Constrained Parameterization With Applications To Graphics And Image Processing

Thesis by

HONGCHUAN YU

in partial fulfillment of the requirements of Bournemouth University

for the degree of Doctor of Philosophy

National Centre for Computer Animation
Bournemouth University, Poole, UK
March 2012

COPYRIGHT STATEMENT

ACKNOWLEDGEMENTS

There are many people whose guidance and support made my research possible. First, I thank my supervisor, Prof Jian Jun Zhang, for his comprehensiveness and generosity. With his kind guidance, I have worked in a relaxed and motivating academic environment which has shaped me as a researcher and as a scientist. For these experiences I am forever grateful.

I am also deeply indebted to my second supervisor, Dr Hammadi Nait-Charif, whose encouragement and enthusiasm not only elevated this work but have also profoundly affected me personally and philosophically. I have been extremely fortunate to work with many exceptionally talented and gifted researchers over the years. Dr Lihua You, Dr. Xiaosong Yang, Dr Jian Chang, and Dr Richard Southern have been a pleasure to work with and have enriched my experiences both in research and personally.

Finally, and most importantly, I thank my family, my wife, my lovely daughter, my parents, my brother and sister. Your love and encouragement kept me sane and grounded in this wonderful chapter of my life.

ABSTRACT

Surface parameterization is to establish a transformation that maps the points on a surface to a specified parametric domain. It has been widely applied to computer graphics and image processing fields. The challenging issue is that the usual positional constraints always result in triangle flipping in parameterizations (also called foldovers). Additionally, distortion is inevitable in parameterizations. Thus the rigid constraint is always taken into account. In general, the constraints are application-dependent. This thesis thus focuses on the various constraints depended on applications and investigates the foldover-free constrained parameterization approaches individually. Such constraints usually include, simple positional constraints, tradeoff of positional constraints and rigid constraint, and rigid constraint. From the perspective of applications, we aim at the foldover-free parameterization methods with positional constraints, the as-rigid-as-possible parameterization with positional constraints, and the well-shaped well-spaced pre-processing procedure for low-distortion parameterizations in this thesis.

The first contribution of this thesis is the development of a RBF-based re-parameterization algorithm for the application of the foldover-free constrained texture mapping. The basic idea is to split the usual parameterization procedure into two steps, 2D parameterization with the constraints of convex boundaries and 2D re-parameterization with the interior positional constraints. Moreover, we further extend the 2D re-parameterization approach with the interior positional constraints to high dimensional datasets, such as, volume data and polyhedrons.

The second contribution is the development of a vector field based deformation algorithm for 2D mesh deformation and image warping. Many presented deformation approaches are used to employ the basis functions (including our proposed RBF-based re-parameterization algorithm here). The main problem is that such algorithms have infinite support, that is, any local deformation always leads to small changes over the whole domain. Our presented vector field based algorithm can effectively carry on the local deformation while reducing distortion as much as possible.

The third contribution is the development of a pre-processing for surface parameterization. Except the developable surfaces, the current parameterization approaches inevitably incur large distortion. To reduce distortion, we proposed a pre-processing procedure in this thesis, including mesh partition and mesh smoothing. As a result, the resulting meshes are partitioned into a set of small patches with rectangle-like boundaries. Moreover, they are well-shaped and well-spaced. This pre-processing procedure can evidently improve the quality of meshes for low-distortion parameterizations.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

This thesis addresses a classic topic of differential geometry, surface parameterization, with applications to the texture mapping, mesh skinning and image warping. Surface parameterization can be viewed as a bijective mapping between a surface and a suitable parameter domain. In general, the parameter domain itself is a surface and parameterization means computing a both one-to-one and onto mapping in between the parameter domain and the embedded surface. Typically, a sphere is projected onto a plane domain using a system of gridline, i.e. longitude and latitude. In general, a surface is represented or approximated by a triangular mesh and hence the problem of computing such a mapping is referred to as mesh parameterization. The mapping between the meshes and the parameter domains are piecewise linear.

Mesh parameterizations between surface meshes and a variety of parameter domains have many applications in computer graphics, geometry processing and image processing, such as image registration, scattered data fitting and geometry modeling. In recent works, many methods have been presented for satisfying diverse parameter domains and maintaining different parameterization properties. However, the challenging issues from differential geometry of surface theory and numerical computation are becoming desperate for being solved, for example, hard positional constraints, foldover or triangle flips, isometric (namely zero distortion), and conformal maps.

The main motivation of this thesis is to look for the mathematic essence underlying the distortion challenge, e.g. triangle foldovers. This is because the foldover is not only reported to appear in many applications of computer graphics, e.g. texture mapping [Kraevoy et al. 2003, Eckstein et al. 2001, Levy 2001, and Sander et al. 2001], remeshing [Dong et al. 2006, Gu et al. 2002, and Guskov et al. 2000, and Lee et al. 2000], mesh editing [Alexa et al. 2000, James et al. 2005, Lewis et al. 2000, and Sorkine et al. 2004], but also is reported as differential issue in image registration applications [Dong et al. 2009, Wang et al. 2008, Wang et al. 2010, and Schaefer et al. 2006] as well. Particularly, due to the quickly developing 3D scanning technology and the resulting demand for efficient scattered data fitting, parameterization or further compression methods of increasingly complex triangulations have been the active research issues in recent years. Usually, such applications request some specified positional constraints or area and shape-preserving properties, which inevitably leads to the triangle foldover in parameter domains. This thesis focuses on the specified constraint conditions of

parameterizations, since the constraints are always application-dependent and appear in applications of both computer graphics and image processing.

1.2 CHALLENGES IN SURFACE PARAMETERIZATION

Surface parameterizations almost always cause distortion in either angles or areas. Seeking an allowable mapping (i.e. to achieve some distortion tolerance) is an active topic of differential geometry. If the mapping is length-preserving, we call such a mapping as an isometry. This is an ideal mapping, since it preserves just about everything we request, such as areas, angles and lengths. If we only ask for angle-preservation, the mapping is called as conformal mapping. If the mapping is only area-preserving, we call it as equiareal mapping. To find a well-behaved mapping, the constraints of area-preservation and angle-preservation are usually taken into account together. Furthermore, since all the parameterization methods are used to approximate the underlying smooth surface in a discrete setting (e.g. a triangular mesh), we may encounter a serious distortion, i.e. some triangles flip over or foldover. Additionally, in various image and graphics applications, parameterizations are usually requested to satisfy some specified positional constraints. This further aggravates mesh distortion, i.e. foldovers spread over the whole domain.

The following outlines the challenging issues in different constrained surface parameterization applications.

(1) Parameterization with the fixed boundaries. This kind of constrained parameterization methods is categorized into the discrete harmonic maps. In general, harmonic maps are guaranteed to be one-to-one for convex regions [Tutte 1963].

However, the convexity constraint may yield big distortions near the boundary. The challenging issue from such applications is how to choose a suitable boundary mapping such that the constrained parameterization incurs the minimum distortion. This implies that we choose the suitable boundaries not only on the parameter domain, but also on the original surface.

(2) Parameterization without the fixed positional constraints. This kind of constrained parameterization methods is categorized into the discrete conformal mappings. Usually, they maximize the conformality (i.e. angle preservation) of the piecewise linear mapping without the fixed positional constraints. Instead, the shapes of the parameter domains are determined by the methods themselves. The challenging issue from such applications is that a number of constraints have to be taken into account to guarantee the validity of the solution. In general, solving a large nonlinear system is not simple task.

The above mentioned parameterization applications indeed pursue the isometric mapping. Thus, the constraint of the minimum distortion can be referred to rigid constraint. To distinguish from the rigid constraint, we will refer to the internal constraint points as the positional constraints in this thesis.

(3) Parameterization with the internal positional constraints. Unlike the rigid constraint, these positional constraints are used to establish maps with feature correspondence between objects. The challenging issue is two-fold, 1) parameterization can accommodate the specified positional constraints; 2) parameterization incurs distortions as small as possible.

1.3 DISSERTATION CONTRIBUTIONS

This thesis is concerned with the surface constrained re-parameterization, and further investigates the parameterizations (1) and (3) mentioned in Section 1.2. Because the constraints are application-dependent, this thesis aims at the following three kinds of constraints in terms of applications, that is, positional constraints, tradeoff of positional constraints and rigid constraint, and rigid constraint. The encountered challenges include, the triangle foldovers, the suitable boundary mapping and minimum distortion. Our contributions are shown as follows.

1.3.1 SIMPLE POSITIONAL CONSTRAINTS

We focus on the application of foldover-free texture mapping with the specified positional constraints. Texture mapping needs to embed a 3D surface into a 2D domain with some positional constraints. Usually, it is concerned with the positional constraints instead of the caused distortions. The specified positional constraints always result in large deformation in such scenarios. We present the foldover-free reparameterization method based on radial basis functions (RBF). The major contributions are as follows:

- To the best of our knowledge, the proposed method is the first RBF-based approach that ensures user-specified constraints are satisfied and that foldovers are avoided. An explicit mathematical condition guarantees that no mesh foldover is generated during the RBF-based reparameterization. This is called the *foldover-free condition*;
- The RBF-based method is a mesh-free approach. Thus, generating smooth

texture mapping is possible without an extra computationally expensive smoothing optimization, as required in [Kraevoy et al. 2003 and Lee et al. 2008];

- To the best of our knowledge, the proposed method presents the first implementation of satisfying positional constraints without predefined fixed boundaries. Furthermore, the proposed method can handle models with interior boundaries without additional treatment, such as cutting the model into several pieces, as required in [Lee et al. 2008].

The related work has been published in,

- Yu, Hongchuan, Lee, TongYee, et al., 2012, RBF-Based Reparameterization Method for Constrained Texture Mapping, IEEE Trans. on Visualization and Computer Graphics, Vol.18, Issue 7.
- Yu, Hongchuan, Zhang, Jian J. and Lee, Tong-Yee, Foldover-free High Dimensional Deformation, The Visual Computer Journal, (to appear).

1.3.2 TRADEOFF OF POSITIONAL CONSTRAINTS AND RIGID CONSTRAINT

This indeed involves two kinds of constraints, rigid constraint and positional constraints. The former refers to the distortion caused by deformation as small as possible. Image warping and registration usually expect reduce distortion caused by deformation as much as possible. For example, the bones should remain rigid in image registration rather than any bending. The latter refers to the usual positional constraints. We focus on the application of 2D shape and image deformation with such constraints. A segment or patch of an image has been selected. The user deforms this segment/patch by moving a set of constraint points within the segment. We hope to compute a natural deformation of this segment to align the specified positions of the constraint points. Simultaneously, we expect that such deformation

causes a fairly minor distortion as well, that is, deformation is close to isometric. The challenging issue is to seek a tradeoff between the positional constraints and as-rigid-as possible deformation. We presented a topology preserved shape deformation algorithm accordingly. The major contributions include,

- Consistency of deformation fields. Unlike the previous approaches, we do not directly solve 2D shape and image deformation as a global optimal problem with constraints, but first convert the deformation into vector fields taking into account the specified positional constraints. Then, the deformation field is further reconstructed from a feasible subspace maintaining vector field consistency. This can effectively overcome the foldover challenge.

- As-Rigid-As-Possible Deformation. We incorporate with the consideration of as As-Rigid-As-Possible deformation in our implementation. This can effectively reduce distortion caused by deformation.

The related work has been published in,

- Yu, Hongchuan and Zhang, Jian J., 2012, Topology preserved shape deformation, The Visual Computer Journal, Vol.28, No.6-8, pp.849-858.

1.3.3 WELL-SHAPED AND WELL-SPACED MESH

This essentially requests a suitable boundary mapping to reduce distortion caused by parameterizations. The emphasis is the rigid constraint instead of the internal positional constraints. The rigid constraint is usually requested by various mesh editing systems. Consider such a map between two surfaces that allows transferring the details from one model to another, or interpolating the shape and appearance of

the two models. In this application, overlapping the two 2D parameterizations for matching is the foundation. Unfortunately, the current 2D parameterization approaches inevitably incur large distortion except the developable surfaces. Recent works [Gu et al. 2002, Guskov et al. 2000, and Lee et al. 2000] indicate that the small mesh with rectangle, triangle or circle-like and smooth boundaries is easy to flatten. Moreover, the well-spaced and well-shaped meshes distribute samples more uniformly over the surface and hence better capture surface features. In other words, the meshes need to be pre-processed for low-distortion parameterizations. Following this issue, we develop two pre-processing procedures for that we would like in our application of skin sliding. The main contributions of our work include,

- Partitioning a whole mesh into a set of patches. Moreover, the boundaries of the patches are becoming smooth by a further refinement procedure;
- Mesh smoothing. We present a new approach to smooth meshes while preserving their intrinsic features. The resulting meshes remain well-shaped and well-spaced.

The related work has been submitted to,

- Yu, Hongchuan, Yang, Xiaosong and Zhang, Jian J., Laplacian Editing For Skin Sliding, submitted to Computer Graphics Forum Journal.

1.4 THESIS OVERVIEW

The rest of this thesis is organized in the following six chapters:

- Chapter 2 gives a detailed literature review of the related works.

- Chapter 3 addresses the issue of the positional constraints in surface parameterization. In this chapter, this foldover problem is addressed by developing the radial basis function (RBF)-based re-parameterization. Given initial 2D embedding of a 3D surface, the proposed method can re-parameterize 2D embedding into a foldover-free 2D mesh, satisfying a set of user-specified constraint points. In addition, this approach is mesh-free. Therefore, generating smooth texture mapping results is possible without extra smoothing optimization.

- Chapter 4 further addresses the generalization of the proposed RBF-based re-parameterization approach to high dimensional datasets rather than 2D parametric domain. We further apply the generalized approach to 3D brain volume dataset deformation and 3D polyhedral deformation.

- Chapter 5 still addresses the positional constraints in 2D parameterization. Unlike Chapter 3 and 4, there is an additional constraint to be taken into account here, that is, rigid constraint. The main problem in the proposed RBF-based re-parameterization method and its extension in Chapter 3 and 4 is that they have infinite support. This implies that any local deformation could result in small distortions over the whole image domain. Our 2D shape and image deformation application requests the resulting distortion as small as possible. Thus, this is indeed to seek a tradeoff of the positional constraints and as-rigid-as-possible deformation.

- Chapter 6 addresses the pre-processing for 2D low-distortion parameterizations, that is, mesh partition and smoothing. This is because well-spaced and well-shaped meshes suffer small distortion in 2D parameterization. The application is the skin sliding that simulates the skin moving over underlying layers of fat, muscle and bone. Skin sliding, as a secondary animation technique, brings about extra realism to character animation. The physically based skin sliding

approaches usually suffer from inherent computational complications. To avoid this numerical challenge, some interpolation techniques are applied to the implementation of skin sliding. However, there are many deficiencies in practice, e.g. missing out features of skin surface and smoothness issue etc., which greatly reduce a realistic appearance. Essentially, skin sliding assumes that the shape can be preserved and the features of skin surface can be transferred to the target mesh. We focus on these two aspects and further reformulate the implementation of skin sliding based on the graph Laplacian framework, which helps our proposed algorithm to implement the mesh partition, shape and feature preservation. The elements worth mentioning include the mesh partition and mesh smoothing. The former is a new application of the active research issue: mesh segmentation. The novelty of our method is to automatically extract the skinning regions and further smooth the partition boundaries. The latter is a new method for mesh smoothing.

- Chapter 7 summarizes our research work presented in this thesis, and further gives out our future works.

CHAPTER 2

LITERATURE REVIEW

Surface parameterization was first introduced into computer graphics as a method of texture mapping [Bennis et al. 1991], which has been becoming a vital tool for many graphics applications in the last decade. Parameterization not only refers to mapping a surface into a plane domain, but it is also requested to map the surface into some 3D simplex, such as spheres. Floater et al. [Floater and Hormann 2005] and Sheffer et al. [Sheffer et al. 2006] have given the detailed surveys on this topic respectively. From the viewpoint of users, this chapter surveys the applications which benefit from surface parameterization.

2.1 TEXTURE MAPS

Texture map is usually expected to enrich the appearance of a model in a static image. The challenging problem is computing texture coordinates to satisfy user-specified correspondence between the 3D model and texture image. However, this has not been given much consideration in literature. Few studies have been conducted on meeting soft constraints [Levy and Mallet 1998, Cabral et al. 2009] (i.e., to satisfy the positional constraints approximately). Levy [Levy 2001] and Desbrun et al. [Desbrun et al. 2002] proposed a least-squares system and Lagrange multipliers as solutions, respectively. The basic idea can be simply described as follows,

$$E(T) = \sum_{j=1}^{m} \left(X_j - T(U_j)\right)^2 + \lambda \iint_{\Omega} \left(\left(\frac{\partial^2 T}{\partial u^2}\right)^2 + \left(\frac{\partial^2 T}{\partial v^2}\right)^2 \right) du dv , \qquad (2.1)$$

where T is a function representing a surface, X_j denotes a set of m constraint points that are passed through by the surface $T(U)$, and these constraint points are associated with the parameter-space points U_j. Minimizing the above functional yields a suitable mapping T from the parameter space to the surface. The first term represents the squared deviation at the constraint points and the second term enforces the smoothness of the solution. The resulting solution may be a compromise between the accuracy of the fitting and the smoothness of the solution. The given constraints are therefore called as “soft constraints”. However, these two methods fail to guarantee a bijective embedding. Zhang et al. [Zhang et al. 2005] focused on a special case (i.e., deforming a patch by stretching its boundary). Occurrence of foldovers when internal positional constraints are added in the original patch, and whether these can converge to the expected positions, were not clearly stated.

