2D triangular mesh M, a set of constrained mesh vertices $V_c = \{V_{c,i} \mid i = 1, 2, \cdots, h\}$, and a set of matching points $P_c = \{P_{c,i} \mid i = 1, 2, \cdots, h\}$ corresponding to V_c. The output is a valid transformed mesh of M or its refinement, in which all $V_{c,i}$ are replaced by $P_{c,i}$. The algorithm contains four components listed below. More details of the components are described in the following subsections.

(1) Preprocess: We perform a similarity transformation to roughly align the constrained point pairs in order to reduce unnecessary warping in subsequent steps.

(2) Warping trajectories: For constrained vertex set V_c and their matching point set P_c, a set of immediate points $C^j = \{C_i^j\}, j = 0, 1, \cdots, m$ with $C^0 = V_c$ and $C^m = P_c$ is generated. Each C_i^j is connected to C_i^{j+1} by a straight line. All these line segments form piecewise linear trajectories, along which the constrained vertices $V_{c,i}$ move to $P_{c,i}$ during the warping (or iterative mapping). These trajectories do not intersect each other.

(3) RBF-based mapping: The mapping from C^j to C^{j+1} is achieved by iteratively applying the RBF-based interpolation technique. Rather than interpolating absolute positions, we interpolate the displacements of the constrained vertices. A safe stepsize assuring the mapping derived from the RBF-based interpolation to be locally bijective on the mesh is presented.

(4) Foldover-free transformation: Mesh transformation is induced from the RBF-based mapping. A stepsize for the mesh transformation to be foldover free is derived. If the stepsize is sufficient to make the transformation reach the target or the stepsize is greater than or equal to the safe stepsize of the RBF-based mapping, the transformation is constructed. Otherwise, to avoid using too small stepsize, we use the LEB-based refinement (i.e., Algorithm 1) to refine the mesh. Then we repeat steps (3) and (4).

4.1 Preprocess

A similarity transformation is composed of translation, rotation and uniform scaling, thus preserving angles, and can be represented by

$$u = ax - by + c, \quad v = bx + ay + d$$

where (x, y) and (u, v) represent the coordinates of points before and after the transformation, respectively, and a, b, c and d are the coefficients to be determined. Suppose that $V_{c,i}$ and $P_{c,i}$ have coordinates (x_i, y_i) and (u_i, v_i) for $i = 1, 2, \cdots, h$. We find the similarity transformation by minimizing the following objective function:

$$\sum_{i=1}^{h} \left((ax_i - by_i + c - u_i)^2 + (bx_i + ay_i + d - v_i)^2 \right),$$

which gives

$$\begin{aligned} a &= \frac{h(\sum_{i=1}^{h} u_i x_i + \sum_{i=1}^{h} v_i y_i) - \sum_{i=1}^{h} x_i \sum_{i=1}^{h} u_i - \sum_{i=1}^{h} y_i \sum_{i=1}^{h} v_i}{h \sum_{i=1}^{h} (x_i^2 + y_i^2) - (\sum_{i=1}^{h} x_i)^2 - (\sum_{i=1}^{h} y_i)^2} \\ b &= \frac{h(\sum_{i=1}^{h} v_i x_i - \sum_{i=1}^{h} u_i y_i) + \sum_{i=1}^{h} y_i \sum_{i=1}^{h} u_i - \sum_{i=1}^{h} x_i \sum_{i=1}^{h} v_i}{h \sum_{i=1}^{h} (x_i^2 + y_i^2) - (\sum_{i=1}^{h} x_i)^2 - (\sum_{i=1}^{h} y_i)^2} \\ c &= \frac{\sum_{i=1}^{h} u_i - (\sum_{i=1}^{h} x_i) a + (\sum_{i=1}^{h} y_i) b}{h} \\ d &= \frac{\sum_{i=1}^{h} v_i - (\sum_{i=1}^{h} y_i) a - (\sum_{i=1}^{h} x_i) b}{h}. \end{aligned}$$

The transformation is applied to mesh M. Without loss of generality, in the rest of the paper we assume that the mesh M has been transformed.

4.2 Construction of non-intersecting warping trajectories

Given constrained vertex set $V_c = \{V_{c,i}\}$ and their matching point set $P_c = \{P_{c,i}\}$, directly connecting each corresponding point pair often causes intersection. We aim to find non-intersecting trajectories for $V_{c,i}$ to move to $P_{c,i}$. Specifically, for each vertex $V_{c,i}$, a trajectory is a polyline $\mathbf{C}_i(s)$ with $m + 1$ nodes $C_i^j, j = 0, 1, \cdots, m$ where $C_i^0 = V_{c,i}$ and $C_i^m = P_{c,i}$. $\mathbf{C}_i(s)$ is parameterized by

$$\mathbf{C}_i(s) = (1 - ms + j)C_i^j + (ms - j)C_i^{j+1}, s \in [\frac{j}{m}, \frac{j+1}{m}].$$

Non-intersection of $\mathbf{C}_i(s)$ means that for any $i \neq k$ and any $s \in [0, 1]$, $\mathbf{C}_i(s) \neq \mathbf{C}_k(s)$.

Computing non-intersecting trajectories of points from the source to the target has been well studied in morphing [29], which requires the input to be two compatible triangulations. Two triangulations are called compatible if their face lattices are isomorphic. In general, determining whether two sets of points are compatible is NP-hard [30], [31].

Here we propose to compute compatible triangulations of two point sets using the warping scheme of [1]. First, we perform Delaunay triangulation on the

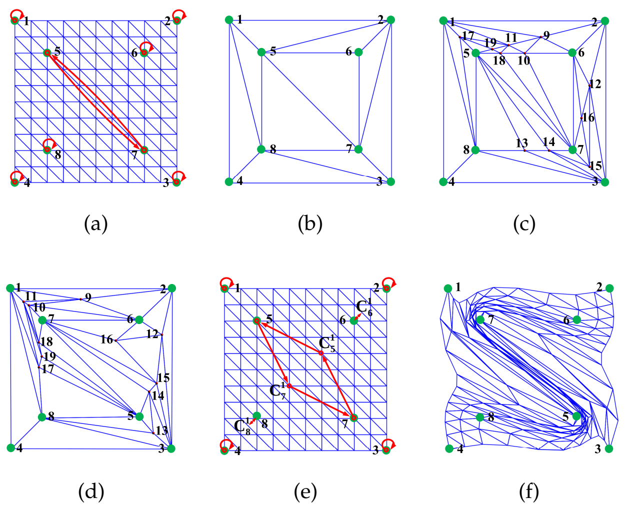


Fig. 9. Generation of non-intersecting warping trajectories. (a) The input mesh and constrained points. (b) The Delaunay triangulation on the constrained points. (c) and (d) The compatible triangulations of the source and target. (e) The warping trajectories. (f) Foldover-free mapping.

constrained vertex set V_c, yielding a triangular mesh S. Then we let $T = S$. T serves as the input mesh for the warping scheme of [1]. Iteratively, the warping scheme aligns T with the matching positions $P_{c,i}$. If we need to refine T during the iteration, we apply the same refinement to S, too. In this way, the warping scheme preserves the compatibility and the final T interpolates $P_{c,i}$ and has the same connectivity as S. Since the input in this approach consists of only the constrained vertices, the process of computing compatible triangulations is usually very fast. Figure 9 shows a simple example of generating compatible triangulations.

