Constrained Texture Mapping and Foldover-Free Condition

Hongchuan Yu, Xiaosong Yang, and Jian J. Zhang

Abstract—**Texture mapping has been widely used in image processing and graphics to enhance the realism of CG scenes. However to perfectly match the feature points of a 3D model with the corresponding pixels in texture images, the parameterisation which maps a 3D mesh to the texture space must satisfy the positional constraints. Despite numerous research efforts, the construction of a mathematically robust foldover-free parameterisation subject to internal constraints is still a remaining issue. In this paper, we address this challenge by developing a two-step parameterisation method. First, we produce an initial parameterisation with a method traditionally used to solve structural engineering problems, called the bar-network. We then derive a mathematical foldover-free condition, which is incorporated into a *Radial Basis Function* based scheme. This method is therefore able to guarantee that the resulting parameterization meets the hard constraints without foldovers.**

Index Terms—**Foldover, constrained texture mapping, parameterization.**

I. Introduction

Texture mapping is an effective means in image processing and graphics to achieve improved visual realism. Existing research has largely concentrated on the production of planar parameterization [1]-[7] in order to map a 3D mesh to the planar domain. Most recent works are concerned with texture distortion reduction when mapping a planar image (texture) to a curved surface. Although this is an important issue, in practice the animator is also challenged with other problems in texture mapping. One of them is to register a texture map accurately with the features of a 3D model. For example, if one is to texture map a human face, in addition to reducing texture distortion, one has to ensure the important feature points and lines on the 3D model match those on the texture plane during the mapping process, such as the eyes, nose, eyebrows and lips. In another word, one needs to accurately register the 3D features with their 2D counterparts.

With the current production practice, this registration operation is almost completely manual. Once a texture map is generated by animation software, the animator has to painstakingly tweak the unwrapped mesh on the texture plane to align the key feature points on the texture image with the 3D features. He/she has to manually move many vertices around each feature in order to avoid texture distortion being concentrated and visible. This is a time-consuming task.

Manuscript received September 18, 2012; revised December 7, 2012.

The authors are with the National Center for Computer Animation, Bournemouth University, Poole, UK. (email:cnyuhc@yahoo.com, xyang@bournemouth.ac.uk, jzhang@bournemouth.ac.uk).

Attempts have been made to formulate it as a constrained optimization problem [3]-[5],[7] where the important features in the texture images are to be located correctly on the 3D surface. Despite varying degree of success, a key issue yet to be solved is there is currently no robust solution to controlling the spread of the mesh points such that the generated new mesh is mathematically predictable. For a texture point to be reliably mapped to the corresponding position on the 3D surface, one must ensure there exists a one-to-one mapping between the 2D and 3D domains. In other words, there must be no mesh foldovers during the unwrapping process. To the best of our knowledge, there is no robust solution exists, which mathematically guarantees the elimination of mesh foldovers during mesh manipulation.

In this paper, we present a novel constrained texture mapping method by developing a foldover-free parameterisation. The user can interactively specify the feature points where constraints are set between the corresponding 3D vertices on the mesh and the 2D points on the texture image. We first generate an initial parameterisation by unwrapping the 3D mesh using a method from structural mechanics, called the bar-networks, without considering any internal constraints. We will demonstrate that this parameterisation is foldover-free.

In order to satisfy the mesh constraints placed by the user, we then re-parameterise the initial mesh using *Radial Basis Functions*. By incorporating a novel foldover-free condition, our method will guarantee the resulting re-parameterization is free of foldovers.

In comparison with the other methods, ours has the following advantages:

- Foldover-free. Our work is primarily concerned with a robust foldover-free parameterisation. We first derive an explicit mathematical condition which guarantees no mesh foldover occurs during the parameterisation. This is called the *foldover-free condition*. By incorporating this condition, a *RBF-based* re-parameterization algorithm is then developed accordingly to produce foldover-free meshes satisfying constraints.

- High efficiency. Our method is based on C^2 continuous mapping functions, which avoids the discontinuity problem at some edges suffered by the Delaunay triangulation-based methods [3], [8]. As a result, no extra Steiner points are needed in our algorithm. In addition, since our calculation is based on *Radial Basis Functions*, only a small number of constraint points are involved. Updating the *RBF* coefficients is both cheap and relatively straightforward. There is no need to introduce extra optimization operation steps to improve the quality of parameterisation.

- Little distortion. Our *RBF-based* scheme is globally supported. It can be decomposed into a global affine and a local smoothing component. The properties ensure the errors are naturally spread resulting in little mesh distortion.
- Robustness. In addition to a mathematically robust solution, our experiments also demonstrate that the developed method can handle a large number of user-specified constraints, some of which are topologically complex.