In contrast, hard constraints were studied in [Kraevoy et al. 2003 and Eckstein et al. 2001] because a perfect texture alignment is essential at certain delicate areas of a

mesh. Eckstein et al. [Eckstein et al. 2001] proposed a constrained simplification to align constraints, adding Steiner vertices to avoid foldovers. Theoretically, although the above method can handle large sets of constraints, it is extremely complicated and not very robust [Kraevoy et al. 2003]. In addition, only simple examples were shown in [Eckstein et al. 2001]. Thus, whether the above method can handle more complicated constraints is not clear. Kraevoy et al. [Kraevoy et al. 2003] and Lee et al. [Lee et al. 2008] performed embedding by adding a fixed rectangular virtual boundary, after which the Delaunay method was applied to triangulate the region between true and virtual boundaries. After aligning user-specified hard constraints, the embedding is usually highly distorted. Therefore, a post-smoothing procedure is required to reduce the distortion, adding to computation costs. To avoid triangle foldovers, swapping edge is applied here. Figure 2-1 illustrates the idea of swapping edge [Lee et al. 2008]. When the vertex *v* moves to v_α as shown in Fig 2-1a, the shaded triangles will be degenerated. Firstly, we can swap the edge $\overline{vq_4}$ with $\overline{q_3q_5}$ to the triangle Δvq_3q_4. Then, we swap the edge $\overline{vq_3}$ with $\overline{q_2q_5}$ to remove the triangle Δvq_2q_3. After that, swapping the edge $\overline{vq_7}$ with $\overline{q_6q_8}$ to remove the triangle Δvq_7q_8, all of the potential folodover triangles are removed.

Kraevoy et al. [Kraevoy et al. 2003] failed to completely remove foldovers because the consistent neighboring ordering was not considered in finding matching triangulations. Fujimura et al. [Fujimura and Makarov 1998] presented an image-warping method. To satisfy positional constraints, the Delaunay triangulation and edge swaps were repeatedly used in their work to avoid foldovers. However, edge swaps can damage the geometric surface when used to texture map a 3D mesh, as discussed in [Eckstein et al. 2001].

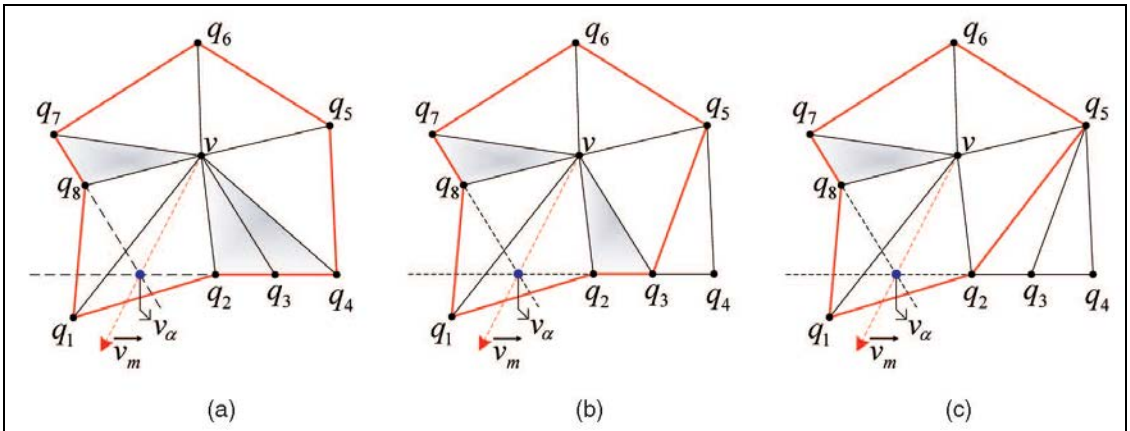


Figure 2-1. Illustration of swapping edge.

Tang et al. [Tang et al. 2003] and Lee et al. [Lee and Huang 2003] proposed an RBF-based parameterization method. However, neither method mentioned the foldover challenge. Tiddeman et al. [Tiddeman et al. 2001] applied the condition of positive Jacobian determinant to remove foldovers in their image warping application. This condition is well known in differential geometry to ensure one-to-one mapping [Meisters and Olech 1963]. The method starts from an initial dense mapping that is likely to contain foldovers. Foldovers are then removed by iteratively scaling the given mapping. However, dense mapping is difficult to establish beforehand. Moreover, the primary deficiency of this method is that the convergence cannot be guaranteed. In a given discrete setting, scaling a given dense mapping usually results in iterative step-length towards zero quickly, as admitted by the authors. In a few extreme cases, the method cannot satisfy the specified positional constraints.

2.2 TEXTURE SYNTHESIS

Texture synthesis refers to creating texture over an arbitrary surface mesh using a given 2D texture element, which is usually in the form of an image or a patch. For simplicity, we can synthesize the texture directly on a 2D parameterization in terms of the given exemplar texture instead of overlapping with a whole texture image. To distinguish from solid textures, we use the term of surface texture to describe such geometry-influenced textures here. The type of texture can be quite varied, e.g. the natural examples of surface texture including the pattern of bark on a tree, spots and stripes of horses, fishes and birds etc., and the patterns of flowers and trees on a hillside.

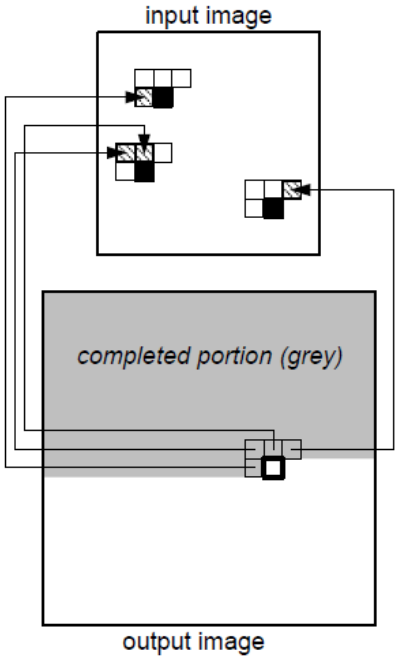


Figure 2-2. Illustration of texture synthesis.

The current techniques for texture synthesis on surfaces can be roughly categorized into two groups. The first group [Gorla et al. 2001, Tong et al. 2002, Zhang et al. 2003, and Ashikhmin 2001] is based on per-pixel nonparametric sampling. [Heeger et al. 1995] utilizes the image pyramid strategy to capture the statistical properties of the exemplar image at different levels of resolution.

Initializing the synthesized texture by random noise, each level of the synthesized pyramid will converge to the appearance of the corresponding exemplar image pyramid by using Histogram matching operator iteratively. However, this approach usually works well on stochastic textures while is not suitable for the structured exemplar texture. The relationship between pixel color and its spatial neighborhood is another research focus of this group. In [Efros et al. 1999], each pixel of the exemplar texture is first modeled by its neighborhood kernel. Then, each pixel in the target image is substituted one by one with the one that has the most similar neighbor in the exemplar texture. [Wei and Levoy 2000] further extends this method in the following aspects, implementing multi-resolution by using Gaussian pyramids, and modifying the original neighborhood matching to the order-dependent matching. The basic idea is illustrated in Figure 2-2. Pixels are generated in scanline order. Thus, each pixel maintains their individual L-shaped neighborhood. The value of a pixel is determined by choosing the best pixel in the input image. Best pixel is the one whose L-shaped neighborhood most closely resembles the neighborhood of the pixel currently being synthesized in the output image plane.

The disadvantage of this group is that per-pixel sampling is susceptible too much. This is because the used L_2-norm is poor to perceptual similarity. As a result, it is difficult for the algorithms in this group to maintain texture patterns with certain types [Zhang et al. 2003, Ashikhmin 2001].

The second group refers to the patch based methods [Efros and Freeman2001, Praun et al. 2000]. Roughly speaking, this kind of algorithms is namely to copy and paste image patches repeatedly to fill the target image. [Efros and Freeman 2001] makes use of the overlap region between adjacent patches to appropriately quilt them, so that these patches all fit together. The patch is selected from a set of candidates

according to an overlap error minimization. To speed up this algorithm, [Kwatra et al. 2003] solves the error minimization problem using a graph-cut algorithm. Moreover, [Kwatra et al. 2005] presents a global optimization framework to synthesize a new texture. It essentially minimizes an energy function that considers all the pixels together. The energy function measures the similarity to the exemplar texture. To hide patch seams and enhance the smoothness across the seams, this kind of algorithms also performs graphcut on surfaces.

Additionally, users always interactively select the exemplar texture for the different regions of a surface separately in practice, i.e. cutting a patch for texture synthesis and then pasting it back. The potential challenges include, surface partition, mesh remeshing and seamless embedding. This essentially requires a pre-processing on the input meshes before the following parameterization procedure.

2.3 IMAGE WARPING/DEFORMATION

We first mention the medical imaging, since the use of medical imaging has been rapidly increasing in recent years. Image registration is indeed one of the kernel techniques in medical imaging. Medical image data usually refers to computed tomography (CT), ultrasound (US), magnetic resonance imaging (MRI), positron emission tomography (PET) and single photon emission computed tomography (SPECT). For visualization and pathology analysis purposes, such image datasets are usually required to align to a specified template or vice versa, i.e. to establish a bijective mapping between images. Such mapping is called as image registration. The ideal registration method should include local as well as global deformation, that

is, fully affine and elastic deformation. The usual constraints are the positional constraints (also called as landmarks).

Roughly speaking, image registration can be classified into two categories, PDEs based approaches and parametric model based ones. In the first category, image registration is usually found by solving a nonlinear PDE system. For example, brain tissues are regarded as a linear elastic or viscous fluid that is subject to a constrained deformation [Karacali and Davatzikos 2004, Haber and Modersitzki 2007, Beg et al. 2005]. The basic idea is namely to find a smooth transformation *T* by minimizing the following functional,

$$E(T) = D(I_0, I_1(T(X))) + \alpha S(T), \quad X \in R^d, d = 2 \text{ or } 3, \qquad (2.2)$$

where I_0 and I_1 are the given images and α is a regularization parameter and compromises between similarity and regularity. The functional *D* measures the distance between I_0 and I_1 in terms of the sum of squares difference, i.e.,

$$D(I_0, I_1(T)) = \left\| I_0 - I_1(T) \right\|^2. \qquad (2.3)$$

The regularization term is usually expected to yield a smooth and unique transformation *T*. It can be implemented by the elastic regularization as below,

$$S(T) = \beta_1 \sum_j \left\| \nabla T^j \right\|^2 + \beta_2 \left\| \nabla \cdot T \right\|^2, \qquad (2.4)$$

where β_1 and β_2 are the so-called Lame-constants. Moreover, for the large deformation scenarios, it is hard to prevent foldovers based on the above-mentioned functional. To deal with this challenge, Jacobian constraint is requested in many algorithms, that is,

$$0 < C(T) = \det\left(I_d + \nabla T\right) < \infty, \quad \text{for all } X, \qquad (2.5)$$

where I_d denotes an identity Matrix. As a result, the functional is rewritten as,

$$E(T) = D\big(I_0, I_1(T)\big) + \alpha S(T) + \gamma \left\| C(T) - 1 \right\|^2 . \qquad (2.6)$$

The second category is to model the registration by a parametric function with a set of the undetermined parameters. Usually the mapping is known continuously and the resolution of the mapping can be controlled independently of the image resolution. [Kybic and Unser 2003] introduced B-splines to model the image registration. As the usual PDEs based registration, they added a penalization term as the regularization term. [Sorzano et al. 2005] further evaluated the efficiency of [Kybic and Unser 2003] and showed that B-splines were a good alternative compared to other parameterization, such as wavelet or Fourier representation. In [Rohlfing et al. 2003], the penalization term is defined as the absolute value of the log of the Jacobian of the deformation or the square of the second derivative for each voxel. They used finite differences to compute the gradient of the Jacobian term. Due to computational complexity, [Musse et al. 2001] suggested a block nonlinear Gauss-Seidel algorithm to minimize the energy between the images with the constraint that the Jacobian was continuously positive. We hope to point out that the constraints on the Jacobian are also the foldover-free constraint described in this thesis.

Then, we summarize the some recent image warping applications as well. Image resizing is used to stretch-and-squeeze image so as to fit different display devices. The salient image regions should remain unchanged or have a minimal distortion. Video retargeting further extends the image resizing techniques to a video display. Like the other image warping or morphing applications [Fujimura and Makarov 1998, Tiddeman et al. 2001, Weber et al. 2009], the challenges include pixel overlap,

jagged and blurred edges. Many current interpolation techniques, e.g. bilinear and bicubic interpolation, can effectively amend the issue of edge smoothness. However, to the best of my knowledge, pixel overlap (i.e. non-bijective warping) is still an unsolved problem. I therefore focus on this challenge in this thesis. Additionally, image or video frame are usually viewed as the quadrangle meshes that are initially a grid of axis-aligned squares. Pixel overlap can be viewed as triangle/quad flipping. It is called the triangle/mesh foldover. Compared to triangle meshes, it is hard to formulate this challenge by using quadrangle meshes.

2.4 MESH EDITING

Shape deformation or interpolation is an important research topic in Computer graphics. A number of current shape deformation approaches focus on the As-Rigid-As-Possible techniques [Igarashi et al. 2005] that computes a natural shape deformation. This idea is further extended to 3D mesh deformation [Alexa et al. 2000] and 3D mesh parameterization [Sorkine and Alexa 2007, Liu et al. 2008]. It is interesting that they both adopted an alternating least squares scheme (i.e. known as local/global algorithm) to approximate the As-Rigid-As-Possible deformation. In the local step, to measure the rigidity of a deformed mesh, we have the following functional,

$$E(S') = \sum_{i=1}^{n} w_i \sum_{j \in N(i)} w_{ij} \left\| \left(p'_i - p'_j\right) - R_i\left(p_i - p_j\right) \right\|^2 , \qquad (2.7)$$

where w_i and w_{ij} are some fixed cell and edge weights, S' denotes a deformation of the mesh S and R_i denotes a rotation of some cell. For one cell, the optimal rotation

R_i bring about rigid deformation instead of any non-rigid ones. Within the neighborhood of some vertex, we have the following linear system by minimizing the above functional,

$$\sum_{j\in N(i)} w_{ij}\left(p_i' - p_j'\right) = \sum_{j\in N(i)} \frac{w_{ij}}{2}\left(R_i + R_j\right)\left(p_i - p_j\right). \qquad (2.8)$$

In the global step, applying the discrete Laplace-Beltrami here yields the following linear system,

$$L\mathbf{p}' = \mathbf{b}, \qquad (2.9)$$

where vector $\mathbf{p}'$ contains the unknown coordinates of the vertices, and the constraint vertices can be concatenated to the parameter matrix L and vector b. This local/global scheme can be applied to 2D meshes as well [Karni et al. 2009]. Other approaches, such as [Weng et al. 2005], cast deformation as an energy minimization problem. Detail-preserving and positional constraints are added into the energy functional as constraint terms. The distinct advantage of the former is to easily add the constraints to each triangle by solving a local optimization problem. However, regardless of global optimization or local/global schemes, foldovers of the underlying mesh have not been overcome yet. [Karni et al. 2009] suggested to explicitly restricting the local step to avoid it, while [Wang et al. 2008] proposed to prevent it in a heuristic manner. For 3D scenarios, foldovers usually take place at skinny triangles. Numerical instability caused by skinny triangles is essentially from triangle reflection.

Moreover, mesh editing benefits from the local parameterization between pairs of mesh patches as well. Usually, one can locally parameterize the regions of interest (ROI) on the two models in a 2D domain and then overlap the 2D parameterizations.

The cut-and-paste transfer presented in [Biermann et al. 2002] can effectively transfer the details between models using local parameterization. [Sorkine et al. 2004, Levy 2003] also used the local parameterization for mesh composition in a similar manner. They first overlapped the 2D parameterizations of ROIs to yield a mapping and then made use of it for extracting and smoothly blending shape from the two models. In such applications, there are no any positional constraints in 2D parameterizations except the fixed boundaries. However, for more precise controls, it is natural to add some internal positional constraints. This will lead to the challenging issue of triangle foldovers again.

2.5 MESH COMPLETION AND COMPRESSION

Triangulation on range data usually result in a mesh containing holes and multiple components. [Levy 2003] extracted the hole boundaries using 2D parameterization and triangulated those. To a set of patches, we have to build a model by registration. In many cases, there exists prior knowledge on the overall shape of the scanned model. For example, for human scan, a generic human shape model is readily available. [Allen et al. 2003, Anguelov et al. 2005] respectively used this prior model to facilitate completion of scans. They calculated a mapping between the scan and the prior human model. Moreover, [Kraevoy and Sheffer 2005] presented a general and robust template based approach for completion of any type of scans. They fixed the boundary of a group of base mesh faces, updated the barycentric coordinates in the interior, and then possibly re-assigned some vertices to different faces inside the group.

While keeping the mesh completion, mesh compression is used to compactly store geometry models [Alliez and Gotsman 2003]. Compression rate is inversely proportional to the data entropy. When meshes are regular in both topology and geometry, the compression rates are higher. Topological regularity usually refers to meshes where almost all vertices have the same degree. Geometry regularity refers to meshes where almost all the triangles are similar to each other in terms of shape and size, and vertices are close to the centroid of their neighbors [Gu et al. 2002, Hoppe and Praun 2005]. Such meshes can be obtained by parameterizing the original mesh and remeshing them [Guskov et al. 2000, Khodakovsky et al. 2003].

CHAPTER 3

RBF-BASED RE-PARAMETERIZATION METHOD FOR CONSTRAINED TEXTURE MAPPING

3.1 PROBLEM FORMULATION

3.1.1 BASIC IDEA AND MOTIVATION

This chapter addresses the surface re-parameterization with the positional constraints. The challenging is to overcome the triangle foldovers in mesh parameterizations. The overview of the proposed algorithm is as follows. An input 3D surface is first embedded into a 2D convex domain with harmonic mapping [Guo et al. 2005]. A mathematical foldover-free condition (see Section 3.2) is derived, and incorporated into an RBF-based reparameterization algorithm. The algorithm then iteratively aligns user-specified positional constraints. The main idea is to first estimate the iterative step length (i.e., scaling factor) subject to the foldover-free

condition, and then to successively approximate the desired positions through RBF-based deformation. In short, RBF is used to iteratively deform the 2D mesh to align user-specified constraints. With the foldover-free condition at each iterative step, the deformation is prevented from being over-aggressive (i.e., to induce foldovers). Due to more mathematic computations, we first describe the proposed RBF-based re-parameterization method briefly, and then separately address the each step in details in the following sections.