After compatible triangulations S and T are obtained, the morphing algorithm in [29] is used to generate the non-intersecting trajectories of the constrained vertices. Then $\mathbf{C}_i(s)$ can be obtained by appropriately sampling the trajectories.

4.3 Iterative RBF-based mapping

Once the construction of warping trajectories is complete, we obtain $\{C_i^0\} \to \{C_i^1\} \to \cdots \to \{C_i^m\}$. Now we consider the mapping over the span $\{C_i^j\} \to \{C_i^{j+1}\}$. The mapping over the other spans is similarly constructed.

The mapping is an interpolation function that maps C_i^j to C_i^{j+1} for all i. We use radial basis functions for this purpose. Radial basis functions are popular for scattered data interpolation. They do not require the data points to lie on regular grids. Notice that there may not exist a single locally bijective RBF-based interpolation over the span $\{C_i^j\} \to \{C_i^{j+1}\}$. Our idea is to accomplish the interpolation of displacements $C_i^{j+1} - C_i^j$ by several steps and each step interpolates a portion of displacements, $\delta^{j,l}(C_i^{j+1} - C_i^j)$, for some stepsizes $\delta^{j,l}$ and is assured to be locally bijective. As a result, the mapping over the span $\{C_i^j\} \to \{C_i^{j+1}\}$ is a composition of all the mappings defined in each step. The core techniques here include two aspects: (1) how to compute the interpolation mapping and (2) how to compute the stepsize to ensure local bijectivity, which are explained in the rest of this subsection.

Let $C_i^{j,l}$ be the point obtained from C_i^j after l steps, with $C_i^{j,0} = C_i^j$. Also denote by $M^{j,l}$ the 2D triangular mesh at the step where the constrained vertices have become $\{C_i^{j,l}\}$. Let $f_i^j = C_i^{j+1} - C_i^j$, $i = 1, 2, \cdots, h$, be the displacements of the span $\{C_i^j\} \to \{C_i^{j+1}\}$. We want to find a smooth interpolating function

$$D^{j,l}(V) = \sum_{i=1}^{h} \gamma_i \phi(|V - C_i^{j,l}|) + a_1 x + a_2 y + a_3 \tag{6}$$

satisfying

$$D^{j,l}(C_i^{j,l}) = \delta^{j,l} f_i^j \tag{7}$$

where $V = (x, y)^T$, $\delta^{j,l}$ is a stepsize controlling the displacement, $a_1x + a_2y + a_3$ accounts for linear transformation, and the radial basis function $\phi(r)$ is chosen to be the thin plate spline $r^2 log(r)$. The thin plate spline has been widely used in 2D scattered data interpolation, it has closed-form solutions for interpolation and usually presents a very smooth interpolation. All these properties make the thin plate spline suitable for our problem.

The constraints (7) give h equations. To determine the coefficients γ_i, the following orthogonality conditions are introduced:

$$\sum_{i=1}^{h} \gamma_i = \sum_{i=1}^{h} \gamma_i x_i = \sum_{i=1}^{h} \gamma_i y_i = 0. \tag{8}$$

Hence we have a linear system of size $(h+3) \times (h+3)$:

$$K^{j,l} \begin{pmatrix} \mathbf{\Gamma} \\ \mathbf{A} \end{pmatrix} = \begin{pmatrix} \delta^{j,l} \mathbf{f}^j \\ \mathbf{0} \end{pmatrix} \tag{9}$$

where

$$K^{j,l} = \begin{bmatrix} \Phi_{11} & \Phi_{12} & \cdots & \Phi_{1h} & x_1 & y_1 & 1 \\ \Phi_{21} & \Phi_{22} & \cdots & \Phi_{2h} & x_2 & y_2 & 1 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\ \Phi_{h1} & \Phi_{h2} & \cdots & \Phi_{hh} & x_h & y_h & 1 \\ x_1 & x_2 & \cdots & x_h & 0 & 0 & 0 \\ y_1 & y_2 & \cdots & y_h & 0 & 0 & 0 \\ 1 & 1 & \cdots & 1 & 0 & 0 & 0 \end{bmatrix} \tag{10}$$

with $\Phi_{i,k} = \phi(\|C_i^{j,l} - C_k^{j,l}\|)$, $C_i^{j,l} = (x_i, y_i)^{\mathrm{T}}$, $\mathbf{\Gamma} = (\gamma_1, \cdots, \gamma_h)^{\mathrm{T}}$, $\mathbf{A} = (a_1, a_2, a_3)^T$, and $\mathbf{f}^j = (f_1^j, \cdots, f_h^j)^{\mathrm{T}}$. It has been known that the linear system (10) is guaranteed to be invertible if the locations of the data points to be interpolated do not lie on a line and do not intersect. This can be easily satisfied if we move the constrained vertices along the warping trajectories generated in Section 4.2. Then the solution to the linear system is:

$$\begin{pmatrix} \mathbf{\Gamma} \\ \mathbf{A} \end{pmatrix} = \left(K^{j,l}\right)^{-1} \begin{pmatrix} \mathbf{f}^j \\ \mathbf{0} \end{pmatrix} \delta^{j,l}. \tag{11}$$

After obtaining the RBF's coefficients, we define a mapping $g^{j,l} : \mathbb{R}^2 \to \mathbb{R}^2$ for this step:

$$g^{j,l}(V) = V + D^{j,l}(V) = V + \mathbf{M}^{j,l}(V) \left(K^{j,l}\right)^{-1} \begin{pmatrix} \mathbf{f}^j \\ \mathbf{0} \end{pmatrix} \delta^{j,l} \tag{12}$$

where

$$\mathbf{M}^{j,l}(V) = \left(\phi(\|V - C_1^{j,l}\|^2), \cdots, \phi(\|V - C_h^{j,l}\|^2), x, y, 1\right). \tag{13}$$