A. *Related Works*

To map a texture image onto a 3D surface model, a correspondence needs to be established between the 3D mesh and its 2D parametric plane. This process is called parameterisation. A lot of work has been carried out in the past years. Some [2], [6] targeted the validity of the resulting parameterization, e.g. bijective mapping. Some tried to minimize the distortion according to different distortion metrics [6], [9]-[12].

The above mentioned methods can be roughly classified into two categories. One is to map a 3D mesh onto a 2D region with a specified convex boundary [2], [6], [9], [13]. The others tried to solve the problem without considering the boundary constraints [11]-[12], [14] in order to achieve a lower level of distortion. However little consideration was given for satisfying internal positional constraints. In practice, when an animator maps a texture image, such as a photograph of a human face, onto a 3D head model, the feature pixels (e.g. the corners of the lips, eyes and nose) should be mapped exactly onto the correct locations on the 3D mesh. Some research was undertaken to meet the soft constraints [15], i.e. to satisfy the inside positional constraints approximately. Levy [5] proposed a “least-squares” method to solve the constraints. However it did not provide a valid parameterization for situations where there are a large number of constraints. Hard constraints were further studied in [4], [16], since a perfect alignment of texture is essential at certain delicate places of a mesh. Tang *et al.* [17] proposed a RBF-based parameterisation method. But the relationship between deformation and foldover was not investigated. Kraevoy *et al.* [3] and Lee [8] applied Delaunay triangulation to mesh parameterization for converting internal constraints to convex boundaries’ ones. But post-processing is usually necessary, which checks and corrects foldover triangles, leading to additional computation costs and instability.

Because meshes are being moved during optimisation, a main problem encountered with constrained texture mapping is mesh foldover. Unfortunately little success was made to robustly remove this problem. With the possibility of mesh foldovers, solving an optimisation problem subject to hard constraints becomes computationally much more expensive than otherwise. In addition, the overlapped parts of a mesh will usually need manual tweaking, which further increases the production cost.

The rest of this paper is organized as follows. Section 2 demonstrates the triangle foldover problem encountered with various existing methods. Section 3 introduces our bar-net based initial parameterisation method. In Section 4, we first derive the foldover-free condition. A *RBF-based* re-parameterization algorithm is then introduced by incorporating the foldover-free condition. Experiment results and discussions are given in Section 5. The conclusions and future work are presented in Section 6.

II. Foldover

Foldovers within a mesh can be observed when the internal positional constraint points are added into the parameterization of a polygonal mesh. Figure 1 shows some examples of parameterization, which satisfy the given constraints. The 3D mesh is mapped onto a fixed 2D region whose boundary is given in advance. In order for us to have a clear picture of the existing parameterisation methods with regard to their ability to overcome the foldover problem, we have implemented four methods as shown in Figure 1b-1e. One can remove foldover triangles by either adding Steiner vertices to subdivide the triangles or swapping the edges. This both increases the computation burden and changes the original topology. Further expenses arise when it is necessary to detect and modify foldover meshes and minimise the resulting distortion.

Our experience from implementing the four methods suggests that none of them are able to robustly overcome the mesh foldover problem. It is still an unsolved issue in mesh parameterisation. In the following we present our solution to this issue. We will first develop an initial foldover-free parameterisation using a method from classic structural engineering, and then derive a mathematical foldover-free condition. After that, we will formulate a *RBF-based* algorithm by which the final parameterisation can be obtained from the initial one and will satisfy all the constraints set by the user.

III. Initial Parameterization

In this section we discuss the first step of our foldover-free algorithm, to develop an initial parameterization using a method from the classic structural engineering, called the bar-network (bar-net for short). This initial parameterization does not take consideration of the user-specified internal constraints within the mesh. But it is necessary for the subsequent refinement, which starts from a foldover-free parameterization and will eventually satisfy all the constraints.

A. *Bar-Net*

A *bar-net* [18] is a structure commonly used in structural engineering. Its shape depends on the structural and material properties and the forces acting upon it. A bar-net connects n_s points, in three-dimensional space with straight-line segments, called *bars*. These points on the net are known as the *nodes*. The nodes can be either fixed or free. Fixed nodes do not change their positions whether subject to external forces or not. Free nodes can be moved to balance the acting forces on the net. Each bar connects two nodes. These bars can be stretched and squashed resulting in the repositioning of the end nodes, but they remain topologically constant. The final shape represents the rest shape of the network and is the result of the balance of all external and internal forces.

Assuming there are n_s nodes (n free nodes and n_f fixed nodes) and m bars in the network. A matrix $C_s(m*n_s)$ called the branch-node matrix can be formed, which represents in a tabular form the graph of the network. For each row which represents a bar, the elements on the column i and j which are the two linked nodes indices, will be set *1* and *-1*, all others will be set to *0*. This branch-node matrix can be further subdivided into two sub-matrices, C and C_f, by grouping the free-node columns and fixed-node columns of the original matrix respectively. These matrices are used in computing the rest shape of a bar-net. The effect of stiffness of a network can be approximated by the quantity of force-length ratios of all the bars, called the *force density*. One major advantage of this method is it is able to solve the form finding problem with a set of sparse linear equations. A detailed description is given in [18].