3.1.2 ITERATIVE RBF-BASED REPARAMETERIZATION PROCEDURE

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

$$T: \begin{cases} X = (x, y)^T \in S \rightarrow U(X) = (u(X), v(X))^T \in \Omega \\ \text{subject to } U(X_i^*) = U_i^*, i = 1, ..., m \end{cases}. \tag{3.1}$$

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

$$\Delta U = P(X) + \sum_{i}^{m} \boldsymbol{\lambda}_i \phi(\|X - C_i\|) \qquad (3.2)$$

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

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

Loop:
(1) Estimate the scaling factor δ by Eq.3.14 (see Section 3.2.2) based on the configuration of the current m constraint points $C_i^{(j)}$;
(2) Compute the current m constraint point displacements by $\Delta C_i^{(j+1)} = \delta\left(C_i^* - C_i^{(j)}\right)$, such that $C_i^{(j+1)} = C_i^j + \Delta C_i^{(j+1)}$;
(3) If $\delta < \delta_{threshold}$, apply triangle subdivision (see Section 3.3) then go to Step (1); otherwise,
(4) Compute the displacements of the points on $S^{(j)}$ by Eq.3.7 (see Section 3.2.2) based on all m updated $C_i^{(j+1)}$ and updating $S^{(j)} \rightarrow S^{(j+1)}$;
Repeat until $C_i^{(j)} = C_i^*$.

In this procedure, C_i^* denotes the desired positions. Initially, if C_i^* is applied to directly deform S in Equation 3.2, the result is usually too aggressive and foldovers may occur. Therefore, in Step (1), a conservative scaling factor δ needs to be computed and used to ensure that $C_i^{(j+1)}$ is not over-aggressive. Note that the iterative RBF-based reparameterization procedure can definitely change mesh S to a

foldover-free state. However, the final positions may not align exactly with the constraints C_i^* in some extreme scenarios. This implies that the method is only able to handle soft constraints. To alleviate this problem (i.e., to approximate hard constraints as much as possible), the mesh in Step (3) is subdivided by adding extra Steiner vertices. For more details, see Section 3.3.

3.2 ITERATIVE MECHANISM

3.2.1 FOLDOVER-FREE CONDITION

From a mathematical perspective, a "foldover-free" parameterization yields a "one-to-one" mapping between corresponding surfaces (or meshes) and parametric domains. In the present work, the initial 2D embedding of a 3D surface is given in advance. Focus is given on deforming this initial embedding with a set of internal constraint point pairs. This requires that the mapping T is globally univalent or "globally one-to-one" (i.e., the topology or the relationship between any pair of vertices in the mesh should remain unchanged before and after parameterization). Mathematically, this means the determinant of the Jacobian matrix must always be positive [Meisters and Olech 1963]:

$$\det(\nabla U) > 0 . \qquad (3.3)$$

According to the Gerschgorin circle theorem [26], a *sufficient* condition of satisfying Eq.3.3 can be described as follows:

$$\begin{cases} \dfrac{\partial u}{\partial x} > \left|\dfrac{\partial u}{\partial y}\right| \\ \dfrac{\partial v}{\partial y} > \left|\dfrac{\partial v}{\partial x}\right| \end{cases}. \qquad (3.4)$$

This is usually called as the foldover-free condition. The geometric meaning of Eq.3.4 is simply that the two vectors $\partial u/\partial(x,y), \partial v/\partial(x,y)$ are linearly independent of each other; thus, their included angle is less than π. The former is easy to understand. The latter implies that the right-hand rule in vector calculus is satisfied over the entire domain. Holding $\det(\nabla U) < 0$ at any point would result in left-handedness instead of right-handedness. This change would cause mesh foldover.

3.2.2 ITERATIVE STEP-LENGTH ESTIMATION

Our reparameterization algorithm employs an iterative framework and the displacements of vertices are estimated considering the condition of Eq.3.4. Equation 3.2 must be rewritten to implement the procedure, such that the displacements of some points linearly depend on the constrained points. This implies that deformation the mesh is achieved by adjusting the displacements of the constrained points in an iterative manner. A further expectation is that foldovers will be avoided by controlling the displacement of the constrained points in each iteration.

The RBF coefficients $(\boldsymbol{\lambda}_u, \boldsymbol{\lambda}_v, \mathbf{a}, \mathbf{b})$ of Eq.3.2 are first computed, where $\mathbf{a} = (a_1, a_2, a_3)^T$ and $\mathbf{b} = (b_1, b_2, b_3)^T$. For a given set of constrained points and their displacements, this can be achieved by solving the following linear system:

$$K\begin{pmatrix}\boldsymbol{\lambda}_u\\ \mathbf{a}\end{pmatrix}=\begin{pmatrix}\Delta\mathbf{x}_c\\ 0\end{pmatrix} \text{ and } K\begin{pmatrix}\boldsymbol{\lambda}_v\\ \mathbf{b}\end{pmatrix}=\begin{pmatrix}\Delta\mathbf{y}_c\\ 0\end{pmatrix}, \tag{3.5}$$

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

$$\begin{pmatrix}\boldsymbol{\lambda}_u & \boldsymbol{\lambda}_v\\ \mathbf{a} & \mathbf{b}\end{pmatrix}=K^{-1}\begin{pmatrix}\Delta\mathbf{x}_c & \Delta\mathbf{y}_c\\ 0 & 0\end{pmatrix}. \tag{3.6}$$

(For a detailed RBF computation, refer to [Carr et al. 2001].)

Substituting $(\boldsymbol{\lambda}_u, \boldsymbol{\lambda}_v, \mathbf{a}, \mathbf{b})$ into Eq.3.2, the new expression is as follows:

$$\begin{cases}\Delta u=\mathbf{M}(X)K^{-1}\begin{pmatrix}\Delta\mathbf{x}_c\\ 0\end{pmatrix}\\ \Delta v=\mathbf{M}(X)K^{-1}\begin{pmatrix}\Delta\mathbf{y}_c\\ 0\end{pmatrix}\end{cases}. \tag{3.7}$$

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

Note that Equation 3.7 describes a linear system of the displacement of any X (i.e. $\Delta u, \Delta v$) and of the constraint points (i.e. $\Delta\mathbf{x}_c, \Delta\mathbf{y}_c$). Whether or not the resulting mesh satisfies the condition of Eq.3.4 should depend on the configuration of the current constraint points [i.e., $\mathbf{M}(X)$ and K^{-1}], rather than their displacements, $\Delta\mathbf{x}_c, \Delta\mathbf{y}_c$. Moreover, during iterations, $\mathbf{M}(X)$ and K^{-1} are unfixed and depend on the configuration of the current constraint points. Hence, $\mathbf{M}(X)$ and K^{-1} are given focus. The derivatives of $\partial(u,v)/\partial(x,y)$ are computed as follows:

$$\begin{cases} \dfrac{\partial u}{\partial x} = 1 + \mathbf{M}_x K^{-1} \begin{pmatrix} \Delta \mathbf{x}_c \\ 0 \end{pmatrix} \\ \dfrac{\partial u}{\partial y} = \mathbf{M}_y K^{-1} \begin{pmatrix} \Delta \mathbf{x}_c \\ 0 \end{pmatrix} \\ \dfrac{\partial v}{\partial x} = \mathbf{M}_x K^{-1} \begin{pmatrix} \Delta \mathbf{y}_c \\ 0 \end{pmatrix} \\ \dfrac{\partial v}{\partial y} = 1 + \mathbf{M}_y K^{-1} \begin{pmatrix} \Delta \mathbf{y}_c \\ 0 \end{pmatrix} \end{cases} \tag{3.8}$$

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

Substituting the above derivatives into Eq.3.4 yields

$$\begin{cases} 1 + \mathbf{M}_x K^{-1} \begin{pmatrix} \Delta \mathbf{x}_c \\ 0 \end{pmatrix} > \left| \mathbf{M}_y K^{-1} \begin{pmatrix} \Delta \mathbf{x}_c \\ 0 \end{pmatrix} \right| \\ 1 + \mathbf{M}_y K^{-1} \begin{pmatrix} \Delta \mathbf{y}_c \\ 0 \end{pmatrix} > \left| \mathbf{M}_x K^{-1} \begin{pmatrix} \Delta \mathbf{y}_c \\ 0 \end{pmatrix} \right| \end{cases}. \tag{3.9}$$

In general, the displacement vectors of the constraint points $\left(\Delta c_x^i, \Delta c_y^i\right)^T$ can be obtained by the differences of the current constraint points' coordinates and their individual targets' coordinates. To satisfy the above inequalities, we may limit the length of each displacement vector by scaling the vectors $\Delta \mathbf{x}_c, \Delta \mathbf{y}_c$ as follows,

$$\begin{cases} 1 + \mathbf{M}_x K^{-1} \begin{pmatrix} \Delta \mathbf{x}_c \\ 0 \end{pmatrix} \delta > \left| \mathbf{M}_y K^{-1} \begin{pmatrix} \Delta \mathbf{x}_c \\ 0 \end{pmatrix} \right| \delta \\ 1 + \mathbf{M}_y K^{-1} \begin{pmatrix} \Delta \mathbf{y}_c \\ 0 \end{pmatrix} \delta > \left| \mathbf{M}_x K^{-1} \begin{pmatrix} \Delta \mathbf{y}_c \\ 0 \end{pmatrix} \right| \delta \end{cases}, \tag{3.10}$$

where, δ denotes a scaling factor and δ>0.

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

$$\begin{cases} \Omega_1(\delta) = \left\{ (1+\alpha,\beta) : |\alpha| \le \left| \mathbf{M}_x K^{-1} \begin{pmatrix} \Delta\mathbf{x}_c \\ 0 \end{pmatrix} \right| \delta, |\beta| \le \left| \mathbf{M}_y K^{-1} \begin{pmatrix} \Delta\mathbf{x}_c \\ 0 \end{pmatrix} \right| \delta, 1+\alpha > |\beta| \right\} \\ \Omega_2(\delta) = \left\{ (\alpha,1+\beta) : |\alpha| \le \left| \mathbf{M}_x K^{-1} \begin{pmatrix} \Delta\mathbf{y}_c \\ 0 \end{pmatrix} \right| \delta, |\beta| \le \left| \mathbf{M}_y K^{-1} \begin{pmatrix} \Delta\mathbf{y}_c \\ 0 \end{pmatrix} \right| \delta, 1+\beta > |\alpha| \right\} \end{cases}$$

(3.11)

where (α,β) denotes a point in Ω. Figure 3-2 illustrates the regions $\Omega_1(\delta),\Omega_2(\delta)$. The scaling factor δ is not a constant, and depends on the displacements of the constraint points. Thus, the dashed line is used to highlight these undetermined boundaries.

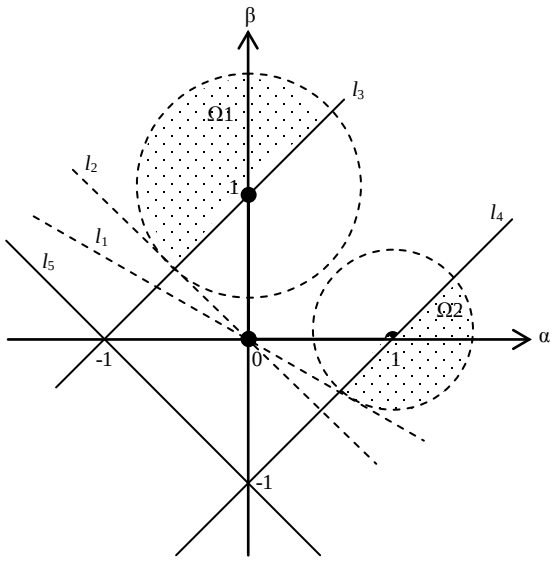


Figure 3-2. Illustration of the condition of Eq.3.4. The dashed lines denote the undetermined boundaries

The Eq.3.4 condition implies that vectors $\partial u/\partial(x,y), \partial v/\partial(x,y)$ should be linearly independent of each other. Figure 3-2 intuitively illustrates this concept by the five lines: l_1, l_2, l_3, l_4, l_5. For example, α and β should be above the line l_3 or under the ine l_4 and above the line l_5, so that the linear independence can be guaranteed. This can be achieved by the scaling factor in Eq.3.11. Moreover, for simplicity, assume that the vectors $\Delta\mathbf{x}_c, \Delta\mathbf{y}_c$ have the same distribution. The regions $\Omega_1(\delta),\Omega_2(\delta)$

would then have the same size in terms of Eq.3.11. This will lead to the overlap of straight lines l_1 and l_2 and form a new dividing line, which is α+β=0 in Fig.3-2. Line α+β=0 guarantees the included angle is less than π. Consequently, the condition of Eq.3.4 can be re-expressed as

$$\begin{cases} \left(\mathbf{M}_x K^{-1}\begin{pmatrix}\Delta\mathbf{x}_c\\0\end{pmatrix}\right)^2\delta^2+\left(\mathbf{M}_y K^{-1}\begin{pmatrix}\Delta\mathbf{x}_c\\0\end{pmatrix}\right)^2\delta^2\le\frac{1}{2} \\ \left(\mathbf{M}_x K^{-1}\begin{pmatrix}\Delta\mathbf{y}_c\\0\end{pmatrix}\right)^2\delta^2+\left(\mathbf{M}_y K^{-1}\begin{pmatrix}\Delta\mathbf{y}_c\\0\end{pmatrix}\right)^2\delta^2\le\frac{1}{2} \end{cases}. \tag{3.12}$$

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

To satisfy the above inequalities, let

$$\begin{cases} \left(\left|\mathbf{M}_x K^{-1}\begin{pmatrix}\Delta\mathbf{x}_c\\0\end{pmatrix}\right|+\left|\mathbf{M}_y K^{-1}\begin{pmatrix}\Delta\mathbf{x}_c\\0\end{pmatrix}\right|\right)\delta\le\frac{1}{\sqrt{2}} \\ \left(\left|\mathbf{M}_x K^{-1}\begin{pmatrix}\Delta\mathbf{y}_c\\0\end{pmatrix}\right|+\left|\mathbf{M}_y K^{-1}\begin{pmatrix}\Delta\mathbf{y}_c\\0\end{pmatrix}\right|\right)\delta\le\frac{1}{\sqrt{2}} \end{cases}. \tag{3.13}$$

Therefore, the scaling factor is estimated as,

$$\delta=\min_{X\in S}\delta(X), \tag{3.14}$$

where,

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

for all vertices X of S. Note that the vectors $\Delta\mathbf{x}_c, \Delta\mathbf{y}_c$ are the differences of the current constraint points' positions and their individual tragets' positions. Scaling factor δ depends on the configuration of the current constraint points. For any constraint point C_i, its displacement needs to be scaled by δ in an iterative way so as to approximate its target C_i^*. The configuration of all the current constraint points may be defined as the current state of the mesh. When the constraint points' positions are updated, the state is changed accordingly as well. Thus, δ can further be viewed as the iterative step length of updating state of the mesh.

Our iterative scheme for constrained texture mapping has been outlined. The iterative step length is adaptively estimated by the current constraint point configuration. Before proceeding further, the iterative step length δ of Eq.3.14 is taken as an estimate of the lower bound for our purpose of foldover-free reparameterization. The estimate of Eq.3.14 is sufficient [i.e., there may be an iterative step length δ beyond the estimate of Eq.3.14 to yield a foldover-free solution]. Note that scaling the displacements of the constraint points only eliminates all probable foldover cases in order to guarantee that the mesh topology is continually preserved. The goal of Eq.3.14 is to guarantee that the domain is completely foldover-free. Thus, Eq.3.14 only provides an estimate of the lower bound.

An inevitable issue is the convergence of the proposed iterative process. To answer this issue, let us first give out a proposition,

Proposition:

The algorithm proposed in section 3.1 converges to the desired positions.

Before proceeding our proof, we need to define some notations. Our algorithm proposed in Section 3.3 is denoted as $U = P(X)$, where P denotes the mapping from the mesh to the parameteric domain. Following the Banach fixed point theorem [Kirk and Khamsi 2001], we rewrite it as, $X = P(X) - U + X$, and denote it as $X = A(X)$, where A is viewed as an operator. We expect that the constrained points X can converge to the desired positions U^* in the final deformed mesh. Thus, for the constrained points (note that X refers to the constrained points in this appendix), our algorithm is expressed as $X = P(X) - U^* + X$ and once converged, it is expected that $U^* = A(U^*)$.

Proof:

Let Ω be a complete metric space, e.g. Euclidean space. Define the sequence $\{X_i\}$, for $\forall X_1 \in \Omega$, let $X_2 = A(X_1), X_3 = A(X_2), \ldots\ldots$ This means that the constrained points in our Algorithm might start from any initial locations.

First consider $\forall X, Y \in \Omega$. Let $0 \le \alpha < 1$ and $d(.,.)$ denotes the metric function, e.g. Euclidean distance. Herein let $d(X,Y) = \max\left(\left\|X - U^*\right\|, \left\|Y - U^*\right\|\right)$. Our algorithm yields, $\begin{cases} X = P(X) - U^* + X \\ Y = P(Y) - U^* + Y \end{cases}$. Due to radial shrink toward U^*, we have $d(A(X), A(Y)) \le \alpha d(X,Y)$.

Then consider the sequence of $\{X_i\}$. We have,

$$d(X_2, X_3) = d(A(X_1), A(X_2)) \le \alpha d(X_1, X_2)$$
$$d(X_3, X_4) = d(A(X_2), A(X_3)) \le \alpha d(X_2, X_3) \le \alpha^2 d(X_1, X_2)$$
.....
$$d(X_{n-1}, X_n) = d(A(X_{n-2}), A(X_{n-1})) \le \alpha d(X_{n-2}, X_{n-1}) \le \alpha^{n-2} d(X_1, X_2)$$
.....