Next we consider how to find an appropriate stepsize $\delta^{j,l}$ to make sure that the mapping $g^{j,l}$ is locally bijective. Let K^x and K^y represent two $(h+3)$-dimensional column vectors consisting of the x and y components of $\left(K^{j,l}\right)^{-1} \begin{pmatrix} \mathbf{f}^j \\ \mathbf{0} \end{pmatrix}$, respectively. The Jacobian of $g^{j,l}$ can be written:

$$\begin{aligned} |\nabla g^{j,l}| &= \begin{vmatrix} 1 + \frac{\partial \mathbf{M}^{j,l}(V)}{\partial x} K^x \delta^{j,l} & \frac{\partial \mathbf{M}^{j,l}(V)}{\partial y} K^x \delta^{j,l} \\ \frac{\partial \mathbf{M}^{j,l}(V)}{\partial x} K^y \delta^{j,l} & 1 + \frac{\partial \mathbf{M}^{j,l}(V)}{\partial y} K^y \delta^{j,l} \end{vmatrix} \\ &= \alpha (\delta^{j,l})^2 + \beta \delta^{j,l} + 1 \end{aligned}$$

where

$$\begin{aligned} \alpha &= \left(\frac{\partial \mathbf{M}^{j,l}(V)}{\partial x} K^x\right) \left(\frac{\partial \mathbf{M}^{j,l}(V)}{\partial y} K^y\right) \\ &\quad - \left(\frac{\partial \mathbf{M}^{j,l}(V)}{\partial y} K^x\right) \left(\frac{\partial \mathbf{M}^{j,l}(V)}{\partial x} K^y\right), \\ \beta &= \left(\frac{\partial \mathbf{M}^{j,l}(V)}{\partial x} K^x\right) + \left(\frac{\partial \mathbf{M}^{j,l}(V)}{\partial y} K^y\right). \end{aligned}$$

$|\nabla g^{j,l}|$ is a quadratic function in $\delta^{j,l}$. When $\delta^{j,l} = 0$, $|\nabla g^{j,l}| = 1$ is positive, which implies that if $\delta^{j,l}$ is sufficiently small, the Jacobian of $g^{j,l}$ is positive and the mapping $g^{j,l}$ is thus locally bijective. Now we show how to find an appropriate stepsize $\delta^{j,l}$ according to three situations:

- $\alpha = 0$: If $\beta \geq 0$, then for any $\delta^{j,l}$, $|\nabla g^{j,l}|$ is always positive. If $\beta < 0$, $|\nabla g^{j,l}|$ has one positive root $\delta = -1/\beta$. When $\delta^{j,l} < \delta$, $|\nabla g^{j,l}| > 0$.
- $\alpha < 0$: $|\nabla g^{j,l}|$ has one positive root $\delta = \frac{-\beta - \sqrt{\beta^2 - 4\alpha}}{2\alpha}$. When $0 < \delta^{j,l} < \delta$, $|\nabla g^{j,l}|$ is positive.
- $\alpha > 0$: If $\beta \geq 0$ or $\beta^2 - 4\alpha < 0$, then for any $\delta^{j,l}$, $|\nabla g^{j,l}|$ is positive. Otherwise, $|\nabla g^{j,l}|$ has two positive roots. The smaller one is $\delta = \frac{-\beta - \sqrt{\beta^2 - 4\alpha}}{2\alpha}$ and for any $\delta^{j,l} \in (0, \delta)$, $|\nabla g^{j,l}| > 0$.

If we set

$$\delta = \frac{2}{| - \beta + \sqrt{|\beta^2 - 4\alpha|}|}, \tag{14}$$

it can be easily verified that any $\delta^{j,l} \in (0, \delta)$ assures $|\nabla g^{j,l}| > 0$ in all above three situations. That is, the δ of Eq.(14) is a bound sufficient for $\delta^{j,l}$ to guarantee positivity of $|\nabla g^{j,l}|$. Note that $| - \beta + \sqrt{|\beta^2 - 4\alpha|}|$ is a continuous function in $V \in \Omega$. Hence it can achieve its maximum value $\max\limits_{V \in \Omega} | - \beta + \sqrt{|\beta^2 - 4\alpha|}|$. Therefore if we define

$$\delta_{RBF} = \min \left\{ \rho \frac{2}{\max\limits_{V \in \Omega} | - \beta + \sqrt{|\beta^2 - 4\alpha|}|}, 1 \right\} \tag{15}$$

with a positive $\rho < 1$ (in this paper we set $\rho = 0.9$), δ_{RBF} is a positive number such that for any $\delta^{j,l} \in (0, \delta_{RBF}]$, $|\nabla g^{j,l}| > 0$ holds. Thus we call δ_{RBF} a safe stepsize.

In the warping process, if we let $\delta^{j,l} = \delta_{RBF}$, when $\sum_{l}^{\kappa} \delta^{j,l} \geq 1$ for a positive integer κ, the interpolation process successfully reaches C^{j+1} and a locally bijective RBF-based mapping from C^j to C^{j+1} can be obtained by composing all $g^{j,l}$: $g^j = g^{j,1} \circ g^{j,2} \circ \cdots \circ g^{j,\kappa}$.

4.4 Foldover-free induced transformation

While we can construct a locally bijective RBF-mapping g^j as in Section 4.3 and then construct the induced transformation from it together with an LEB-based refinement described in Section 3.2, the stepsize δ_{RBF} may be too conservative, which makes the process require a large number of iterations. On the other hand, Observation 2 in Section 3.1 suggests that we determine the stepsize $\delta^{j,l}$ directly using the condition for the induced transformation $(g^{j,l})^*_{M^{j,l}}$ to be foldover free, with which the RBF-based mapping $g^{j,l}$ may even not be locally bijective.

Consider a triangle $\triangle A_1A_2A_3 \subset M^{j,l}$. It is mapped by the RBF-based mapping $g^{j,l}$ to $\triangle B_1B_2B_3$ with $B_i = g^{j,l}(A_i) = A_i + \mathbf{M}^{j,l}(A_i)\left(K^{j,l}\right)^{-1}\begin{pmatrix}\mathbf{f}^j\\ \mathbf{0}\end{pmatrix}\delta^{j,l}$. The necessary and sufficient condition for $\triangle B_1B_2B_3$ and $\triangle A_1A_2A_3$ to have the same orientation is

$$((A_2 - A_1) \otimes (A_3 - A_1)) \cdot ((B_2 - B_1) \otimes (B_3 - B_1)) > 0$$