Equation (1) shows how to compute the coordinates of the free nodes at the equilibrium state:

$$\begin{cases} \vec{x} = D^{-1}(\vec{p}_x - D_f \vec{x}_f) \\ \vec{y} = D^{-1}(\vec{p}_y - D_f \vec{y}_f) \\ \vec{z} = D^{-1}(\vec{p}_z - D_f \vec{z}_f) \end{cases} \quad (1)$$

where $D = C^T QC$ and $D_f = C^T QC_f$, Q is the diagonal matrix (q_{ii}), where q_{ii} is the force density of the bar i. $(\vec{p}_x, \vec{p}_y, \vec{p}_z)$ are the external force vectors. Therefore, with a given interconnection, the force density vector, the external forces and the coordinates of the fixed nodes, the positioning of the free nodes are determined by the equilibrium of the forces.

B. Bar-Net Based Initial Parameterization

There are many different unwrapping methods to map a 3D mesh to a 2D plane to produce a parameterization [2],[6],[11],[19]-[22]. To a great extent, they all are able to produce an initial parameterization. We use the bar-net method, because, as can be seen below, the mechanical properties allow it to produce a foldover-free parameterization naturally, a prerequisite for our subsequent *RBF-based* operations. Another advantage is it is computationally very efficient, as it involves solving only one sparse matrix.

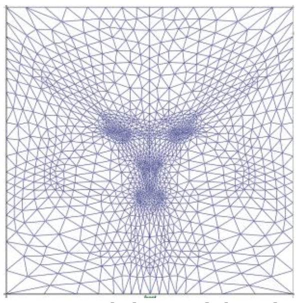
Fig. 2. Bar-net unwrapping with the outside boundary constraints only.

The crucial part of this method is to find a mapping between a 3D mesh and the 2D square. Before we get into the detail of our unwrapping method, we need to build up the correspondence between these two meshes. First from the connectivity of the edges and vertices, we identify the boundary edge loop B of the 3D mesh. Boundary B will be mapped to the 2D square's four edges. All other holes and internal boundaries are mapped to the inside area of the 2D square. On boundary B we can automatically choose four points which are evenly distributed along the edge loops. These four points v_k ($k=0...3$) are mapped to the four corners of the 2D square. All other vertices on the boundary loop are mapped to the four edges of the square accordingly. The whole boundary edge loop can then be split into four sections from the original mesh's vertex connection. In order to map the 3D patch to the 2D plane, we now define the new position of each vertex on the boundary edges to be aligned to the four edges of the square $[0,1]\times[0,1]$ according to their original edge length in the 3D mesh and again all these points will be set as fix points. If the 3D mesh has more than one boundary loop, all the vertices on the other boundaries (except B) will be set as free ones at this first unwrapping step. The vertices on B are fixed during the *bar-net* embedding while leaving all inside vertices to freely settle inside the square. Matrices C and C_f can be defined from the connection relationships of the bars and the nodes within the network. In order to keep the connection topology of the 3D mesh unchanged, we define the force density q_i of edge i according to its original length $Len(e_i)$.

$$q_i = 1 - \frac{Len(e_i)}{Max(Len(e_j), j = 1..m)} + \varepsilon \quad (2)$$

where ε is a very small value to prevent q from zero, m is the number of edges involved in this patch. The external force vectors, $p_x = p_y = p_z = 0$, so all the free nodes will stay on the 2D plane and remain inside the 2D square. The final equilibrium state of this *bar-net* will be the 2D mapping of the 3D mesh. Figure 2 shows an example of the unwrapping of the front part of a human face model (Figure 1a).

From the mechanics point of view, the rest state of such a network, i.e. an equilibrium state, requires the resulting internal force to be 0 at every internal node. Any foldover can only occur when there are extra external forces applied to some of the internal nodes, i.e. away from equilibrium, as seen in Fig. 3. As a result, once a bar-net has reached its equilibrium, it naturally rules out the possibility of foldover. Since our above method is equivalent to finding a rest shape of a bar-net, there is no room for foldovers.

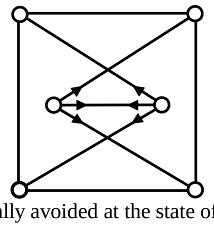
Fig. 3. Foldover is naturally avoided at the state of equilibrium of a bar-net

IV. 2D RBF-Based Re-Parameterization

Based on the initial foldover-free parameterisation produced above, let us now discuss how a new foldover-free parameterisation can be formulated, which satisfies a set of given internal constraints. From a mathematic point of view, a "*foldover-free*" parameterisation gives a "*one-to-one*" mapping between the corresponding 2D and 3D meshes.