For any natural number *k*, we can further yield,

$$\begin{aligned} d(X_n, X_{n+k}) &\le \sum_{i=1}^{k} d(X_{n+i-1}, X_{n+i}) \\ &\le (\alpha^{n-1} + \alpha^n + ... + \alpha^{n+k-2}) d(X_1, X_2) \\ &\le \frac{\alpha^n}{1-\alpha} d(X_1, X_2) \to 0, \quad (n \to \infty) \end{aligned}$$

Thus, let $n \to \infty$, we have $X_n \to U^*$, i.e. $U^* = A(U^*)$.

Furthermore, consider the uniqueness of our algorithm. Suppose that another $\hat{U}^*$ exists and $\hat{U}^* \neq U^*$. Then we have,

$$0 < d(\hat{U}^*, U^*) = d(A(\hat{U}^*), A(U^*)) \le \alpha d(\hat{U}^*, U^*) < d(U^*, U^*),$$

which is contradiction. Proof ends.

Remark

A number of existing approaches [Kraevoy et al. 2003, Lee et al. 2008] have also been used to achieve a foldover-free solution by adding Steiner vertices and using edge-swap operations [Lee et al. 2008]. These are unlike our proposed method, which utilizes successive approximation. Compared to the previous approaches, our proposed method can generate a smooth solution without the need for postprocessing. In addition, because of the continuity of the RBF function, it leads to smaller distortion during reparameterization. These advantages over other methods are further illustrated in the experiment section.

3.3 TRIANGLE SUBDIVISION

In general, the proposed RBF-based reparameterization can effectively generate a continuous deformation to match positional constraints exactly. However, for extreme scenarios with large deformation, Eq.3.2, together with the estimate of Eq.3.14, may not always converge the mesh to the most ideal position. Looking at Fig.3-3 for example, two constraint points are to be swapped while the other two points are fixed. Without triangle subdivision, although the scheme of Eqs.3.2 and 3.14 ensure that the mesh will converge to a foldover-free state (see Fig.3-3b), the position is not ideal. This is a deficiency of our proposed scheme in Eqs.3.2 and 3.14.

New vertices should be added by subdividing the triangles to circumvent this issue. This step is similar to that presented in [Kraevoy et al. 2003, Lee et al. 2008], in which extra Steiner vertices are added. The basic idea of the subdivision strategy in the present study is to first determine the potential folding vertices, and then to identify the edges that the vertices will most likely cross. Thus, the triangles sharing these edges can be subdivided by adding new vertices around the potential folding vertices. The underlying idea is very simple: to approximate the continuous implicit function (i.e. RBF) by local upsampling. More sampling points provide more freedom and the higher the probability that foldovers could be avoided. The iterative step lengths $\delta(X)$ is estimated with Eq.3.14 for all vertices, to determine the potential folding vertices when their $\delta(X)$s are below an empirically selected threshold $\delta_{threshold}$. The approach is summarized as follows. Assume N selected folding vertices:

Determining the Most Probable Edges

DO i = 1, N,

(1) Extract the 1-ring of the selected vertex v_i, then compute the probable location v_i' of v_i by Eq.3.7 (Section 4)** using 2–3 times the threshold $\delta_{threshold}$ (i.e., multiplying the vectors $\Delta\mathbf{x}_c$ and $\Delta\mathbf{y}_c$ with $2\delta_{threshold}$ or $3\delta_{threshold}$;

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

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

END DO

**Equation 3.7 is another expression of Eq.3.2 because Eq.3.7 offers a linear expression about the displacement of the current constraint points.

3.4 EXPERIMENTS AND DISCUSSIONS

In this section, the proposed method is applied to a number of examples to evaluate its validity, efficiency, and robustness. For simplicity, the 2D meshes are normalized in [0,1]×[0,1] domain, and the texture images are similarly normalized, regardless of the aspect ratio. Based on this normalization, the threshold used in the algorithm of Section 3.3 can be preset without further tuning.

3.4.1 FOLDOVER

Figure 3-1 shows the results produced using several established methods [Kraevoy et al. 2003, Tang et al. 2003, Sorkine and Cohen-Or 2004]. As shown in the figure, the methods are incapable of completely circumventing foldovers during the reparameterization process. The first experiment in the present study is to test the proposed method on the same head model in shown in Figure 3-1a for comparison purposed. The initial 2D mesh (i.e., embedding or parameterization) obtained by conventional harmonic mapping is shown in Fig.3-4a. Figure 3-4 shows the results

with different iterations of the RBF-based reparameterization. In Fig.3-4a, red stars mark the constraints that need to move to the points circled in white. No foldover triangles occur during the iterations; thus, the internal constraints are satisfied. Note that during the iterative reparameterization process, the boundary of the 2D parameterized mesh does not have to be fixed on the initial predefined convex domain. Readers are referred to the accompanying video (at http://nccastaff.bournemouth.ac.uk/jzhang/projects.htm). This distinct advantage offers more freedom than the other methods to reduce mesh distortion.

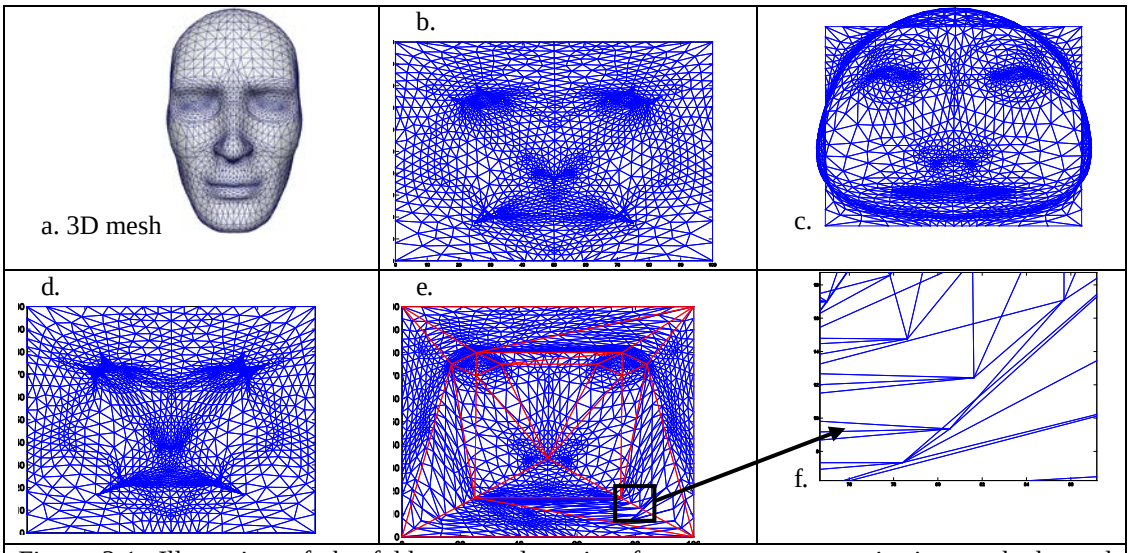


Figure 3-1. Illustration of the foldover results using four recent parameterization methods, and considering internal constraints (Note: the mesh details can be seen more clearly by zooming in on the document): a) 3D mesh; b) least squares meshes [Sorkine and Cohen-Or 2004]; c) RBF-based embedding [Tang et al. 2003]; d) harmonic mapping [Guo et al. 2005, Floater and Hormann 2005]; e) Delauney triangulation-based mapping [Kraevoy et al. 2003] (the red lines mark the boundaries of the triangle patches within which there is no foldover; however, foldover triangles can be observed around the red lines.); and f) inset showing the details of distortion around the red line

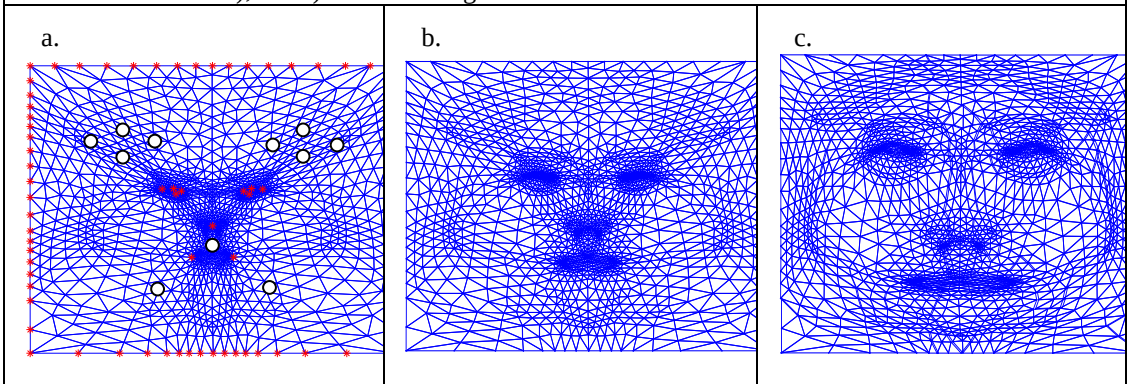


Figure 3-4. Illustration of the iterative results of the proposed method: a) initial mesh with constraint point pairs; b) 3 iterations; and c) 5 iterations

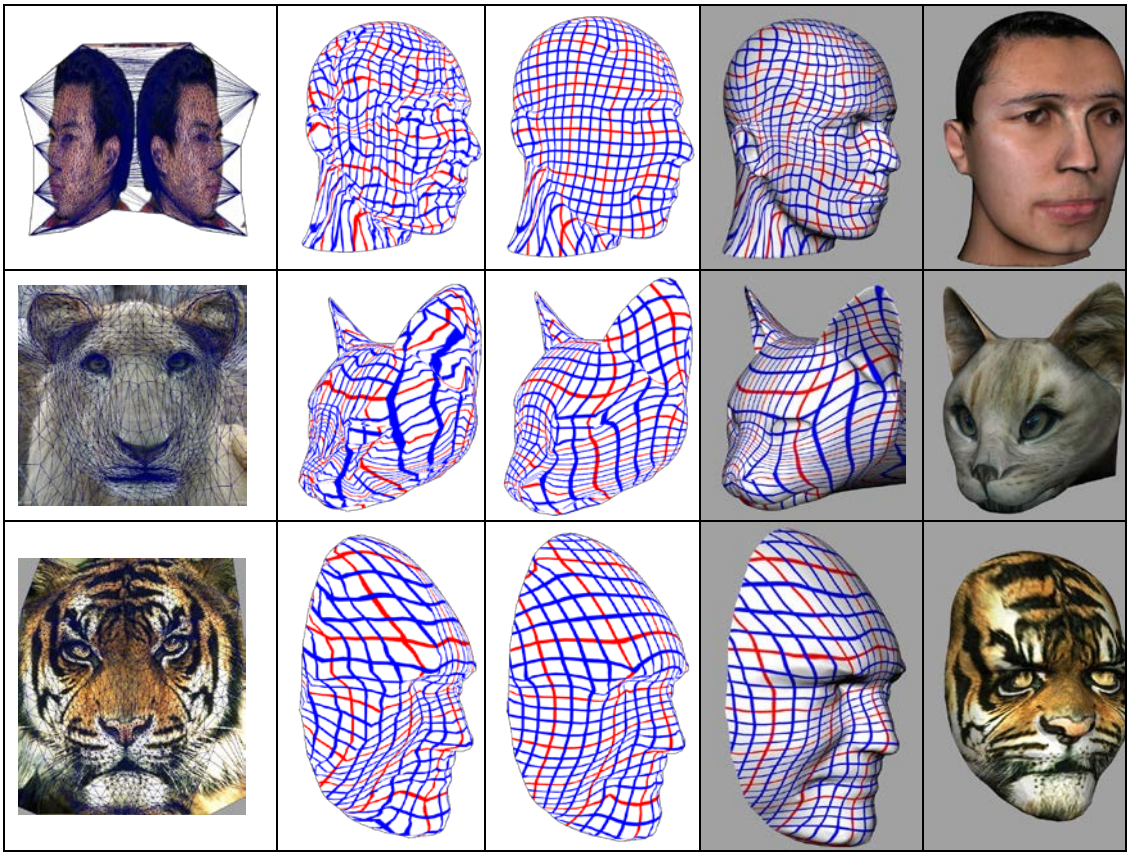
3.4.2 COMPARISON OF EXPERIMENTAL RESULTS

The techniques proposed in Refs. [Kraevoy et al. 2003, Lee et al. 2008] represent state-of-the-art methods in texture mapping subject to hard constraints. [Lee et al. 2008] experimentally showed that their method is able to handle challenging examples and generate satisfactory results. Therefore, in this chapter, the proposed algorithm is compared with the work of [Lee et al. 2008], tested on the same set of models. A further test was performed with a chessboard texture for smoothness comparison. A visual comparison shows that the smoothness of using the proposed approach is much better than that in [Lee et al. 2008] (Fig.3-5, fourth column). In particular, the areas of the constraint points are smoother with the proposed method. This is because the previous method [Lee et al. 2008] cannot ensure smoothness of deformation in such areas. As a result, no further postprocessing for smoothing is necessary in the implementation of the proposed algorithm. In [Kraevoy et al. 2003, Lee et al. 2008], this required postprocessing time usually takes much longer than that of the feature matching process, and becomes the bottleneck of the entire algorithm. However, their results are not very satisfactory without such postprocessing (Fig.3-5, second column).

Moreover, to quantitatively study the distortion of reparameterization, the stretch metrics defined in [Sander et al. 2001] are used. The L-2 norm is used to measure the overall stretch of the parameterization, whereas the L-Inf measures the greatest stretch. Good parameterization is expected to have very small L-2 and L-Inf. These two metrics are used to measure distortion of all the examples in Fig.3-5 (see Table

3-1). The proposed approach performs significantly better in most cases than that of [Lee et al. 2008], even with their smoothing process.

Furthermore, the proposed approach is capable of handling special models that have more than one border. Figure 3-6 shows an example of texture mapping the photograph of an orangutan onto a 3D human head model with three boundaries. This figure shows that the proposed method produces a very smooth parameterization while keeping the two interior boundaries (i.e., the eyes). Applying previous methods [Lee et al. 2008] to this example would usually require extra treatment, such as cutting it into several pieces to ensure each piece has no interior boundaries. The proposed approach is essentially a mesh-free method and does not need new any additional treatment.



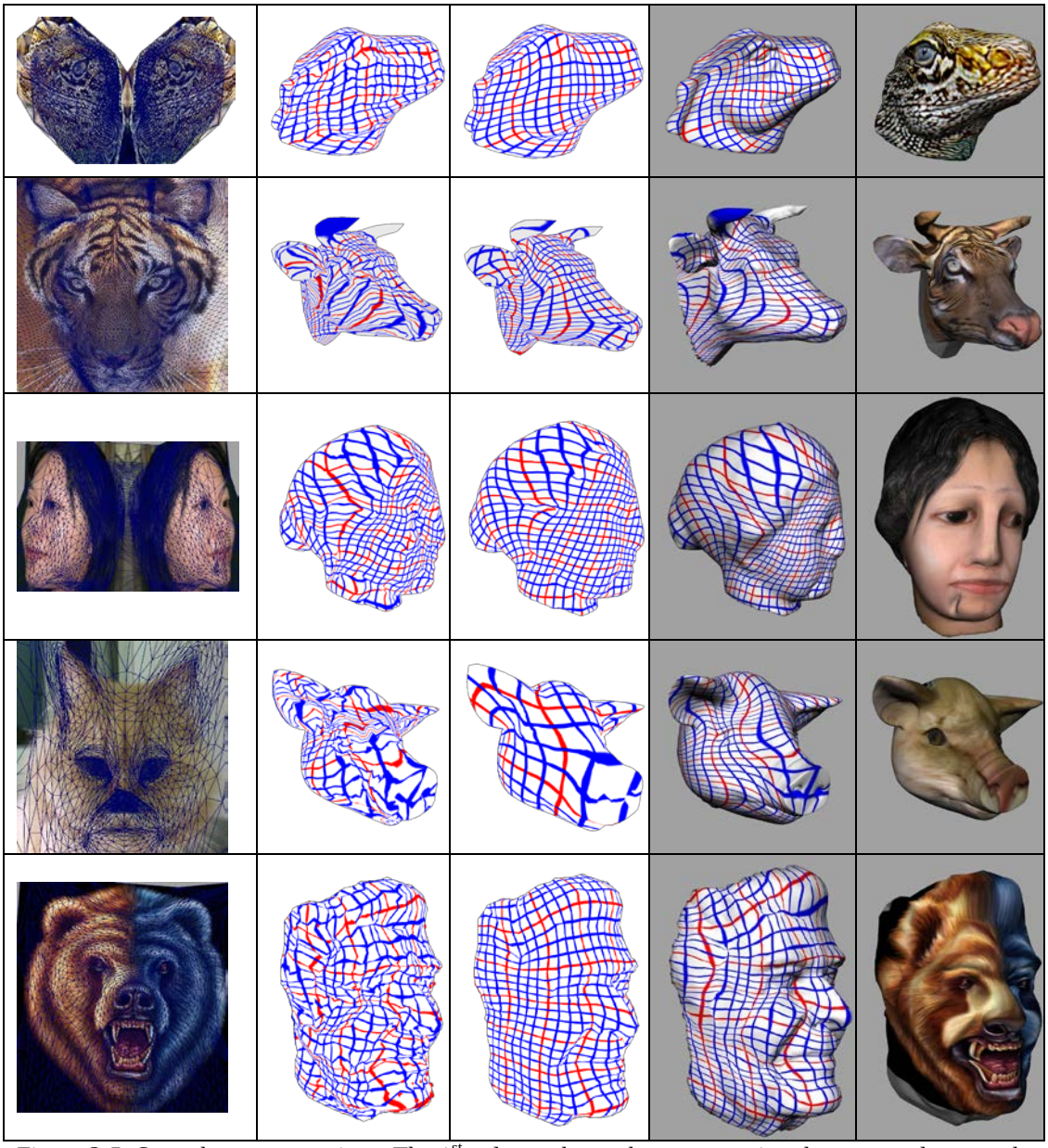


Figure 3-5. Smoothness comparison. The 1st column shows the texture using the proposed approach. The 2nd column shows the results without the post-processing procedure [Lee et al. 2008]. The 3rd column shows the results with the post-processing procedure [Lee et al. 2008]. The 4th column shows the results using our proposed approach. The 5th column shows the final texture mapping results using the proposed approach.