which gives

$$F(\delta^{j,l}) \triangleq a(\delta^{j,l})^2 + b\delta^{j,l} + 1 > 0$$

where

$$\begin{aligned}
n_1 &= \tfrac{1}{(A_2 - A_1)\otimes(A_3 - A_1)}\\
a &= n_1\left(\left(\mathbf{M}^{j,l}(A_2) - \mathbf{M}^{j,l}(A_1)\right)\left(K^{j,l}\right)^{-1}\begin{pmatrix}\mathbf{f}^j\\ \mathbf{0}\end{pmatrix}\right)\\
&\quad \otimes\left(\left(\mathbf{M}^{j,l}(A_3) - \mathbf{M}^{j,l}(A_1)\right)\left(K^{j,l}\right)^{-1}\begin{pmatrix}\mathbf{f}^j\\ \mathbf{0}\end{pmatrix}\right)\\
b &= n_1(A_2 - A_1) \otimes \left(\mathbf{M}^{j,l}(A_3) - \mathbf{M}^{j,l}(A_1)\right)\left(K^{j,l}\right)^{-1}\begin{pmatrix}\mathbf{f}^j\\ \mathbf{0}\end{pmatrix}\\
&\quad - n_1(A_3 - A_1) \otimes \left(\mathbf{M}^{j,l}(A_2) - \mathbf{M}^{j,l}(A_1)\right)\left(K^{j,l}\right)^{-1}\begin{pmatrix}\mathbf{f}^j\\ \mathbf{0}\end{pmatrix}
\end{aligned}$$

Now we look for a $\delta \in (0, 1]$ such that for any $\delta^{j,l} \leq \delta$, $F(\delta^{j,l}) > 0$. Following the approach in Section 4.3, we can find the following δ that satisfies the requirement:

$$\delta = \begin{cases} \min\left\{\rho\frac{2}{-b+\sqrt{b^2-4a}}, 1\right\}, & \text{if } a < 0 \text{ or } (b < 0, b^2 \geq 4a)\\ 1, & \text{otherwise}\end{cases} \tag{16}$$

with a positive a positive $\rho < 1$. We find such a δ for each triangle in mesh $M^{j,l}$ to assure no foldover and then choose the smallest one.

Once we obtain the stepsize δ, the algorithm proceeds in three different ways:

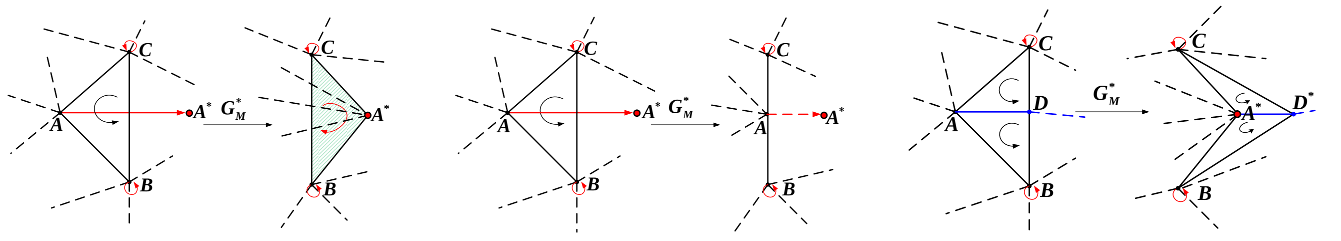
- If δ is greater than or equal to $1 - \sum_{k=1}^{l-1} \delta^{j,k}$ which is sufficient for the process to reach the target C^{j+1}, we simply let $\delta^{j,l} = 1 - \sum_{k=1}^{l-1} \delta^{j,k}$, construct the RBF-based mapping $g^{j,l}$ accordingly and then the induced transformation $(g^{j,l})^*_{M^{j,l}}$.
- Else if $\delta \geq \delta_{RBF}$, we let $\delta^{j,l} = \delta$, and construct the RBF-based mapping $g^{j,l}$ and then the induced transformation $(g^{j,l})^*_{M^{j,l}}$.
- Otherwise, $\delta < \delta_{RBF}$. If we use this δ for computing the RBF-based mapping without adding Steiner points, in the subsequent warping δ may become smaller and smaller and the process cannot reach the target. Figure 10 shows such an example, where A cannot cross the line BC without refinement. To overcome this problem, we let $\delta^{j,l} = \delta_{RBF}$ and construct a locally bijective RBF-based mapping $g^{j,l}$. Then Algorithm 1 (LEB-based refinement) is applied to refine $M^{j,l}$ to construct a foldover free induced transformation.

Algorithm 2 Constrained 2D Mesh Transformation

```
Initialization:
 1: Preprocess.
 2: Compute compatible triangulation of the user specified constrained vertices $V_c$ and the matching points $P_c$.
 3: Construct non-intersecting warping trajectories $\{\mathbf{C}_i(s)\}$.
 4: $j \leftarrow 0$
Iterative Process:
 5: while $j \neq m$ do
 6:     Set $remaining\_ratio = 1$
 7:     while $remaining\_ratio > 0$ do
 8:         Compute $\delta_{RBF}$ at $M^{j,l}$ using (15).
 9:         Compute the stepsize $\delta$ using (16).
10:         if $\delta \geq remaining\_ratio$ then
11:             Set $\delta = remaining\_ratio$.
12:             Set $remaining\_ratio = 0$.
13:             Compute induced mesh transformation.
14:             Break.
15:         end if
16:         if $\delta \geq \delta_{RBF}$ then
17:             Compute induced mesh transformation.
18:         else
19:             Set $\delta = \delta_{RBF}$
20:             Apply Algorithm 1 to the mesh.
21:         end if
22:         $remaining\_ratio \leftarrow remaining\_ratio - \delta$.
23:     end while
24:     $j \leftarrow j + 1$.
25: end while
Output: $M$.
```

Summarizing all the above steps, we arrive at a complete algorithm which is outlined in Algorithm 2. Moreover, Algorithm 2 can be proven to always work.

Theorem 4.1: Algorithm 2 terminates in a finite number of steps and outputs a foldover-free 2D mesh transfor-



(a) Foldover happens for a constrained mapping.

(b) The foldover-free condition prevents vertex A from moving across edge BC in the warping without refinement.

(c) Foldover is resolved after adding a Steiner vertex.

Fig. 10. An example of constrained mapping that maps A, B and C to A^*, B and C, respectively.