Many researchers have attempted to produce a foldover-free parameterisation. We believe that to find a robust answer one needs to define a criterion which is able to provide a definitive check. In the following we first derive a mathematical condition, which will guarantee a *one-to-one* mapping. Guarded by this condition, we further develop a *RBF-based* algorithm to re-parameterise the meshes leading to a new parameterisation where all the constraints are satisfied.

A. Foldover-Free Condition

For a given 2D mesh S of R^2, a transformation T is a one-to-one mapping which maps the points $X \in S$ into another 2D subdomain $U \in \Omega$ of R^2 with arbitrary m constraint point pairs $\left(X_i^* \leftrightarrow U_i^*\right)$, i.e.

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

In a 2D re-parameterization, starting from an initial one which satisfies the one-to-one mapping property, if this property is to be preserved, the mesh topology and their connection relationship should also be preserved. 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 stay unchanged before and after the parameterisation. A more accurate description is that the determinant of the Jacobian matrix must be positive everywhere, i.e.

$$\det(\nabla U) > 0 . \tag{3}$$

It is well known that the determinant of the Jacobian matrix ∇U is equal to the product of its two eigenvalues. We bound the spectrum of ∇U according to the Gerschgorin circle theorem as follows,

$$\left|\sigma - \frac{\partial u}{\partial x}\right| < \left|\frac{\partial u}{\partial y}\right| \text{ or } \left|\sigma - \frac{\partial v}{\partial y}\right| < \left|\frac{\partial v}{\partial x}\right|.$$

The real and imagery parts of the eigenvalues are limited respectively within,

$$\begin{cases} \operatorname{Re}\{\sigma\} \in \left[\frac{\partial u}{\partial x} - \left|\frac{\partial u}{\partial y}\right|, \frac{\partial u}{\partial x} + \left|\frac{\partial u}{\partial y}\right|\right] \bigcup \left[\frac{\partial v}{\partial y} - \left|\frac{\partial v}{\partial x}\right|, \frac{\partial v}{\partial y} + \left|\frac{\partial v}{\partial x}\right|\right] \\ \operatorname{Im}\{\sigma\} \in \left[-\left|\frac{\partial u}{\partial y}\right|, \left|\frac{\partial u}{\partial y}\right|\right] \bigcup \left[-\left|\frac{\partial v}{\partial x}\right|, \left|\frac{\partial v}{\partial x}\right|\right] \end{cases}. \tag{4}$$

In terms of Equation (4), a sufficient condition of satisfying Equation (3) can be described as,

$$\begin{cases} \frac{\partial u}{\partial x} > \left|\frac{\partial u}{\partial y}\right| \\ \frac{\partial v}{\partial y} > \left|\frac{\partial v}{\partial x}\right| \end{cases}. \tag{5}$$

The geometric meaning is simply saying that the two vectors $\partial u/\partial(x,y), \partial v/\partial(x,y)$ are linearly independent of each other, and their included angle is less than π. The former is easy to understand. The latter implies that the right-hand rule is satisfied all over the domain. If det(∇*U*)<0 at some point, this would result in the left-handedness instead of the right-handedness. It can be imagined that the change of the right-handedness to left-handedness at some point would cause the foldover of mesh.

This is called the *foldover-free condition*. In the following, we incorporate this condition into our RBF-based re-parameterisation scheme, which will then preserve the one-to-one mapping property.

B. RBF-Based Re-Parameterization Scheme

The RBF scheme is a mesh-free approach, and is C^2 continuous. This is suited for applying Equation (5) to it. What needs noting is unlike the previous applications of RBFs [16,17], in our case the RBF scheme is used to compute the displacement of the point's coordinates,

$$\Delta X = P(X) + \sum_i^m \lambda_i \phi(\|X - C_i\|) \tag{6}$$

where the coefficient $\lambda_i = \left(\lambda_x^i, \lambda_y^i\right)^T$ is a vector, $C_i = \left(c_x^i, c_y^i\right)^T$ denotes the constraint points, $\Delta X = \left(\Delta x, \Delta y\right)^T$ and $P(X)$ is a affine transformation $P(X) = \begin{pmatrix} a & b \\ c & d \end{pmatrix} X + \begin{pmatrix} e_x \\ e_y \end{pmatrix}$. Although there are various radial basis functions, we adopt the thin plate spline (i.e. $\phi(r) = r^2 \ln r$) as ϕ here for simplicity. The deformed U is obtained by updating $U(X) = X + \Delta X$.