Figure 3-6. Illustration of texture mapping with two interior boundaries.

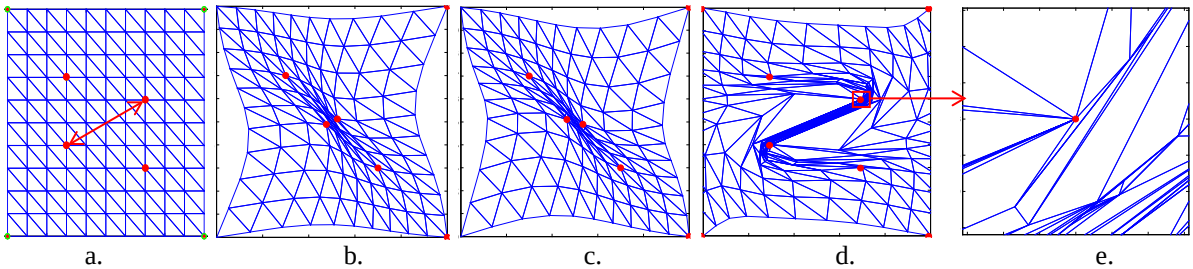
3.4.3 COMPLEXITY ANALYSIS

The core advantage of the proposed RBF-based reparameterization is that the RBF coefficients are updated at every iteration. The main computation cost, $O(2M^3)$, is to determine the inverse of a real symmetric matrix, where M is the number of the constraint points. The time complexity can be estimated as $O(2KM^3)$, where K denotes the number of iterations. Furthermore, considering the triangle subdivision procedure, computing the estimated iterative step lengths using Eq.3.14 at every iteration is necessary. This will cost $O(N)$ each time, where N denotes the number of vertices on the mesh. The time for triangle subdivision is nearly fixed for each selected folding vertex. At each iteration, the running time of the triangle subdivision depends on the number of selected folding vertices m, which is generally much fewer than N. Therefore, the total time cost is $O(K(2M^3 + N + m))$. The majority of the time spent is on the computation of matrix inverse when there are a number of constrained points. The time spent for triangle subdivision is not an issue.

All the experiments were conducted with MatLab on an Intel Pentium 4 3.2 GHz PC with 1 Gb of RAM. Table 3-2 shows the running time of all the examples in Fig.3-5 using the proposed approach, which usually converges around 5–8 iterations. Each example usually takes only several seconds to compute with the proposed method because post-processing for mesh smoothing is unnecessary. In [Lee et al. 2008], the post-processing takes more than 1 minute to obtain the result.

3.4.4 LIMITATIONS

The proposed method deals with soft constraints. In Section 3.5, a subdivision approach is proposed to increase the chances of exactly matching the desired positional constraints. However, this simple approach has its limitations. For example in Fig.3-3a, two constrained points can move close to each other, but not reach the desired positions, even when the triangle subdivision strategy is applied. The displacement vectors can be rotated to circumvent this issue. The distance between any two constrained points is taken, and the displacement vectors of the two selected constrained points are rotated 90° clockwise. Figure 3-3c shows the intermediate result of rotating the displacement vectors. Figures 3d–3e show the convergence result with triangle subdivision and displacement rotation, where the desired deformation is achieved without triangle foldover. Thus, the above issue is successfully addressed. However, rotating the displacement vectors might fail if too many constrained points crowd together. Fortunately, such extreme cases are rarely seen in texture mapping applications. Another limitation is that the convergence of the proposed iterative algorithm has not been proven. Although this is not an empirical issue, ideally, mathematical comprehensive proof should still be given. This will be studied in future work.



a. b. c. d. e.

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

Table 3-1. Distortion metrics of texture mapping examples in Fig.3-5.

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

We highlight the cases that the performance of our proposed approach is worse than that of [Lee et al. 2008] by shading.

Table 3-2. Statistics of texture mapping examples in Fig.3-5.

Examples	Row 1	Row 2	Row 3	Row 4	Row 5	Row 6	Row 7	Row 8
#Vertices	5184	1770	1808	10017	10736	4149	1772	1657
#Triangles	10354	3526	3602	20008	21404	8284	3450	3300
#Features	83	24	25	54	32	71	21	27
#Added Vertices	0	60	0	0	1556	0	2	0
#Added Vertices [Lee]	183/38	77/7	128/10	94/34	127/8	122/25	383/21	69/2
Time (s)	8.94	5.12	3.86	13.71	16.93	7.21	4.08	3.45

Row 6 shows the numbers of the added points before and (/) after mesh optimization [Lee et al. 2008].

Table 3-2 shows the results of the triangle subdivision of all the examples in Fig.3-5. In Row 5, the number of the added vertices in the proposed approach is greater than that in [Lee et al. 2008]. This implies that the simple subdivision approach adds multiple redundant vertices. Looking at Fig.3-6, the number of added vertices may depend on the level of smoothness and distortion because additional vertices are necessary for smoothness and low distortion. Nevertheless, the newly added vertices only increase the vertex number N on the mesh, rather than the constraint vertex number M or the selected folding vertex number m. Therefore, this does not result in a visible increase of total running time.

Further extension of the triangle foldover issue will incur another research focus: global self-intersection, that the boundary intersects itself. This still remains challenging. We think, Jacobian constraint of Eq.3.3 cannot sufficiently prevent the

global self-intersection.

3.5 CHAPTER SUMMARIZATION

This Chapter addresses the issue of the surface parameterization with the usual positional constraints. The motivation is from the classic texture mapping applications. Thus, this chapter focuses on the positional constraints and the caused triangle foldovers rather than the caused distortions. To this end, we first presented the foldover-free condition, and further proposed the RBF-based reparameterization approach incorporating with this constraint condition. However, in some extreme scenarios, the proposed method cannot converge to the desired positions. To handle such challenges, we have to introduce the triangle subdivision techniques into the implementation of the proposed RBF-based reparameterization approach.

Our method is applied to the applications of 2D mesh re-parameterization. It can actually be extended into 3D or higher dimensional datasets, e.g. volume data and polyhedrons. This will be the task of the next chapter.

CHAPTER 4

EXTENSION OF RBF-BASED RE-PARAMETERIZATION METHOD TO HIGH DIMENSIONAL DATASETS

4.1 FOLDOVERS IN HIGH DIMENSIONAL DATASETS

This chapter will extend the RBF-based reparameterization approach presented in Chapter 3 to high dimensional datasets. In computer graphics, high dimensional warping approaches have been widely applied to scattered data interpolation, feature-based metamorphosis for 2D images and 3D volume data [Lerios et al. 1995], texture mapping, motion synthesis [Kovar and Gleicher 2004], free form deformations (FFDs) [Sederberg and Parry 1986] and many more. For visualization purposes, we perform it on surface parameterization, polyhedral deformation and 3D volume data registration.

In surface parameterization, a 3D surface might firstly be mapped onto a convex 2D domain without foldovers, and then the resulting parameterization is further deformed to match the internal positional constraints. Foldovers within a mesh can be observed when the internal constraint points are taken into account. Figure 3-1 of Chapter 3 shows some examples of parameterization, which satisfy the given positional constraints. Triangle foldovers can be observed clearly.

Voxel-based volume deformation has been widely used in image segmentation and registration, e.g. CT and MRI datasets. Foldovers within the volume deformation usually hide in the volume dataset instead of its surface. For images or volume dataset warping, the connection relationship of the pixels or voxels is pre-defined by their grid structures. Due to dense pixels or voxels, it is easy to observe that multiple pixels or voxels are mapped into the same place, which is called foldover [Fujimura and Makarov 1998]. Usually, foldovers result in unexpected unsmooth features in the deformed images or volume datasets, such as irregular edges, spikes and jumps. To see it clearly, we performed 3D FFDs on the precentral gyrus of a segmented MRI brain volume dataset as shown in Figure 4-6a to 6c. We use the Marching Cubes algorithm to extract the isosurface. One can observe the tetrahedral folding. For 3D meshes, although the distribution of the vertices is sparser than that of the voxels in 3D volume dataset, foldovers would still take place during a large deformation. Figure 4-1 shows a simple example of blowing up a segment of tube. The internal and external surfaces of the tube are represented in a polyhedral form. When the deformation of the internal surface is too large, the internal surface intersects with the external one.

Our experience from implementing the six existing methods suggests that none of them are able to robustly overcome the challenge of mesh or tetrahedral foldover,

which is an unsolved issue in high dimensional deformable fields as well. In the following we present our solution to this issue. We will first give out an overview of our foldover-free deformation method, which incorporates radial basis functions in our iterative deformation mechanism. Then, we will further give out the estimate of the iterative step length for 2D mesh and high dimensional dataset deformation.

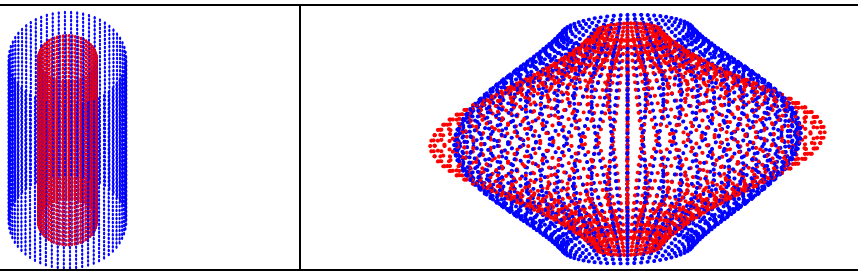


Figure 4-1. The illustration of the tetrahedral foldover on a 3D polyhedral tube. The red denotes the vertices of the internal surface, while the blue denotes those of the external surface. For illustration, we only show the vertices here.

4.2 ALGORITHM OVERVIEW

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

$$T : \begin{cases} X \in S \rightarrow U(X) \in \Omega \\ \text{subject to } U(X_i^*) = U_i^*, i = 1,...,m \end{cases}. \tag{4.1}$$

Our algorithm is developed based on radial basis functions. This is due to many well behaved properties of the RBF scheme, e.g. mesh-free and C^2 continuous. The most important is that the RBF scheme is suitable for implementation in an iterative manner, as will be seen later.

Unlike the previous applications of RBFs [Guo et al. 2005, Tang et al. 2003], in our algorithm, the RBF scheme is used to compute the displacement of the point's coordinates,

$$\Delta u_i = P_i(X) + \sum_j^m \lambda_{ij} \phi(\left\| X - C_j \right\|), \quad i = 1,...,n \tag{4.2}$$

where the displacement is represented as a vector of $\Delta U = (\Delta u_1,...,\Delta u_n)^T$, RBF coefficients denote λ_{ij}, $C_j \in R^n$ denotes the constraint points, and $P_i(X)$ is a affine transformation, i.e. $P_i(X) = a_{i0} + \sum_k^n a_{ik} x_k$, n denotes the dimensionality of the datasets.

Although there are various forms of radial basis functions, we adopt the thin plate spline as ϕ here for simplicity. The deformed U is obtained by updating $U(X) = X + \Delta U$. For next iteration, let $X \leftarrow U(X)$.

Our deformation algorithm is summarized as below. The proposed algorithm is executed iteratively, and the superscript j stands for the iteration index. The iterative mechanism will be described in detail in the following section.

(1) Input: Initial dataset $S^{(0)}$ and a set of user-specified constraint point pairs $(C_i^{(0)}, C_i^*), i = 1,...,m$;
(2) Loop: estimate the scaling factor δ by Eq.4.15 (see section 4.3.2) based on the configuration of the current m constraint point pairs $(C_i^{(j)}, C_i^*), i = 1,...,m$;
(3) Computing the current constraint points' displacements by $\Delta C_i^{(j+1)} = \delta(C_i^* - C_i^{(j)})$, such that $C_i^{(j+1)} = C_i^j + \Delta C_i^{(j+1)}$;
(4) Computing the displacements of points on $S^{(j)}$ by Eq.4.6 (see section 4.3.2) and updating $S^{(j)} \to S^{(j+1)}$;
(5) End Loop until $C_i^{(n)} = C_i^*$.

4.3 FOLDOVER-FREE ITERATIVE MECHANISM

From a mathematical point of view, a "*foldover-free*" deformation gives a "*one-to-one*" mapping between the original surfaces (or datasets) and their target domains. We will first give out the foldover-free condition, and then formulate our iterative mechanism in a general form.

4.3.1 FOLDOVER-FREE CONDITION

The goal of our work is to develop a foldover-free deformation approach with a set of the positional constraints. This requires that the mapping T is globally univalent or "globally one-to-one", that is, the topology or the connection relationship between any pair of vertices in the datasets should keep unchanged before and after the deformation. Mathematically it means the determinant of the Jacobian matrix must be positive everywhere,

$$\det(\nabla U) > 0 . \tag{4.3}$$

According to the Gerschgorin circle theorem, a sufficient condition of satisfying Eq.4.3 is expressed as,

$$\frac{\partial u_i}{\partial x_i} > \sum_{j=1, j\neq i}^{n} \left| \frac{\partial u_i}{\partial x_j} \right|, \tag{4.4}$$

where i = 1,…,n. The geometric meaning of Eq.4.4 is that the vectors $\partial u_i / \partial (x_1, ..., x_n)$ are linearly independent of each other. For 2D scenarios, this implies that the included angle of the vectors is less than π. For 3D scenarios, such three vectors should not stay within a plane. Usually, Eq.4.4 is called as the *foldover-free condition*.

4.3.2 ITERATIVE FRAMEWORK

Our deformation algorithm employs an iterative framework and the displacements of vertices are determined considering the condition of Eq.4.4, which eliminates foldovers if satisfied. We rewrite Equation 4.2 here, such that the displacements linearly depend on the constraint points. To this end, the RBF coefficients $(\lambda_i, a_i), i = 1,...,n$ can be computed by the following linear system, where $\lambda_i = (\lambda_i^1,...,\lambda_i^m)^T, a_i = (a_i^0,...,a_i^n)^T$,

$$\begin{pmatrix} \lambda_i \\ a_i \end{pmatrix} = K^{-1} \begin{pmatrix} \Delta c_i \\ 0 \end{pmatrix}. \tag{4.5}$$

Herein, there are *m* given constraint points. The displacements of such constraint points denotes as $\Delta c_i = \left(\Delta x_i^1,...,\Delta x_i^m\right)^T$. *K* is a symmetric matrix filled with the radial basis functions $\varphi_{kj} = \varphi\left(\left\|C_k - C_j\right\|\right), k, j = 1,...,m$ and the constraint points' coordinates. (For a detailed RBF representation, refer to [Carr et al. 2001])

Substituting (λ_i, a_i) into Eq.4.2, we can obtain a new expression of Eq.4.2 as follows, *i*=1,…,*n*,

$$\begin{cases} \Delta u_i = M(X) K^{-1} \begin{pmatrix} \Delta c_i \\ 0 \end{pmatrix} \\ M(X) = \left(\varphi(\left\|X - C_1\right\|),...,\varphi(\left\|X - C_m\right\|), x_1,...,x_n,1\right) \end{cases} \tag{4.6}$$

Note that Equation 4.6 describes a linear system of solving the displacement of any *X* (i.e. $\{\Delta u_i\}$) by using those of the constraint points, (i.e. $\{\Delta c_i\}$). The kernel is

$M(X)K^{-1}$ that describes the current configuration of the constraint points. The displacements $\{\Delta c_i\}$ guide to the convergence direction. The derivatives of $\partial(u_1,...,u_n)/\partial(x_1,...,x_n)$ are computed as follows,

$$\frac{\partial u_i}{\partial x_j}=\begin{cases}1+M_{x_j}K^{-1}\begin{pmatrix}\Delta c_i\\0\end{pmatrix} & i=j\\ M_{x_j}K^{-1}\begin{pmatrix}\Delta c_i\\0\end{pmatrix} & i\neq j, i, j=1,...,n\end{cases}, \tag{4.7}$$

where M_{x_j} denotes the partial derivatives of $M(X)$.

Moreover, substituting the above derivatives into Eq.4.4 yields,

$$1+M_{x_i}K^{-1}\begin{pmatrix}\Delta c_i\\0\end{pmatrix}>\sum_{j=1,\,j\neq i}^{n}\left|M_{x_j}K^{-1}\begin{pmatrix}\Delta c_i\\0\end{pmatrix}\right|,\ i=1,\ldots,n. \tag{4.8}$$

In general, the displacements of the constraint points $\{\Delta c_i\}$ can be obtained by the differences of the current constraint points' coordinates and their individual targets' coordinates. To satisfy the above inequalities, we may limit the length of each displacement vector by scaling the vectors $\{\Delta c_i\}$ as follows,

$$1+M_{x_i}K^{-1}\begin{pmatrix}\Delta c_i\\0\end{pmatrix}\delta>\sum_{j=1,\,j\neq i}^{n}\left|M_{x_j}K^{-1}\begin{pmatrix}\Delta c_i\\0\end{pmatrix}\right|\delta, \tag{4.9}$$

where δ denotes a scaling factor and δ>0.