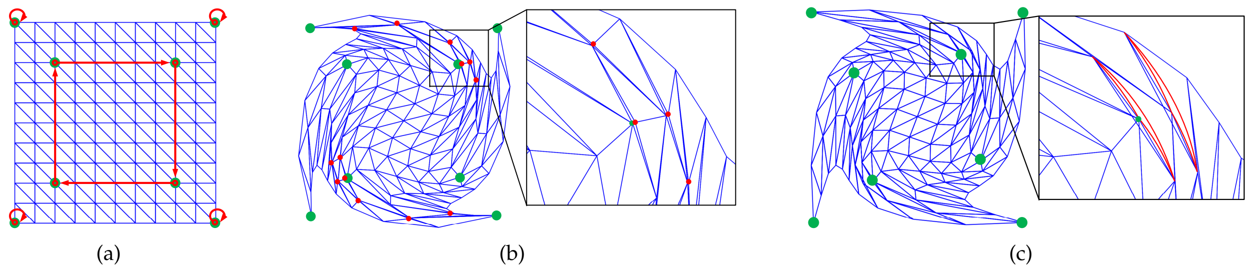


Fig. 11. (a) An input mesh with 8 constraints. The 4 corner vertices are constrained to remain unchanged and each of the 4 inner vertices moves to its next as indicated by the red arrows. (b) A foldover free mesh is obtained by our algorithm, which adds 13 Steiner points. (c) Using the algorithm of [9], we can obtain a locally bijective mapping G, which is depicted by the red curves in the zoomed window, bur foldover occurs when the mapped edges are straightened.

mation that satisfies given positional constraints.

Proof: We only need to prove that the proposed warping process can reach the target in a finite number of iterations. In fact, it can be seen that each stepsize $\delta^{j,l}$ in the warping process except for the last step reaching the target is always greater than or equal to δ_{RBF}. If we further define

$$\delta^*_{RBF} = \min\left\{\rho\frac{2}{\max\limits_{V\in\Omega,s\in[0,1]}|-\beta+\sqrt{|\beta^2-4\alpha|}|}, 1\right\},$$

then $\delta_{RBF} \geq \delta^*_{RBF}$ for all δ_{RBF}, and δ^*_{RBF} is a positive constant since $|-\beta+\sqrt{|\beta^2-4\alpha|}|$ is a continuous function in $V \in \Omega$ and $s \in [0,1]$. Thus the warping from C^j to C^{j+1} needs at most $\lceil\frac{1}{\delta^*_{RBF}}\rceil$ steps. This completes the proof. ■

Figure 11 is an example of constrained 2D transformations. The input is given in Figure 11(a), which includes a 2D triangular mesh and eight constraints highlighted in green. The four corner vertices are mapped to themselves and the four inner vertices are mapped to their respective next counterparts, as indicated by red arrows. Using our algorithm, a foldover-free transformation is automatically constructed with 13 Steiner points being added and the transformed mesh is shown in Figure 11(b). As a comparison, we run the algorithm of [9] on this input. The constructed constrained mapping G is locally bijective, but the induced transformation is not foldover free, which is depicted in Figure 11(c).

5 Experimental Results

This section provides several examples to demonstrate the proposed algorithm. These examples have varying complexity. The number of faces in these triangular mesh models ranges from 2K to 21K and the number of constrained vertices ranges from 21 to 83. Note that among only a few algorithms that can guarantee hard constraints, [1] is a relatively recent one, which is able to handle complicated constraints while adding only a small number of Steiner points, and output visually pleasing results. Hence we also provide the experimental results obtained by [1] for comparison.

Figures 12-14 show the visual results. In the figures, the first column gives the index of the models. The second column shows the input texture on the top and the 3D mesh model at the bottom. The constrained points are highlighted in green. The third column and fourth column display the models textured by a checker image by [1] and our algorithm, respectively. The use of the checker image well depicts the difference of the results created by our method and [1]. It can be seen that our algorithm usually produces a smoother mapping than [1] though a non-linear post-optimization has been performed in [1]. For example, the zoomed view of the cow model (i.e., model (g)) clearly shows the difference. In fact, [1] produces visually apparent distortions in the areas around the constrained points. The last column shows the alignment of the textures with the constrained embedding of the meshes at the top and the constrained textured 3D meshes at the bottom.

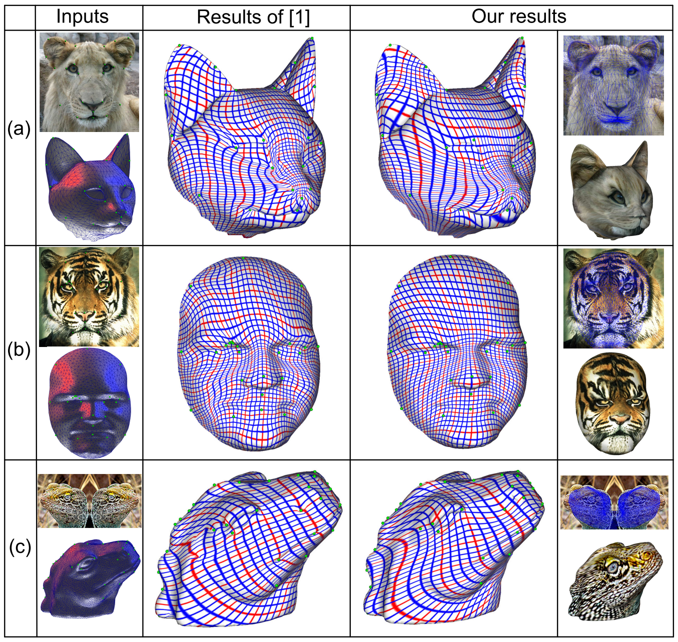


Fig. 12. Texturing models (a)-(c).

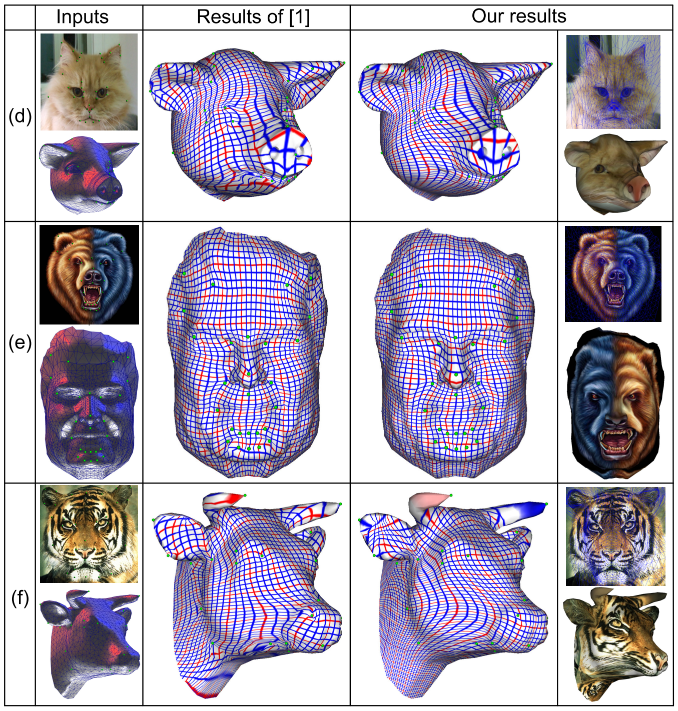


Fig. 13. Texturing models (d)-(f).