Consider the condition of Equation (5). Substituting Equation (6) to it yields,

$$\begin{cases} 1 + a + \sum_i^m \lambda_x^i \frac{d\phi_i}{dr_i} \frac{x - c_x^i}{r_i} > \left| b + \sum_i^m \lambda_x^i \frac{d\phi_i}{dr_i} \frac{y - c_y^i}{r_i} \right| \\ 1 + d + \sum_i^m \lambda_y^i \frac{d\phi_i}{dr_i} \frac{y - c_y^i}{r_i} > \left| c + \sum_i^m \lambda_y^i \frac{d\phi_i}{dr_i} \frac{x - c_x^i}{r_i} \right| \end{cases}. \tag{7}$$

Equation (7) is the concrete representation of our RBF scheme incorporating the foldover-free condition. Now let us summarise our RBF-based re-parameterization scheme as follows,

1) Initial mesh $S^{(0)}$ and a set of user-specified constraint points $C_i^*, i = 1,...,m$,
2) Loop: Updating the current constraint points by $C_i = tC_i^{(0)} + (1-t)C_i^*, C_i^{(0)} \in S^{(0)}$,
3) Checking if the foldover-free condition of Equation (7) is satisfied,
4) If yes, computing the displacements of points on *S* by Equation (6) and updating *S*,
5) Otherwise reducing *t* and go to step 2,
6) End Loop until $C_i = C_i^*$.

Because the RBF scheme can be decomposed into a global affine and a local smoothing components, only the

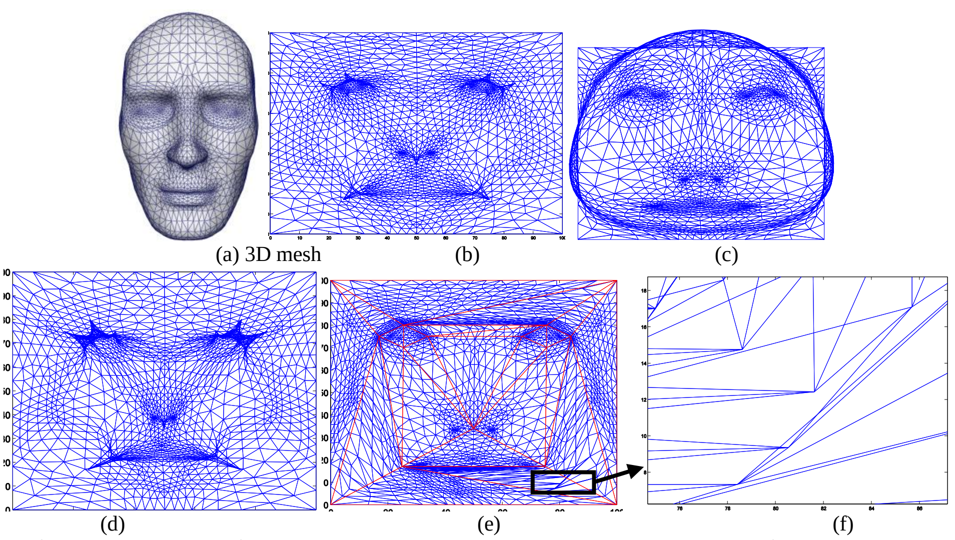

Fig. 1. Illustration of the foldover results using four recent parameterization methods taking consideration of internal hard constraints (Note: the mesh details can be seen more clearly by zooming in the document). (a) 3D mesh, (b) least squares meshes [23], (c) RBF-based embedding [17], (d) harmonic mapping and (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), (f) further shows the details of distortion around the red line.