The regions defined by the above inequalities can be further described as follows, $i=1,\ldots,n$,

$$\Omega_i(\delta)=\left\{\begin{array}{l}(r_1,...,1+r_i,...,r_n):\left|r_j\right|\leq\left|M_{x_j}K^{-1}\begin{pmatrix}\Delta c_i\\0\end{pmatrix}\right|\delta,\ j=1,...,n\\ 1+r_i>\sum_{j=1,\,j\neq i}^{n}\left|r_j\right|\end{array}\right\}, \tag{4.10}$$

where $(r_1,...,r_n)$ denotes a point in Ω. The scaling factor δ is not a constant, and depends on the displacements of the constraint points. For simplicity, assume that the vectors $\{\Delta c_i\}$ have the same distribution. The regions $\Omega_i(\delta)$ would then have the same size in terms of Eq.4.10. In a n-dimension Euclidean space, the condition of Eq.4.4 can be described as,

$$\sum_{j=1}^{n}\left(M_{x_j} K^{-1}\begin{pmatrix}\Delta c_i \\ 0\end{pmatrix}\right)^2 \delta^2 < \frac{1}{n}, \tag{4.11}$$

for the region $\Omega_i(\delta)$ and i=1,…,n.

To further illustrate the geometric meaning of Eq.4.10, we consider the 2D and 3D scenarios here. When n=2, Eq.4.10 is expressed as,

$$\begin{cases} \Omega_1(\delta) = \left\{ (1+r_1, r_2) : |r_1| \le \left| \mathbf{M}_x K^{-1}\begin{pmatrix}\Delta c_1 \\ 0\end{pmatrix}\right| \delta, |r_2| \le \left| \mathbf{M}_y K^{-1}\begin{pmatrix}\Delta c_1 \\ 0\end{pmatrix}\right| \delta, 1+r_1 > |r_2| \right\} \\ \Omega_2(\delta) = \left\{ (r_1, 1+r_2) : |r_1| \le \left| \mathbf{M}_x K^{-1}\begin{pmatrix}\Delta c_2 \\ 0\end{pmatrix}\right| \delta, |r_2| \le \left| \mathbf{M}_y K^{-1}\begin{pmatrix}\Delta c_2 \\ 0\end{pmatrix}\right| \delta, 1+r_2 > |r_1| \right\} \end{cases}. \tag{4.12}$$

Figure 4-2 illustrates the regions of $\Omega_1(\delta),\Omega_2(\delta)$. The dashed line is used to highlight these undetermined boundaries.

The condition of Eq.4.4 implies that vectors $\partial u/\partial(x,y), \partial v/\partial(x,y)$ should be linearly independent of each other. Figure 4-2 intuitively illustrates this concept by the five lines: l_1, l_2, l_3, l_4, l_5. For example, r_1 and r_2 should be above the line l_3 or under the line l_4, and above the line l_5, so that the linear independence can be guaranteed. This can be achieved by the scaling factor in Eq.4.10. Moreover, for simplicity, assume that the vectors $\Delta c_1, \Delta c_2$ have the same distribution. The regions $\Omega_1(\delta),\Omega_2(\delta)$ would then have the same size in terms of Eq.4.10. This will lead to the

overlap of straight lines l_1 and l_2 and form a new dividing line, which is $r_1 + r_2 = 0$ in Fig.4-2. Line $r_1 + r_2 = 0$ guarantees the included angle is less than π. Consequently, the condition of Eq.4.4 can be re-expressed as,

$$\begin{cases} \left(M_x K^{-1} \begin{pmatrix} \Delta c_1 \\ 0 \end{pmatrix} \right)^2 \delta^2 + \left(M_y K^{-1} \begin{pmatrix} \Delta c_1 \\ 0 \end{pmatrix} \right)^2 \delta^2 < \frac{1}{2} \\ \left(M_x K^{-1} \begin{pmatrix} \Delta c_2 \\ 0 \end{pmatrix} \right)^2 \delta^2 + \left(M_y K^{-1} \begin{pmatrix} \Delta c_2 \\ 0 \end{pmatrix} \right)^2 \delta^2 < \frac{1}{2} \end{cases}. \tag{4.13}$$

Moreover, when *n*=3, Eq.4.10 is expressed as,

$$\Omega_1(\delta) = \left\{ \begin{array}{l} (1 + r_1, r_2, r_3) : |r_1| \le \left| M_x K^{-1} \begin{pmatrix} \Delta c_1 \\ 0 \end{pmatrix} \right| \delta, \\ |r_2| \le \left| M_y K^{-1} \begin{pmatrix} \Delta c_1 \\ 0 \end{pmatrix} \right| \delta, \\ |r_3| \le \left| M_z K^{-1} \begin{pmatrix} \Delta c_1 \\ 0 \end{pmatrix} \right| \delta, 1 + r_1 > |r_2| + |r_3| \end{array} \right\},$$

$$\Omega_2(\delta) = \left\{ \begin{array}{l} (r_1, 1 + r_2, r_3) : |r_1| \le \left| M_x K^{-1} \begin{pmatrix} \Delta c_2 \\ 0 \end{pmatrix} \right| \delta, \\ |r_2| \le \left| M_y K^{-1} \begin{pmatrix} \Delta c_2 \\ 0 \end{pmatrix} \right| \delta, \\ |r_3| \le \left| M_z K^{-1} \begin{pmatrix} \Delta c_2 \\ 0 \end{pmatrix} \right| \delta, 1 + r_2 > |r_1| + |r_3| \end{array} \right\},$$

$$\Omega_3(\delta) = \left\{ \begin{array}{l} (r_1, r_2, 1 + r_3) : |r_1| \le \left| M_x K^{-1} \begin{pmatrix} \Delta c_3 \\ 0 \end{pmatrix} \right| \delta, \\ |r_2| \le \left| M_y K^{-1} \begin{pmatrix} \Delta c_3 \\ 0 \end{pmatrix} \right| \delta, \\ |r_3| \le \left| M_z K^{-1} \begin{pmatrix} \Delta c_3 \\ 0 \end{pmatrix} \right| \delta, 1 + r_3 > |r_2| + |r_1| \end{array} \right\}.$$

Figure 4-3 shows the regions of $\Omega_1(\delta),\Omega_2(\delta),\Omega_3(\delta)$. Taking the condition of Eq.4.4 into account, we hope to point out that the dividing plane of $r_1+r_2+r_3=0$ can guarantee the determinant of Jacobians is greater than Zero. Similar to 2D scenarios, the condition of Eq.4.4 is expressed as,

$$\begin{cases} \left(M_x K^{-1}\begin{pmatrix}\Delta c_1\\ 0\end{pmatrix}\right)^2\delta^2+\left(M_y K^{-1}\begin{pmatrix}\Delta c_1\\ 0\end{pmatrix}\right)^2\delta^2+\left(M_z K^{-1}\begin{pmatrix}\Delta c_1\\ 0\end{pmatrix}\right)^2\delta^2<\frac{1}{3} \\ \left(M_x K^{-1}\begin{pmatrix}\Delta c_2\\ 0\end{pmatrix}\right)^2\delta^2+\left(M_y K^{-1}\begin{pmatrix}\Delta c_2\\ 0\end{pmatrix}\right)^2\delta^2+\left(M_z K^{-1}\begin{pmatrix}\Delta c_2\\ 0\end{pmatrix}\right)^2\delta^2<\frac{1}{3}. \\ \left(M_x K^{-1}\begin{pmatrix}\Delta c_3\\ 0\end{pmatrix}\right)^2\delta^2+\left(M_y K^{-1}\begin{pmatrix}\Delta c_3\\ 0\end{pmatrix}\right)^2\delta^2+\left(M_z K^{-1}\begin{pmatrix}\Delta c_3\\ 0\end{pmatrix}\right)^2\delta^2<\frac{1}{3} \end{cases} \tag{4.14}$$

Furthermore, the scaling factor is estimated as,

$$\delta=\min_{X\in S}\delta(X) \tag{4.15}$$

where,

$$\delta(X)=\min_{i=1}^{n}\left\{\frac{1}{\sqrt{n}}\left(\sum_{j=1}^{n}\left(M_{x_j}K^{-1}\begin{pmatrix}\Delta c_i\\ 0\end{pmatrix}\right)^2\right)^{-1/2}\right\}.$$

Note that the vectors $\Delta c_i, i=1,...,n$ are the differences of the current constraint points' positions and their individual targets' positions. Scaling factor δ depends on the configuration of the current constraint points. For any constraint point C_i, its displacement needs to be scaled by δ in an iterative way so as to approximate its target C_i^*. The configuration of all the current constraint points may be defined as the current state of the dataset. When the constraint points' positions are updated, the state is changed accordingly as well. Thus, δ can further be viewed as the iterative step length of updating the state of the dataset.

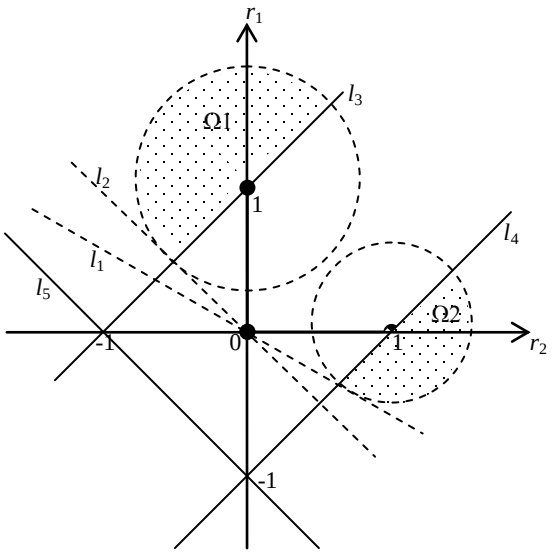


Figure 4-2. Illustration of the condition of Eq.4.4 in 2D scenarios. The dashed lines denote the undetermined boundaries.

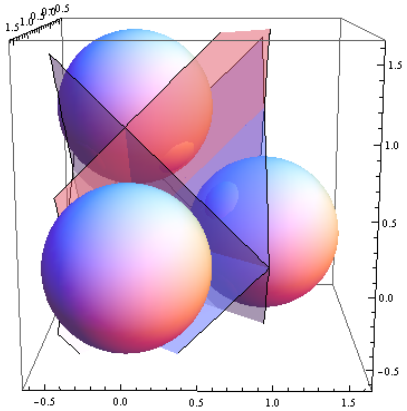


Figure 4-3. The illustration of the condition of Eq.4.4 in 3D scenarios.

So far we have outlined the proposed *iterative deformation mechanism* and given out an estimate of the scaling factor. The displacements of the constraint points are adaptively changed in terms of the estimated scaling factor δ of Eq.4.15. However, a critical problem arises, that is, because the proposed iterative mechanism is namely

to divide $\Delta C_i = C_i^* - C_i^{(0)}$ into a few segments in terms of the scaling factor, how does it guarantee that the dataset could be deformed to the desired deformation in such a successive approximation manner? Before proceeding further, we firstly assume that there exists a foldover-free solution for deformation based on the given constraint point pairs. This is because for some special constraint configurations, the final deformed dataset may not always converge to the most ideal positions, although it can produce a valid useable deformation. Looking at Figure 4-4 for example, there are two constraint points to be moved in opposite directions in a mesh. Occasionally their moving paths may intersect each other, e.g. moving from a red point to the white point. Our approach can only converge at an intermediate state. Obviously, there is no foldover-free solution for the final desired mesh deformation here.

Then, we also hope to point out that Equation 4.15 is a sufficient condition, which means that there might be some solution that does not satisfy Equation 4.15 but still is foldover-free. Note that the presented foldover free condition of Eq.4.15 only eliminates all probable cases of the foldover (see the dividing lines in Fig.4-2 and 4-3), otherwise cannot guarantee the mesh topology-preserving continually. The usual case is that the warp against the Equation 4.15 is foldover-free at some points while causes foldovers at the other points. The goal of Equation 4.15 is to eliminate all potential foldover cases over the domain.

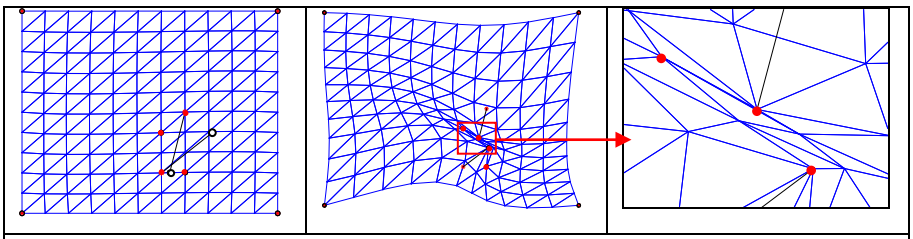


Figure 4-4. Illustration of convergence of an extreme case.

In general, to estimate the scaling factor δ, it is required to test all the points X within a dataset. However, our observation suggests that the constraint points always hold larger displacements than others at each iteration. Thus, it is sufficient to estimate the δ only by testing the constraint points instead of all the points.

The core of our RBF-based deformation approach 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 $O(nm^3)$, where m is the number of the constraint points and n denotes the dimensionality of the dataset. The time complexity can be estimated as $O(knm^3)$, where k denotes the iteration number. In practice, the number of constraint points is always far smaller than that of the points within a dataset. And our algorithm usually converges with 3-6 iterations. Although it is costly to invert a matrix, because the dimension of the matrices to be inverted is small, in practice the cost is negligible.

4.4 EXPERIMENTS AND DISCUSSIONS

In this section, we perform the above-developed method on three kinds of datasets, 3D human brain volume data, 3D polyhedron and 2D mesh.

4.4.1 EXPERIMENTS

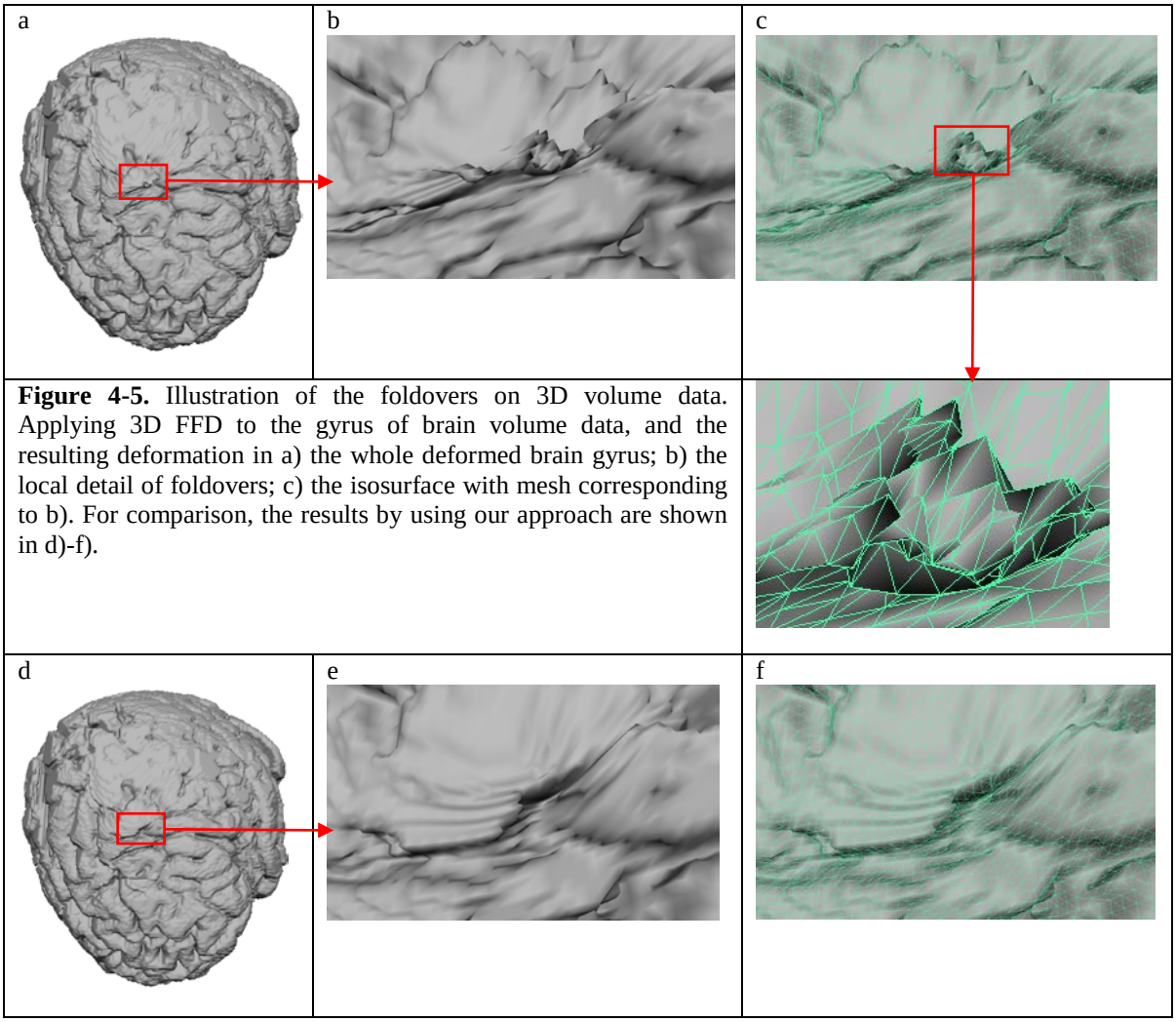


Figure 4-5. Illustration of the foldovers on 3D volume data. Applying 3D FFD to the gyrus of brain volume data, and the resulting deformation in a) the whole deformed brain gyrus; b) the local detail of foldovers; c) the isosurface with mesh corresponding to b). For comparison, the results by using our approach are shown in d)-f).

3D Volume Data

The first 3 images of Figure 4-5 show the deformed precentral gyrus of a human brain MRI volume data using the 3D FFD approach. For comparison, the last 3 images of Figure 4-5 show the results of our approach. We utilized the Marching Cubes to extract the isosurface for rendering purposes. This can highlight the foldovers occurred in the 3D volume data by comparing the meshes shown in Figure 4-5c and 5f. One can note the folding of triangles, i.e. some edges insert in the faces instead of connecting the edges or vertices of triangles in Fig.4-5c.