Table 1 shows some statistics of these texture mapping examples. The fifth column reports the running time of constructing the constrained mapping on an Intel Pentium 4, 3.6GHz PC with 1G RAM. The algorithm is implemented using C++. The sixth and eighth columns show the numbers of the added Steiner vertices using our LEB-based algorithm and [1], respectively. It can be found that except for model (f), the number of Steiner

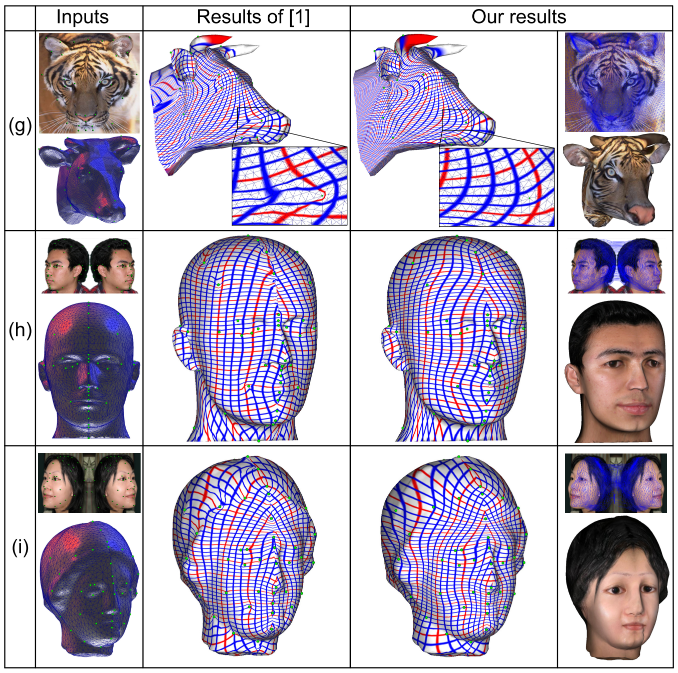


Fig. 14. Texturing models (g)-(i).

vertices added in our method is comparable to that of [1]. Model (f) is special because it contains two cusps in its two horns and has a low resolution. When the resolution increases, the number of Steiner vertices decreases as demonstrated in model (g). It is also worth pointing out that in [4] and [1] a postprocess that removes those unnecessary Steiner vertices is performed and the current implementation of our algorithm does not perform such a postprocess. In future we will incorporate the postprocess into our algorithm, by which the number of Steiner vertices is expected to be reduced significantly as in [4] and [1]. In addition, while we propose to use LEB-based refinement in this paper, there are other possibilities to refine the mesh. For example, we can split a triangle by bisecting its largest angle, which we call the largest angle bisection (LAB). We have tested our algorithm by replacing the LEB-based refinement by the LAB-based refinement, which usually results in fewer Steiner points. However, whether the LAB-based refinement always works is not clear yet, which warrants further investigation.

TABLE 1
Statistics of the texture mapping examples in Figures 12-14

Models	#Vertices	#Triangles	#Constrained vertices	Time (Sec)	#Steiner (LEB)	#Steiner (LAB)	#Steiner ([1])
model (a)	1770	3526	24	0.656	16	7	7
model (b)	1808	3602	25	0.063	0	0	10
model (c)	10017	20008	54	0.766	0	0	34
model (d)	1772	3450	21	0.469	16	4	21
model (e)	1657	3300	27	0.078	0	0	2
model (f)	1697	3384	32	11.593	1382	167	64
model (g)	10736	21404	32	3.922	54	4	8
model (h)	5184	10354	83	6.501	35	17	38
model (i)	4149	8284	71	1.719	6	2	25

The experimental results are also evaluated quantitatively using the stretches defined in [14]. Therein, the L_2 norm measures the root-mean-square stretch of the parameterization over all directions in the domain and the L_∞ norm represents the greatest stretch. A good parameterization is supposed to have small stretches. Table 2 presents the quantitative results of all these texture mapping examples. It can be seen that the stretches of our method are smaller than or comparable to the stretches of [1].

TABLE 2
Stretches of the texture mapping examples in Figures 12-14

Models	Proposed method		Method of [1]	
	L_2	L_∞	L_2	L_∞
model (a)	4.06561	78.958	2.79953	162.037
model (b)	1.20559	2.1982	1.20486	3.41913
model (c)	1.58255	22.452	373.289	83748.3
model (d)	3.10319	38.664	470.781	31045.9
model (e)	1.13447	4.11239	1.10433	4.2356
model (f)	564.94	23146.9	1160.47	66151.9
model (g)	1713.13	207825	1884.34	372774
model (h)	2.06733	104.374	85.2849	14365
model (i)	2.95445	79.2665	1285.42	262252

6 Limitations

Our work has a few limitations. First, it is restricted to models that are topologically equivalent to a disk. For a model with arbitrary topology, there is a need to partition it into segments, each with disk-topology, so that our algorithm can be used on them.

Second, the current construction of non-intersecting warping trajectories appears to be an engineering approach. It depends on the constraint points only and does not take the shape of the mesh into consideration, which may result in a large distortion and more Steiner points. Such an example is given in Figure 15. The input 3D mesh model and image are displayed in Figure 15(a) where the constraint points are highlighted in green squares. Figure 15(b) shows the trajectories generated by the proposed method, the warped 2D mesh and the texture mapping result. Each trajectory consists of two line segments, starting from a position labeled by a disk, passing through a position labeled by a circle and reaching a position labeled by a filled square. It can be seen that the two trajectories starting from points 3 and 4 cause the region bounded by points 1, 2, 3 and 4 in the 3D model to correspond to a very twisted region in the input image. As a result, the warping process causes the insertion of 706 Steiner points and the stretch amount of 1675 in L_2-norm and 166819 in L_∞-norm. Considering the shape of the input 3D model and the input image, we manually modify the two trajectories starting from points 3 and 4, as shown in Figure 15(c). Then the warped mapping and texture mapping are improved. In particular, the number of Steiner points decreases to 192 and the stretch amount decreases to 965 in L_2-norm and 44349 in L_∞-norm.