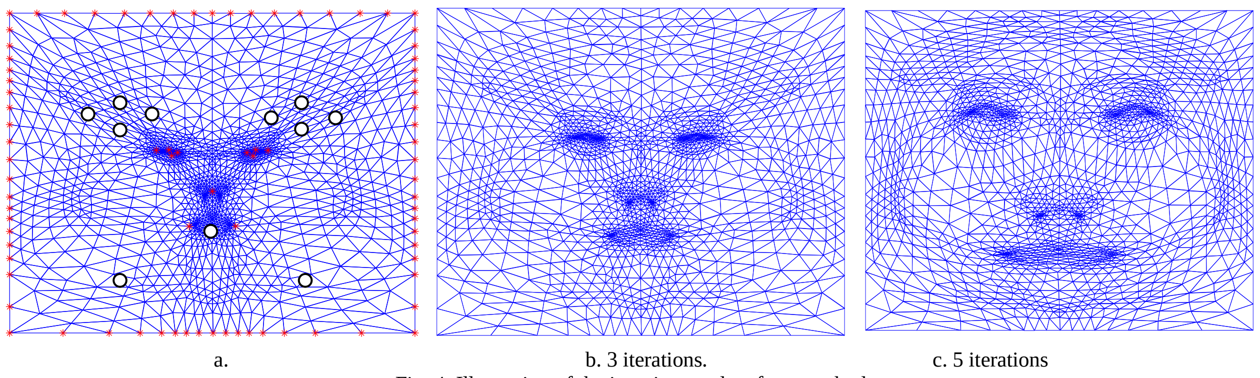

Fig. 4. Illustration of the iterative results of our method

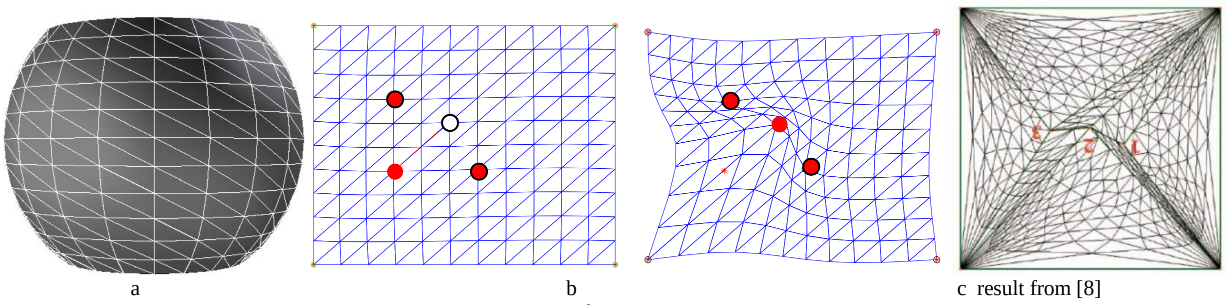

Fig. 5. Distortion from re-parameterization.

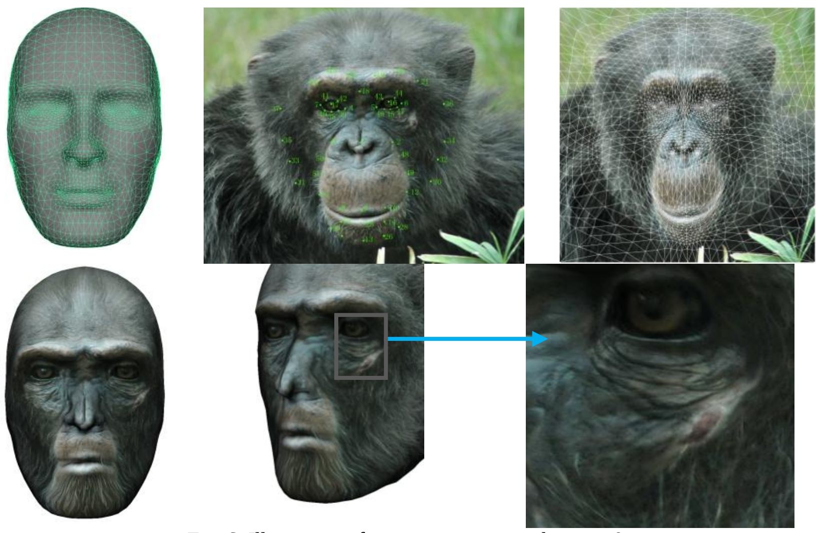

Fig. 6. Illustration of texture mapping with genus 0.

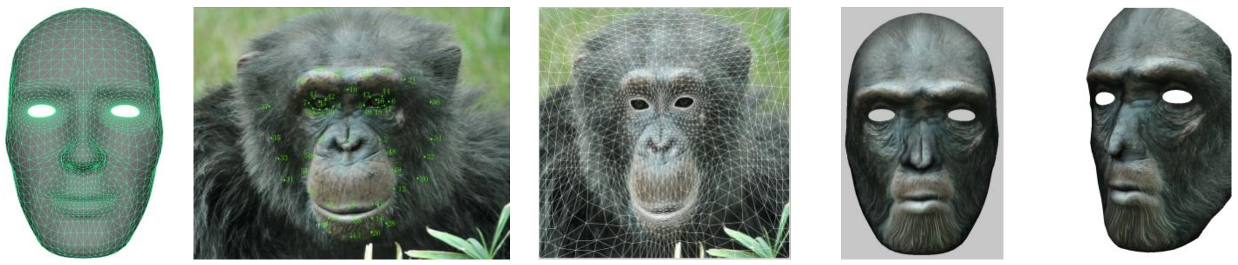
Fig. 7. Illustration of texture mapping with genus 2.

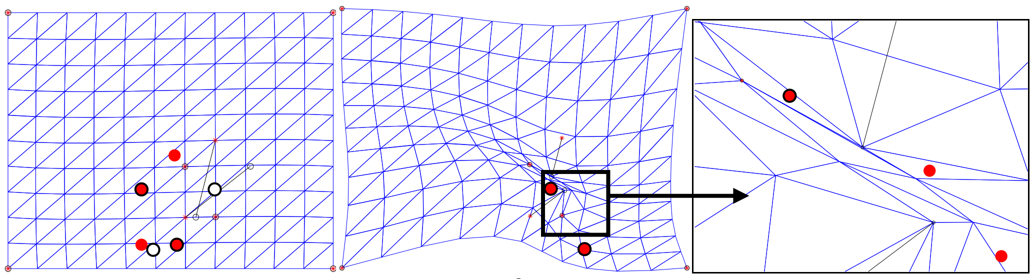

Fig. 8. Illustration of convergence of an extreme case.