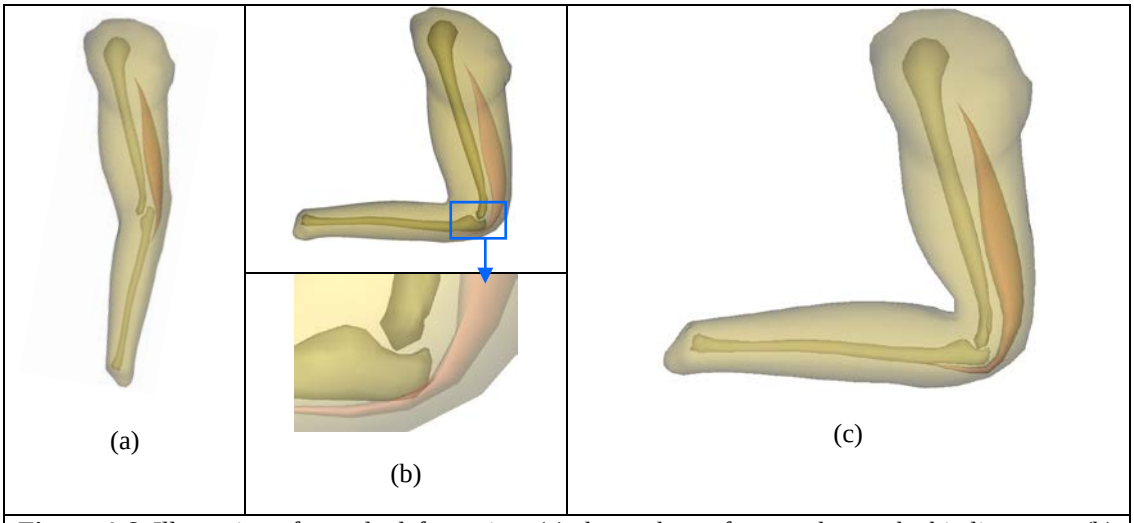


Figure 4-6. Illustration of muscle deformation. (a) shows the surface meshes at the binding pose; (b) shows the collision occurs between the bone and muscle; (c) shows the foldover-free result by using our method, i.e. collision is solved.

3D Mesh

Figure 4-6 shows another useful application of our method for muscle deformation. In this skeleton-muscle-skin three layers system, the bone's movement follows rigid transformations, while the skin and muscle are deformed by using the Maya muscle package. When a character moves its arm, sometimes the collision between these three structures is inevitable. So, we converted the three surface models into a tetrahedral volume mesh. The collision between the surface meshes becomes a foldover problem in the polyhedral mesh. One can note that the low bone goes in the muscle in Fig.4-6b.

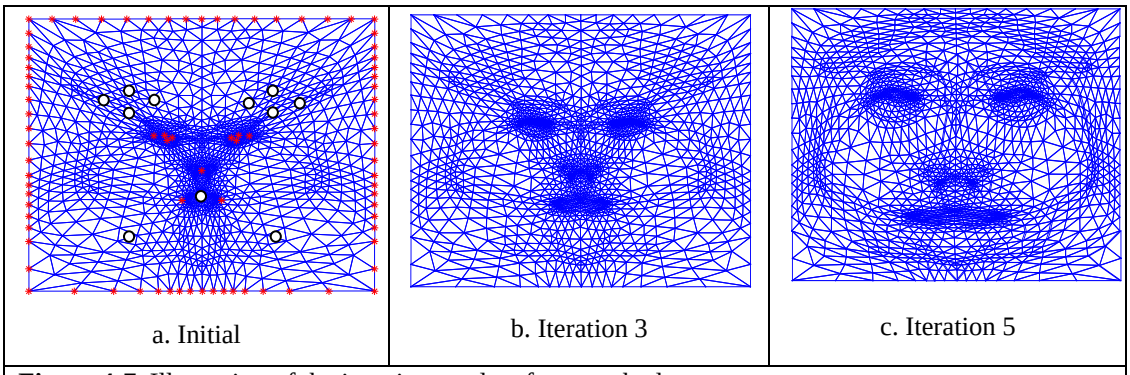


Figure 4-7. Illustration of the iterative results of our method.

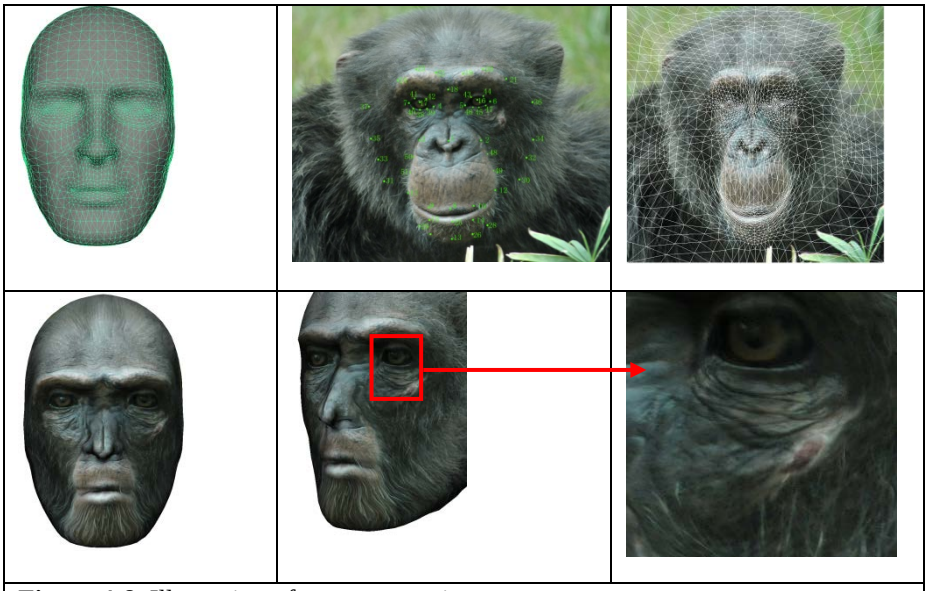


Figure 4-8. Illustration of texture mapping.

2D Mesh

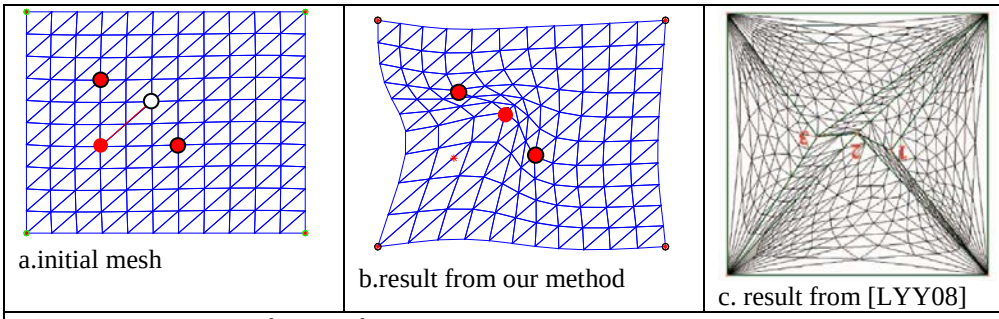
Figure 3-1 shows the results created with four recently published methods, which, as can be seen from the Figure 3-1b to 1f, are not able to avoid foldovers completely during the re-parameterisation process. In order to make a comparison with them, we first tested the above-developed method on the same head model as shown in Figure 3-1a. An initial 2D mesh is generated by harmonic mapping in advance, and the constraint point pairs are shown in Figure 4-7a. The constraint points marked with the red stars need to move to the individual targets with white circles. The results of each iteration with our *RBF-based* re-parameterisation are shown in Figure 4-7b and 7c. The results confirm that by incorporating our iterative deformation mechanism, no folding of triangles appears during the deformation.

As an application of surface parameterization, we then applied our approach to texture mapping. The texture of an orangutang's photo is mapped onto the 3D human head model as shown in Figure 4-8. The zoomed-in image shows that our method produces a very smooth parameterization. This example also demonstrates that our

method is very fast. It converges in only five iterations, even when a large number of constraints are involved. This is very encouraging as high quality texture mapping is essential in animation production.

4.4.2 DISCUSSIONS

There is already existing literature on distortion measurement for 2D meshes [Sander et al. 2001, Knupp 2001]. [Sander et al. 2001] proposed some stretch metrics, whereby 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. Although a full treatment of distortion measurement is beyond the scope of this paper, it is informative to use these two metrics (i.e. L-2 and L-Inf) to measure the distortion from our texture mapping experiment in Fig.4-9 (#vertices:1672, #constraints:51, #iteration:5, L-2:1.019, L-Inf:4.447, time:3.17s). We also used an extreme example similar to Figure 12 in [Lee et al. 2008] for comparison purposes. A square mesh is deformed by our *foldover-free* re-parameterization approach with some specified constraint points. To highlight the issue of distortion, in Figure 4-9a only one constraint point is specified to move to a new location (white circle). The orientation of the other two constraint points is unchanged. It can be seen that our approach does not result in large deformation. Compared with the result (Figure 12e) from [Lee et al. 2008], in terms of deformation, ours is more desirable.



a.initial mesh

b.result from our method

c. result from [LYY08]

Figure 4-9. Distortion from surface re-parameterization.

As can be seen from the results, although our RBF-based deformation approach does specifically aim for minimal distortion, the iterative deformation mechanism guarantees the deformation fields to be diffeomorphic. This is in favour of distortion reduction.

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 for 2D meshes and 3D polyhedrons.

4.5 CHAPTER SUMMARIZATION

This chapter extends the RBF-based reparameterization method proposed in Chapter 3 to high dimensional datasets. The main contribution is to give the generalization of the RBF-based reparameterization with the positional constraints.

However, Chapter 3 and 4 focuses on the positional constraints rather than the caused distortion (i.e. rigid constraint). Next chapter will take the caused distortion into account as well.

CHAPTER 5

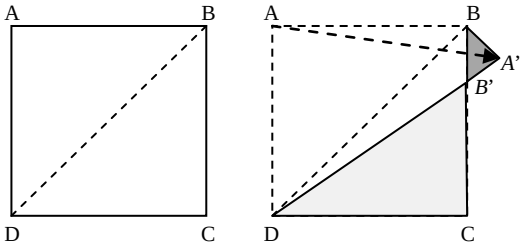
TOPOLOGY PRESERVED SHAPE DEFORMATION

5.1 PROBLEM FORMULATION

Topology preservation and low distortion are always concerned in many image and shape deformation applications. Indeed, the challenging issue is to seek a tradeoff between the topology preservation and the distortion in terms of various applications. In this chapter, we will address these two issues.

Regarding image warping, there exist two kinds of meshes for image discretization, i.e. triangle and quadrangle meshes. In our paper, we prefer triangle mesh. This is due to the following observation. Consider a quad cell of a quad mesh as shown in Fig.5-1a. If we allow a large irregular deformation, the quad mesh can cause many numerical difficulties. For example as shown in Fig.5-1b, point *A* moves a large distance across the edge *BC* to A'. This movement causes a twist of the quad

ABCD, and this twist cannot be detected by computing the area of this quad. The quad area is obtained by the difference between the areas of the triangles DCB' and $A'BB'$. Let point A' be close to point *B*. The quad area increases while the twist happens. If the quad cell is split into two triangles, the twist makes the area of triangle $A'BB'$ become negative. This is called triangle flipping or foldover. There is no foldover if the sign of the triangle area keeps unchanged. Therefore, discretization on a triangulation will effectively help detect and further prevent foldovers and singular Jacobians during deformation.



a) a quad cell b) twist

Figure 5-1. Deformation on quad and triangle meshes.

Moreover, we further illustrate that the triangle foldovers cannot be avoided by using the current proposed As-Rigid-As-Possible approaches [Igarashi 2005, Karni 2009] in Fig.5-2. We first performed these algorithms on an example "mm" of [Igarashi 2005]. To demonstrate triangle foldovers, we set the constraint points (red stars) in an extreme configuration in Fig.5-2. Figure 5-2a and 5-2b are from [Igarashi 2005] with and without scale-adjust respectively. It appears the foldover problem was not considered in [Igarashi 2005]. Sorkine et al. in [Sorkine and Alexa 2007] proposed to apply a local/global scheme to the As-Rigid-As-Possible deformation. The shape deformation with positional constraints was first carried out. Then, the eigenvalues of Jacobian matrix were restricted in the local step for foldover avoidance. The global step was used to reconcile these triangles modified in the local

step. The usual procedure of the global step is to solve a sparse Poisson-type system with specified positional constraints. However, there exist two questions as follows,

(1) Is it true that modifying the eigenvalues of Jacobian matrix in the local step could overcome foldovers?

(2) Is it possible that the global step may introduce foldovers again?

To answer these two questions, we designed two approaches for the local step and two for the global step respectively based on the local/global scheme. The local step includes the method of eigenvalue modification and that of affine decomposition. The former is proposed in [Karni 2009]. The latter will be addressed in the following section. The global step includes the methods of solving a sparse Poisson-type system with and without the specified positional constraints. Figure 5-2c and 5-2d show the combination of the eigenvalue modification (in the local step) plus the global step with and without positional constraints respectively. As can be seen, foldovers occur due to the positional constraints used in the global step (Fig.5-2d). Additionally, due to lack of scale-adjust, the deformed parts appear too large or too small. This means the eigenvalue modification in the local step causes a large scaling distortion. To get around this deficiency, we try to utilize the affine decomposition technique here. The basic idea is to decompose a local triangle affine transform into rotation and scale-shear transformations, and then further constrain the rotation and scale-sheer transformations respectively for a natural shape deformation. Figure 5-2e and 5-2f show the combination of the affine decomposition (in the local step) plus the global step with and without positional constraints respectively. It can be observed that foldovers enter through the positional constraints in the global step

again (Fig.5-2f). Both Figure 5-2d and 5-2f show that foldovers are caused by the positional constraints and enter through the global step again.

It can also be noted that there is no foldover found in Fig.5-2c and 5-2e. Foldovers do not arise via the global step if the positional constraints are not involved. However, this is not enough to conclude that the local step can overcome foldovers, which are still seen in some cases. To illustrate this, we performed the above two local/global algorithms that do not involve the positional constraints in global step on a square triangle mesh as shown in Fig.5-3.

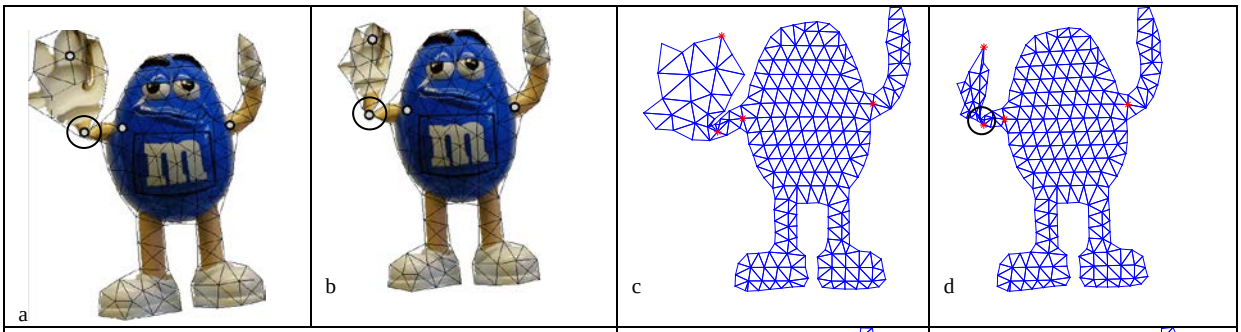
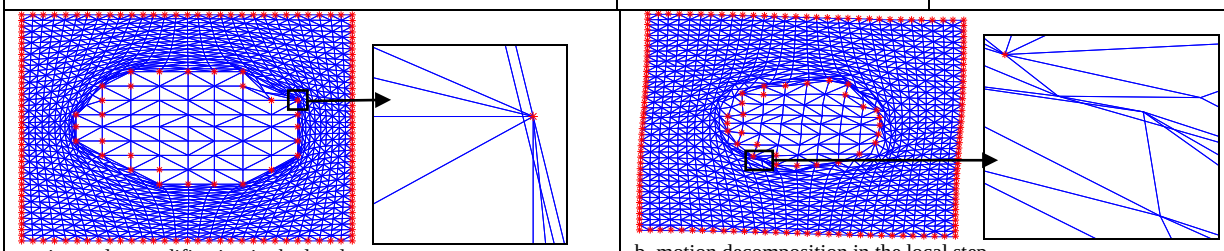


Figure 5-2. Illustration of foldovers by using recent approaches. Red stars indicate the specified constraint points. Foldovers are highlighted by circles. a) and b) are from [Igarashi et al. 2005]; c) eigenvalue modification in the local step plus the global step without constraints; d) eigenvalue modification in the local step plus the global step without constraints; e) motion decomposition in the local step plus the global step without constraints; f) motion decomposition in the local step plus the global step with constraints.



a. eigenvalue modification in the local step

b. motion decomposition in the local step

Figure 5-3. Illustration of foldovers by using the local/global schemes. The red stars indicate the specified constraint points.

Figure 5-2 and 5-3 have shown that the issue of foldover cannot be resolved by existing approaches. *An important observation* is that the previous approaches cannot enforce consistency under the composition of all local triangle deformations,

(i.e. the field that is formed by the displacement of the vertices over a mesh should be smoothed). From a mathematical point of view, the term *consistency* accounts for the properties of vector fields, i.e. integrability or smoothness. For some large deformation scenarios, Poisson-type systems in the global step is not able to average out the inconsistency between the local deformations. The inconsistency usually leads to numerical instability of Poisson systems, such as foldovers. This paper aims to tackle this challenge and further formulate a generalized solution to the foldover-free 2D mesh deformation.