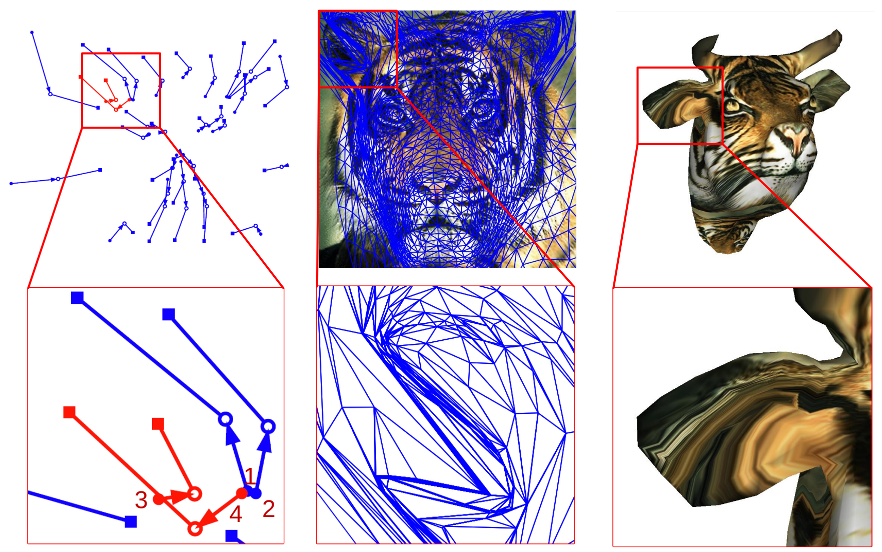
Third, it is worth pointing out that our current implementation uses ABF++ in the first phase, which may fail to produce a solution if the boundary of the model is very complicated. If this happens, ABF++ can be replaced by other methods such as [21] which guarantees a locally injective parameterization for an arbitrary fixed boundary. While our RBF-based interpolation usually gives low distortion in the second phase, the overall performance of the 3D to 2D mapping also depends on the performance of the parameterization in the first phase. Hence it is important to ensure the first step to produce a good unconstrained parameterization and to choose an appropriate boundary to reduce the distortion.

7 Conclusions

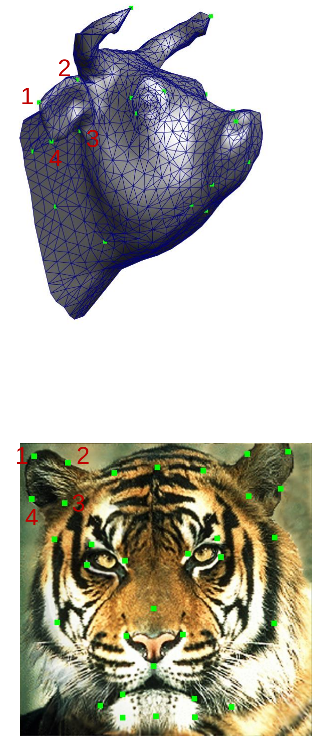
We have carefully analyzed the relation between a C^2 continuous 2D mapping and its induced piecewise linear transformation and proposed a refinement strategy based on the longest edge bisection, which guarantees a transformation induced from a locally bijective C^2 mapping, by adding a few Steiner vertices if necessary, to keep the orientation of each triangle of a 2D triangular mesh. Based on this, we present an efficient and theoretically robust texture mapping algorithm for triangular mesh models in the presence of hard constraints. The mapping is a composition of an unconstrained planar embedding and a series of constrained mesh transformations. The constrained mesh transformations are realized by a non-intersecting warping for constrained vertices, RBF-based interpolation and LEB-based refinement, which are proven to be foldover free. The condition for a mesh transformation to be foldover free and the condition for the RBF-based warp to be locally bijective are derived to determine the displacement step and the threshold for performing mesh refinement, which ensures the efficiency and validity of the algorithm. The use of RBF-based interpolation makes the mesh be smoothly deformed to align the user specified positional constraints exactly, without the need of performing a smoothing postprocess. The experiments with several examples of varying complexity demonstrate that the proposed algorithm can effectively handle hard constraints and produce visually pleasing texture mapping results.

Acknowledgments

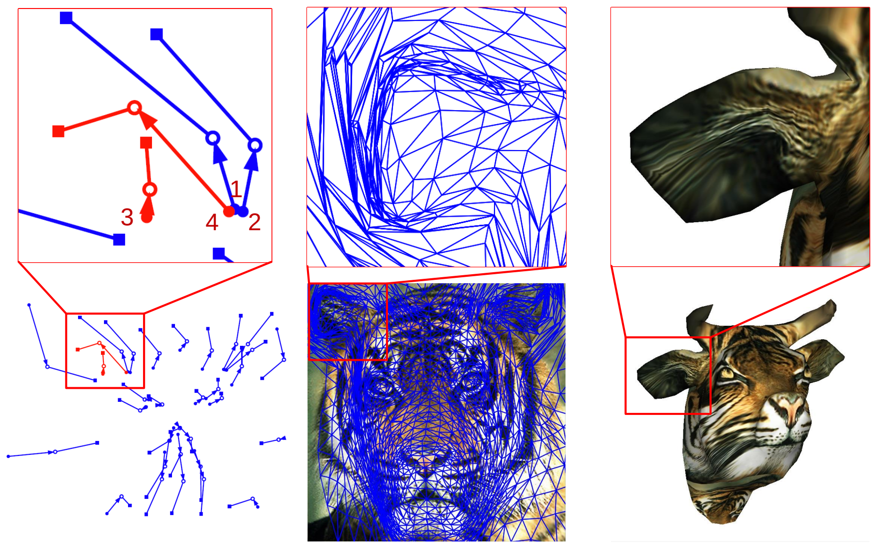
We gratefully recognize excellent feedback from reviewers. We thank Dr Hongchuan Yu at the National Center for Computer Animation, Bournemouth University for providing them some models for experiments. This research is partially supported by Multi-plAtform Game Innovation Centre (MAGIC), funded by the Singapore National Research Foundation under its IDM Futures Funding Initiative and administered by the Interactive & Digital Media Programme Office, Media Development Authority, and MOE Tier-2 grant (MOE2011-T2-2-041) of Singapore. Jian Xie is also supported by ZJNSF(LQ14A010011).



(b) Left: Trajectories generated automatically by the proposed method; Middle: The warped mapping; Right: Texture mapping result



(a) Input mesh & image



(c) Left: The two red trajectories are manually modified; Middle: The warped mapping; Right: Texture mapping result

Fig. 15. Warping trajectories: while the automatically generated trajectories shown in (b) may not fit well with the input 3D model shown in (a), modifying the two red trajectories can improve the mapping result as shown in (c).

REFERENCES

[1] T.-Y. Lee, S.-W. Yen, and I.-C. Yeh, "Texture mapping with hard constraints using warping scheme," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 2, pp. 382 –395, March-April 2008.

[2] B. Lévy, "Constrained texture mapping for polygonal meshes," in *Proceedings of SIGGRAPH'01*. New York, NY, USA: ACM, 2001, pp. 417–424.

[3] I. Eckstein, V. Surazhsky, and C. Gotsman, "Texture mapping with hard constraints," *Computer Graphics Forum*, vol. 20, no. 3, pp. 95–104, 2001.