TABLE I: EXPERIMENT STATISTICS

Experiment	Vertex	Num Of constraints (k)	Num Of Iteration (m)	Distoration		Total time (seconds)
				L-2	L-Inf	
Fig. 6	1672	51	5	0.1019	0.4447	6
Fig. 7	1672	62	5	0.1718	0.7714	6.8

local smoothing component may result in foldovers. In practice, large deformation often occurs around the constraint points. If they can satisfy the condition of Equation (7), their neighbours must be able to satisfy it. To save time, we only check the constraint points at each iteration in our implementation.

The core of our RBF-based re-parameterization is to update the RBF coefficients at each iteration. The main cost is therefore to compute the inverse of a real symmetric matrix, which costs $O(m^3)$, where m is the number of the constraint points. The time complexity can be estimated as $O(km^3)$, where k denotes the iteration number. In texture mapping applications, the number of constraint points is far smaller than that of the vertices, around 10-100. And our algorithm usually converges with 3-5 iterations.

V. EXPERIMENTS AND DISCUSSIONS

In this section, we will apply the above-developed method to a number of examples in order to evaluate its validity, efficiency and robustness.

A. Foldover

Fig. 1 shows the results created with the recently published methods, which, as can be seen from the figure, are not able to avoid foldovers completely during the parameterisation process. In order to make a comparison with them, our first example is to test the above-developed method on the same head model shown in Figure 1a. We first produce an initial mesh by the bar-net parameterisation and the result is given in Fig. 2. The result of each iteration with the *RBF-based* re-parameterisation is shown in Figure 4. In Figure 4a, the points marked with the red stars denote the constraint points, which need to move to the points with white circles. The results confirm that by satisfying the foldover-free condition at each iteration, no foldover triangles appear during the iteration.

B. Distortion

To study the distortion of our algorithm, we use the stretch metrics defined in [6]. The L-2 norm is used to measure the overall stretch of the parameterization, while the L-Inf is used to measure the greatest stretch. A good parameterization is expected to have very small L-2 and L-Inf. We use these two metrics to measure the distortion of our three experiments (see Table I). Here we also use an extreme example similar to the Fig. 12 in [8]. We use a spherical patch as an example for the test. The patch is first flattened with our bar-net based initial parameterisation, and then followed by the *RBF-based* re-parameterization scheme with some specified constraint points. To highlight the issue of distortion, in Figure 5a we request only one constraint point to move to a new location (white circle). The orientation of the corresponding constraint points is changed. It can be seen that our approach does not result in large deformation. Compared with the result (Fig. 5 (c))) from [8], in terms of deformation, ours is more desirable.

C. Texture Mapping

We texture mapped an orangutang's photo onto a 3D human head model with genus 0 and genus 2, respectively, as shown in Fig. 6 and 7. The zoomed-in detailed image shows that our method produces a very smooth parameterization. These two examples also demonstrate that our method is very fast. It converged in only five iteration steps, even when a large number of constraints are involved, which are normally required in animation production in order to achieve high-quality texture mapping results.

The experimental statistics of these examples are given in Table I. In addition to being foldover-free, the figures also confirm that it also enjoys low unwanted distortion. The experiments were undertaken using Matlab on an Intel Pentium 4 3.2GHz PC with 1 Gbyte of RAM. Although the code is far from optimised, because our method has a low computation complexity as discussed earlier, it is very fast.

VI. Conclusions

Texture mapping requires a 3D surface to be optimally parameterised in terms of its deformation and mesh topology. Although mesh distortion has been extensively studied by the computer graphics community, the problem of mesh foldover has not seen a valid solution when there are hard constraints to be satisfied.

In this paper, we have presented a two-step parameterization method to solve this particular issue. The first step is to produce an initial parameterisation without considering the internal constraints. Although other methods are applicable, our work is based on a method used in classic structural engineering, call the bar-network. Its physical properties naturally ensure a foldover-free structure. It is also very cheap to compute, as there is only one sparse matrix to be calculated. Once this initial parameterisation arrives, we have formulated a mathematical condition, called the foldover-free condition, which guarantees to maintain the connection relationship of a mesh during the parameterisation process. Guided by this condition, our second step is to develop a *RBF-based* re-parameterisation scheme. Basically, the meshes will be iteratively moved to satisfy the constraints and at each iteration, the foldover-free condition is used to check the validity of the movement. Our experiments suggest our method works very well even in situations where there exist a many constraints.