[4] V. Kraevoy, A. Sheffer, and C. Gotsman, "Matchmaker: constructing constrained texture maps," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 326–333, Jul. 2003.

[5] Y. Tang, J. Wang, H. Bao, and Q. Peng, "RBF-based constrained texture mapping," *Computers &amp; Graphics*, vol. 27, no. 3, pp. 415 – 422, 2003.

[6] H. Seo and F. Cordier, "Constrained texture mapping using image warping," *Computer Graphics Forum*, vol. 29, no. 1, pp. 160–174, 2010.

[7] Y. Tzur and A. Tal, "Flexistickers: Photogrammetric texture mapping using casual images," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 45:1–45:10, 2009.

[8] A. Sheffer, B. Lévy, M. Mogilnitsky, and A. Bogomyakov, "Abf++: fast and robust angle based flattening," *ACM Trans. Graph.*, vol. 24, no. 2, pp. 311–330, Apr. 2005.

[9] H. Yu, T.-Y. Lee, I.-C. Yeh, X. Yang, W. Li, and J. Zhang, "An RBF-based reparameterization method for constrained texture map-

ping," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 7, pp. 1115 –1124, july 2012.

[10] W. Zhang, Y. Ma, and J. Zheng, "Inversion free and topological compatible tetrahedral mesh warping driven by boundary surface deformation," in *Proceedings of 2013 International Conference on Computer-Aided Design and Computer Graphics*. IEEE, 2013, pp. 228–235.

[11] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution analysis of arbitrary meshes," in *Proceedings of SIGGRAPH'95*. New York, NY, USA: ACM, 1995, pp. 173–182.

[12] M. S. Floater, "Parametrization and smooth approximation of surface triangulations," *Computer Aided Geometric Design*, vol. 14, no. 3, pp. 231 – 250, 1997.

[13] ——, "Mean value coordinates," *Computer Aided Geometric Design*, vol. 20, no. 1, pp. 19 – 27, 2003.

[14] P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe, "Texture mapping progressive meshes," in *Proceedings of SIGGRAPH'01*. New York, NY, USA: ACM, 2001, pp. 409–416.

[15] B. Lévy, S. Petitjean, N. Ray, and J. Maillot, "Least squares conformal maps for automatic texture atlas generation," in *Proceedings of SIGGRAPH'02*. New York, NY, USA: ACM, 2002, pp. 362–371.

[16] M. S. Floater and K. Hormann, "Surface parameterization: a tutorial and survey," in *Advances in Multiresolution for Geometric Modelling*, ser. Mathematics and Visualization, N. A. e. a. Dodgson, Ed. Springer Berlin Heidelberg, 2005, pp. 157–186.

[17] A. Sheffer, E. Praun, and K. Rose, "Mesh parameterization methods and their applications," *Found. Trends. Comput. Graph. Vis.*, vol. 2, no. 2, pp. 105–171, Jan. 2006.

[18] Y. Xu, R. Chen, C. Gotsman, and L. Liu, "Embedding a triangular graph within a given boundary," *Computer Aided Geometric Design*, vol. 28, no. 6, pp. 349–356, 2011.

[19] Y. Lipman, "Bounded distortion mapping spaces for triangular meshes," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 108:1–108:13, 2012.

[20] T. Schneider, K. Hormann, and M. S. Floater, "Bijective composite mean value mappings," *Comput. Graph. Forum*, vol. 32, no. 5, pp. 137–146, 2013.

[21] O. Weber and D. Zorin, "Locally injective parametrization with arbitrary fixed boundaries," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 75:1–75:12, Jul. 2014.

[22] M. Desbrun, M. Meyer, and P. Alliez, "Intrinsic parameterizations of surface meshes," *Computer Graphics Forum*, vol. 21, no. 3, pp. 209–218, 2002.

[23] Y. I. Gingold, P. L. Davidson, J. Y. Han, and D. Zorin, "A direct texture placement and editing interface," in *Proceedings of the 19th annual ACM symposium on User interface software and technology*. New York, NY, USA: ACM, 2006, pp. 23–32.

[24] J. Schreiner, A. Asirvatham, E. Praun, and H. Hoppe, "Inter-surface mapping," in *Proceedings of SIGGRAPH'04*. New York, NY, USA: ACM, 2004, pp. 870–877.

[25] T.-Y. Lee and P.-H. Huang, "Fast and intuitive metamorphosis of 3d polyhedral models using smcc mesh merging scheme," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 1, pp. 85 – 98, Jan.-March 2003.

[26] B. Tiddeman, N. Duffy, and G. Rabey, "A general method for overlap control in image warping," *Computers & Graphics*, vol. 25, no. 1, pp. 59–66, 2001.

[27] K. Fujimura and M. Makarov, "Foldover-free image warping," *Graph. Models Image Process.*, vol. 60, no. 2, pp. 100–111, 1998.

[28] S. Korotov, M. Krizek, and A. Kropac, "Strong regularity of a family of face-to-face partitions generated by the longest-edge bisection algorithm," *Computational Mathematics and Mathematical Physics*, vol. 48, no. 9, pp. 1687–1698, 2008.

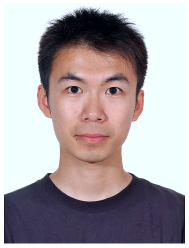
[29] V. Surazhsky and C. Gotsman, "Controllable morphing of compatible planar triangulations," *ACM Trans. Graph.*, vol. 20, no. 4, pp. 203–231, Oct. 2001.

[30] A. Saalfeld, "Joint triangulations and triangulation maps," in *Proceedings of the 3rd Annual Symposium on Computational Geometry*. New York, NY, USA: ACM, 1987, pp. 195–204.

[31] D. L. Souvaine and R. Wenger, "Constructing piecewise linear homeomorphisms," DIMACS Technical Report 94-52.

Yuewen Ma received his PhD degree from School of Computer Engineering, Nanyang Technological University, Singapore, in 2014. He is currently a research engineer at Huawei Technologies (Huawei Central Research Institute). His research interests include computer graphics and computer aided geometric design.

Jianmin Zheng received the B.S. and Ph.D. degrees from Zhejiang University, China. He is an Associate Professor in the School of Computer Engineering with Nanyang Technological University. His current research interest includes computer aided geometric design, computer graphics, animation, visualization, and interactive digital media. He has published more than 100 technical papers in international conferences and journals. He is an Associate Editor of The Visual Computer.



Jian Xie received the A.B. and PhD in Mathematics from Zhejiang University, Hangzhou, China, in 2006 and 2011, respectively. Since 2011, he has been a lecturer with the Department of Mathematics, Hangzhou Normal University. His main research interest is Partial Differential Equations.