As the *RBF* scheme consists of two components, global affine component and local smoothing component, the resulting 2D parameterisation has very small distortion.

The foldover-free condition usually requires checking all points. In practice, however, only the constraint points can cause large deformation around them, because they are fixed in the mesh. To speed up our approach, practically we only need checking these constraint points. Our complexity analysis suggests that the computation cost is low.

Due to various constraint configurations, we have found that the final mesh may not always converge to the most ideal positions, although it will produce a valid useable parameterisation. Looking at Figure 8 for example, there are two constraint points to be moved in opposite directions. Occasionally their moving paths may intersect, e.g. moving from a red point to the white point. Our approach can converge at an intermediate state. This is because with our *RBF* scheme, the moving paths of the constraint points are straight lines during the whole iteration procedure. One possible solution is to change the moving paths of the constraint points to a polyline. In the future this issue will be further investigated.

References

[1] M. Desbrun, M. Meyer, and P. Alliez, “Intrinsic Parameterizations of Surface Meshes,” in *Proc. of Eurographics’02/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. On Graphics*, vol. 22, no. 3, pp. 326-333, 2003.

[4] I. Eckstein, V. Surazhsky, and C. Gotsman, “Texture Mapping with Hard Constraints,” in *Proc. Of Eurographics’01/Computer Graphics Forum*, vol. 20, no. 3, pp. 95-104, 2001.

[5] B. Le´vy, “Constrained Texture Mapping for Polygonal Meshes,” in *Proc. of ACM SIGGRAPH’01*, pp. 417-424, 2001.

[6] P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe, “Texture Mapping Progressive Meshes,” in *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. On 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. on Visualization and Computer Graphics*, vol. 14, no. 2, pp. 382-395, 2008

[9] M. Eck, T. Derose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, “Multiresolution Analysis of Arbitrary Meshes,” in *Proc. of 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. on Visualization and Computer Graphics*, vol. 6, no. 2, pp. 181-189, 2000.

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

[12] M. Desbrun, M. Meyer, and P. Alliez, “Intrinsic Parametrizations of Surface Meshes,” in *Proc. of Eurographics’02/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,” in *Proc. of ACM SIGGRAPH’98*, pp. 343-352, 1998.

[14] P. Sander, S. Gortler, J. Snyder, and H. Hoppe, “Signalspecialized Parametrization,” in *Proc. of Eurographics Workshop on Rendering*, 2002.

[15] B. Guenter, C. Grim, D. Wood, H. Malvar, and F. Pighin, “Making Faces,” in *Proc. of ACM SIGGRAPH’98*, pp. 55-66, 1998.

[16] Y. Guo, J. Wang, H. Sun, X. Cui, and Q. Peng, “A novel constrained texture mapping method based on harmonic map,” *Comput. & Graph,* 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 & Graphics*, vol. 27, no. 3, pp. 415-422, 2003.

[18] H. J. Schek, “The force density method for form finding and computation of general networks,” *Computer Methods in Applied Mechanics and Engineering*, vol. 3, pp. 115–134, 1974.

[19] S. Yoshizawa, A. Belyaev, and H. P. Seidel, “A Fast and Simple Stretch-Minimizing Mesh Parameterization,” in *Proc. of Intel’ Conf. on Shape Modeling and Applications*, 2004.

[20] D. Piponi and G. Borshukov, “Seamless texture mapping of subdivision surfaces by model pelting and texture blending,” in *Proc. of Siggraph’00*, pp. 471-478, 2000.

[21] A. Sheffe and E. Sturler, “Parameterization of Faceted Surfaces for Meshing Using Angle Based Flattening,” *Engineering 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. on Graph*, vol. 24, no. 2, pp. 311-330, 2005.

[23] O. Sorkine and D. Cohen-Or, “Least-squares Meshes,” in P*roc. of Shape Modeling International 2004*, pp. 191-199, 2004.

Hongchuan Yu received his Ph.D. 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, UK. His research interests include Geometry modeling and rendering, face recognition and expression synthesis, and image processing.

Xiaosong Yang is a lecturer at the National Centre for Computer Animation, Bournemouth Media School, Bournemouth University, United Kingdom. He received his B.S. (1993) and M.S. (1996) degrees in Computer Science from the Zhejiang University (P. R. China), and his Ph.D. (2000) in Computing Mechanics from Dalian University of Technology (P. R. China). He worked as PostDoc (2000–2002) in the Department of Computer Science and Technology of Tsinghua University for two years, and as 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.

Jian J. Zhang is currently a Professor of Computer Graphics at the National Centre for Computer Animation, Bournemouth University, UK, 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 modelling, animation and simulation, including geometric modelling, rigging and skinning, motion synthesis, deformation and physics-based simulation.