5.2 PROPOSED METHOD

To deform a continuous surface is to determine a displacement field d between the surfaces before and after deformation. The deformation transformation φ is modelled as, $\varphi(X) = X + d(X)$ or $\phi^{-1}(X) = X - d(X)$, with X in domain Ω. The deformation gradient refers to the derivatives of φ called the Jacobian matrix, $\nabla\varphi$. Topology preservation in a displacement filed is to preserve the connectivity between the neighbouring structures, i.e. the resulting mapping is bijective. If foldover appears in the displacement, it indicates the resulting mapping is not one-to-one and is not invertible. An indicator of determining whether there is foldover is the determinant $J_d(X)$ of the Jacobian matrix $\nabla\varphi$ over the whole domain Ω, which can be computed by,

$$J_d(X) = \det(\nabla\phi) = \det(I + \nabla d(X)), \tag{5.1}$$

where I denotes an identity matrix. The Jacobian matrix $\nabla\varphi$ encodes the local stretching, shearing and rotating of the displacements $d(X)$. If the determinant J_d at some X is negative, this implies that the one-to-one mapping has been broken. To prevent foldovers from occurring, we need to restrict φ with the constraints of positive J_d during the deformation.

5.2.1 OUTLINE OF OUR PROPOSED METHOD

Regarding 2D discrete settings, let X and Y denote the vertices in the meshes before and after deformation respectively, and the deformation transforms defined by $Y = \varphi(X)$. For 2D mesh deformation, due to piecewise affine transformations, the deformation of each triangle can be simply defined by the corresponding edges instead of the vertices,

$$Y = \varphi(X) = AX, \tag{5.2}$$

where, A is a 2×2 matrix, $X = \left(X_2 - X_1, X_3 - X_1\right)$ and $Y = \left(Y_2 - Y_1, Y_3 - Y_1\right)$. Herein, A is namely the Jacobian matrix that is constant within a triangle. Following Eq.5.1, it is requested to restrict the deformation φ with the piecewise positive Jacobians $\det(A)$ to avoid foldovers. Figure 3 further shows that even if the Jacobian of every triangle $\det(A)$ is positive, the displacement vectors between the neighbours may still be inconsistent and can further result in foldovers. Thus, the encountering difficulties include,

(1) how to add the constraint of positive Jacobian $\det(A)$ for every triangle;

(2) how to guarantee the desired accord or consistency over the displacement field.

Our basic idea is to restrict Jacobians for every triangle and further construct a consistent displacement field by the feasible subspace method that will be addressed later. Our proposed method also adopts the local/global scheme due to its simple implementation. Essentially, this scheme is an alternating least squares iterative framework, and in particular, it has been proved to yield an optimal solution in a least squares sense [Karni 2009]. For clarity, our algorithm is outlined as follows.

Foldover-free Constrained Deformation Algorithm

Initialize: $X = X^{(0)}$;
Local Step:
1) Compute the affine for each triangle with the specified positional constraints in terms of the current $X^{(i-1)}$;
2) Add the positive Jacobian constraint to the affine for each triangle respectively;
Global Step:
1) Construct a feasible subspace with the displacement field consistency;
2) Project the inconsistent displacement field onto the feasible subspace;
3) Reconstruct the deformed x^* from the resulting displacement field;
Convergence:
If convergence is reached up to some tolerance, then end; otherwise, go to Local Step.

The above presented algorithm is actually a generalized framework. The local step can adopt various Jacobian constraints in terms of different applications. For example, Section 3.2 will introduce two methods of Jacobian constraints. In the global step, besides the interpolation approaches based on Poisson-type system that will be addressed in Section 3.3, one can adopt other interpolation techniques for deformation, e.g. radial basis functions. Herein, we hope to point out that constructing a consistent displacement field in advance is independent of the

following interpolation methods and is a crucial step to obtain a stable solution at this stage. Moreover, in the following sections we will also address the other issue, that is, the generation of super-resolution image patches, which will be applied to the development of an image and video magnifier.

5.2.2 POSITIVE JACOBIAN CONSTRAINTS

Initially, the undeformed mesh is first deformed with all positional constraints by using the approach of [Igarashi et al. 2005], harmonic map [Eck et al. 1995] or others. There might be foldovers on the resulting mesh. Then, we can determine the affine transform for every triangle accordingly. In the local/global scheme, the target mesh is updated iteratively until convergence is reached.

In the local step, we focus on the affine transform of each triangle. Usually, Jacobian constraints are carried out at that time. In terms of Eq.5.2, the Jacobian matrix $\nabla\varphi$ of the triangle T is denoted by matrix A, i.e. $A = \nabla\varphi$. In order to add the Jacobian constraint, we can simply prescribe the thresholds $(\varepsilon_{\min}, \varepsilon_{\max})$ to bound the Jacobian det(A), and then modify matrix A by solving the following inequality,

$$\begin{cases} \varepsilon_{\min} \le J_\alpha \le \varepsilon_{\max} \\ J_\alpha = \det(I + (A - I)\alpha) \end{cases}, \tag{5.3}$$

where $0 \le \alpha \le 1$. It can be noted that the Jacobian of the triangle T can be bounded in the desired interval by adjusting α. However, this processing usually results in some unexpected distortion.

Moreover, for the purpose of the As-Rigid-As-Possible deformation, matrix A can be further decomposed into a rotation component and a scale-shear one by polar decomposition as suggested in [Alexa et al. 2000],

$$A = RVDV^T, \tag{5.4}$$

where D is a diagonal matrix containing the singular values of A. These two singular values indicate the amount of the triangle stretchiness. Applying logarithm to the rotation matrix R yields the rotation angle, i.e. $\log(R)$. One thus can prescribe the thresholds to the rotation angle and singular values for the As-Rigid-As-Possible deformation.

Nevertheless, the modified Jacobian matrices A' must guarantee the Jacobians $\det(A')$ positive everywhere.

5.2.3. CONSISTENCY OF DISPLACEMENT FIELDS

Following the local step, we obtain a set of modified deformation Jacobian matrices $\{A_i'\}$, i=1…m, corresponding to the individual triangles of the mesh, where m denotes the number of triangles. The global step will make the deformation Jacobian matrices consistent and reconstruct the deformed mesh by Poisson solver. Taking Cartesian system into account, it can be noted that for triangle T_i, the two row vectors of A_i' contribute the displacements to the Cartesian components of the affine φ_i along x and y axes by,

$$\varphi_i(X) = \begin{pmatrix} f_i(X) \\ g_i(X) \end{pmatrix} = \begin{pmatrix} (A_i'(1,1), A_i'(1,2)) \cdot X \\ (A_i'(2,1), A_i'(2,2)) \cdot X \end{pmatrix}. \quad (5.5)$$

To deform the mesh, the two Cartesian components *f* and *g* of deformation φ need to be reconstructed separately from their individual corresponding gradient fields, i.e. $(A'(1,1), A'(1,2))^T$ and $(A'(2,1), A'(2,2))^T$.

For simplicity, consider the deformation of a scalar field *f*(*X*) associated with a gradient field that is denoted as $v(X) = (X_x, X_y)^T \equiv (p,q)^T$ in R^2. The given gradient fields *v*(*X*) are usually inconsistent because of the internal positional constraints and modifications of the local step. Similar to the traditional PCA technique, our basic idea is to construct a subspace of gradient fields with gradient vector consistency, and then project the original field *v*(*X*) onto it to form a feasible gradient field, which will be used to reconstruct the desired deformed scalar fields $f^*(X)$. The challenge is how to build up a feasible subspace of gradients from the given *v*(*X*).

Since the unknown $f^*(X)$ is a scalar field, the consistent gradient field *v*(*X*) should be a potential vector field and satisfy the integrability condition, i.e. $curl(v) = p_y - q_x = 0$, over the domain [Arfken and Weber 1995]. Due to the above-mentioned constraints, the given *v*(*X*) usually cannot satisfy the curl-free condition. To get round this issue, we subtract the curl components from the given *v*(*X*). The problem can then be further formulated that for a given vector field *v*(*X*), our goal is to find a vector field $v_c(X)$ whose curl captures the curl part of the original *v*(*X*). This happens to be the solution of the following minimization problem,

$$\min_{v_c} \int_{\Omega} (\nabla \times v_c - v)^2 dV . \quad (5.6)$$

In a discrete setting, solving the above minimization problem requests to overcome the irregular connectivity of meshes or polyhedral. Recent work on overcoming this difficulty is to approximate the smooth fields with discrete fields and redefine the discrete differential operators for triangulated 2D manifolds [Desbrun et al. 2002, Tong et al. 2003]. A discrete vector field on a mesh is defined as a piecewise-linear one with a constant vector within each triangle. The vector field v_c can be expressed in an affine fashion within a neighbourhood of vertex *X*, i.e. $v_c = \sum_i \phi_i(X) v_{ci}$, with ϕ_i being the piecewise-linear basis function valued 1 at vertex X_i and 0 at all other vertices and v_{ci} being the value of v_c at vertex X_i. For a given discrete vector field *v*(*X*) on a mesh, its curl at X_i is defined as,

$$(curl\ v)(X_i) = \sum_{T_k \in N(i)} (\nabla \phi_{ik} \times v_k) \left| T_k \right|, \qquad (5.7)$$

where, *N*(*i*) is the set of triangles sharing the vertex *i* within 1-ring range, T_k denotes the triangle *k* and $\left|T_k\right|$ is area of T_k, $\nabla \phi_{ik}$ is the gradient vector of ϕ_i within T_k, and v_k denotes the vector of *v*(*X*) inside the T_k. According to the definition of Eq.5.7, for each vertex X_i, one can yield the following sparse and linear system for the unknown curl part vector field v_c by a simple sum over the 1-ring neighbourhood of the underlying mesh [Tong et al. 2003],

$$\sum_{T_k \in N(i)} \nabla \phi_{ik} \times (\nabla \times v_c)_k \left| T_k \right| = \sum_{T_k \in N(i)} \nabla \phi_{ik} \times v_k \left| T_k \right|, \qquad (5.8)$$

where, $i = 1 \ldots n$, with *n* being the vertex number. Subtracting the resulting vector fields $v_c(X)$ from the original *v*(*X*) yields a consistent gradient field, i.e.

$v_g(X) = v(X) - v_c(X)$, that can further be applied to the reconstruction of the unknown $f^*(X)$.

Remark

As far as Eq.5.7 and 5.8 is concerned, we prefer to subtract the curl field $v_c(X)$ from the non-integrable filed $v(X)$ rather than to estimate the divergence field $v_g(X)$ directly. This is due to the fact that the curl part of a non-integrable field contains information about the divergence of the underlying integrable field, that is, the divergence part contains incomplete information about the divergence of the underlying integrable field. We further explain it by the following example. Suppose due to noise or unexpected disturbance, a non-integrable field $v(X)$ in R^2 is equal to the true gradient field $v_0(X)$ rotated by an angle θ. The divergence and curl of the field $v(X)$ are computed respectively as,

$$\begin{aligned} Div(v) &= (\nabla \cdot v_0)\cos\theta + (\nabla \times v_0)\sin\theta \\ &= (\nabla \cdot v_0)\cos\theta \\ Curl(v) &= -(\nabla \cdot v_0)\sin\theta \end{aligned} \quad .$$

It can be noted that the divergence part of the field $v(X)$ is only the projection of the $v(X)$ onto the divergence dimension. Partial information of the field $v_0(X)$ hides in the curl part of the field $v(X)$. Thus, we hope to keep the magnitude of the estimated divergence field $v_g(X)$ as that of the $v_0(X)$, such that,

$$\left|Div(v_g)\right| = \sqrt{Div(v)^2 + Curl(v)^2} = \left|\nabla \cdot v_0\right| .$$

To this end, we need to firstly estimate the curl field $v_c(X)$ from the given non-integrable field $v(X)$, and then subtract it from the $v(X)$. Because,

$$\begin{aligned} Curl(v) &= Curl(v_0) + Curl(v_c) \\ &= Curl(v_c) \end{aligned},$$

where, $Curl(v_0) = 0$, one can estimate the curl field $v_c(X)$ accordingly. Then $v_0(X) = v(X) - v_c(X)$. This can be viewed as an explanation of Eq.5.8.

So far, we have been able to apply the scheme of Eq.5.7 and 5.8 respectively to the Cartesian components *f* and *g* of Eq.5.5 for their individual consistent gradient fields that form feasible Jacobian matrices A_i''. However, it can be noted that there is a large difference between the resulting $\{A_i''\}$ and the original $\{A_i\}$. It is natural to seek a set of feasible Jacobian matrices as close to the original $\{A_i\}$ as possible. Here, we need to approximate the original $\{A_i\}$ based on the resulting $\{A_i''\}$, i.e., to approximate the two gradient fields of A_i, $(A_i(1,1), A_i(1,2))^T$ and $(A_i(2,1), A_i(2,2))^T$). This can be fulfilled as follows.

Consider an integrable vector field *v*(*X*) and its counterpart $v_g(X)$ with the vector consistency. We seek a feasible solution as close to the original *v*(*X*) as possible. The given $v_g(X)$ is utilized to construct a feasible subspace of gradient fields with the vector consistency. The optimal approximation is then obtained by the projection of *v*(*X*) onto the feasible subspace.

In general, each sample of $v_g(X)$ can be extended as a vector in a 2n-dimensional Hilbert space, where n denotes the sample number. Such n vectors can further span a subspace which has dimensionality not greater than n. The orthonormal basis vectors $\{u_i\}$ can be constructed by applying the Gram-Schmidt orthonormalization approach to the sample set of $v_g(X)$. Suppose that the given

field $v(X)$ belongs to the resulting subspace spanned by the resulting orthogonal basis $\{u_i\}$. We can express it as,

$$v = \sum_{i=1}^{k} \left(u_i^T v \right) u_i^T \ .$$

If the vector field $v(X)$ does not belong to this subspace, they can be approximated by,

$$\hat{v} = \sum_{i=1}^{k} \left(u_i^T v \right) u_i^T \ . \qquad (5.9)$$

The geometry meaning says that the projection $\hat{v}(X)$ of $v(X)$ onto the subspace spanned by the basis vectors $\{u_i\}$ is the closest to $v(X)$ in this subspace. Indeed, Eq.5.9 gives out a minimum norm solution in the specified gradient subspace.

Applying Eqs.5.5-5.9 to the Cartesian components f and g yields the desired deformation φ. Moreover, to reconstruct the vertices X^* of the deformed mesh, we define a quadratic error functional associated with each edge of a triangle by,

$$E = \sum_{(i,j)\in\{(1,2),(2,3),(1,3)\}} \left\| X_{ij}^* - \left(f\left(X_{ij}\right), g\left(X_{ij}\right) \right)^T \right\|^2 ,$$

where the edge of $\overline{X_i X_j}$ denotes as $X_{ij} = X_j - X_i$. Instead of a single triangle, we solve this quadratic functional on a triangulation $T = \left\{ T_{(i_1,i_2,i_3)} \right\}$ by minimizing,

$$\min_{\{X_1',\ldots,X_m'\}} \sum_{i\in T} E_i \ , \qquad (5.10)$$

where m denotes the vertex number. The solution to this minimization problem can be obtained by the linear system of normal equations of Eq.5.10. Note that we do not add any positional constraints into this linear system at this stage in order to maintain the consistency of displacement field. The specified constraints will be approximated gradually in an iterative manner. Although we have not been able to provide a strict

proof of reaching the desired positional constraints exactly by our proposed approach, our experiment results have been very positive.

In the following we will apply the above 2D mesh deformation technique to the application of an image and video magnifier for amplifying any specified ROIs within a view field. The amplified ROI patch is usually expected to preserve its original aspects with little distortion. Moreover, the outside of the ROIs should be kept as close to isometric as possible. This can be fulfilled because of the affine decomposition utilized in the local step.

5.3 APPLICATION: IMAGE/VIDEO MAGNIFIER

As an image and video magnifier, in addition to zooming in a specified ROI for high resolution, we set all vertices of the ROI patch boundary as positional constraints to preserve the relationship between the ROI and the surrounding image. We first generate a super-resolution image patch corresponding to the specified ROI and then paste it onto the amplified ROI within the low resolution image/video. This produces the amplified ROI patch without any distortion. The XSW's SR algorithm [Xiong et al. 2010] is used to generate the desired SR image patches.

The XSW's SR algorithm includes two steps. The first one is the PDE-based anisotropic regularization that needs balance artefact removal and primitive preservation. The other one is the learning-based pair matching. Unlike other SR approaches, the training dataset consists of a set of co-occurring patch pairs extracted

from example images at two different resolution levels. The final SR effect mainly relies upon the pair matching accuracy.

Different from the single-image scenario, the interframe interaction and spatio-temporal coherency are taken into account in video application. Fortunately, the PDE regularization strength used in XSW's SR algorithm can well adapt to the variable degradation between the successive frames caused by fast motion or scene switch. Moreover, primitives in the interpolated frames also need to be enhanced. This can be achieved by simply thresholding the pair matching tolerance in the XSW's SR algorithm, effectively reducing the compression artefacts. This is well suitable for various compressed videos.

Before embedding a super-resolution patch into a low resolution image/video, the specified ROI within the original view field is firstly amplified by our foldover-free constrained deformation technique presented in section 3.1. The original low resolution image is then mapped onto the deformed view field. Embedding the SR patch into the specified ROI within the deformed view field is implemented by the Poisson cloning method [Pérez et al. 2003]. Although embedding a super-resolution patch into a low resolution image/video always results in distinct transition of boundaries, the Poisson system can effectively remove them. The solution is obtained by solving the following Laplace equation with boundary conditions,

$$\Delta \tilde{f} = 0, \text{ over } \Omega, \text{ with } \tilde{f}\big|_{\partial\Omega} = \left(f^* - g \right)\big|_{\partial\Omega}, \qquad (5.11)$$

where, g and f^* denote the super-resolution and low resolution image intensities respectively. $\tilde{f}$ within Ω is a membrane interpolation of the difference $(f^* - g)$ between the super-resolution and low resolution images along the boundary $\partial\Omega$.

5.4 EXPERIMENTS AND ANALYSIS

We first perform our proposed deformation algorithm on the example “mm” in Fig.5-2 and the rectangular triangle mesh in Fig.5-3. Eq.5.4 is employed in the local step. The intermediate and final iterative results are shown in Fig.5-4.

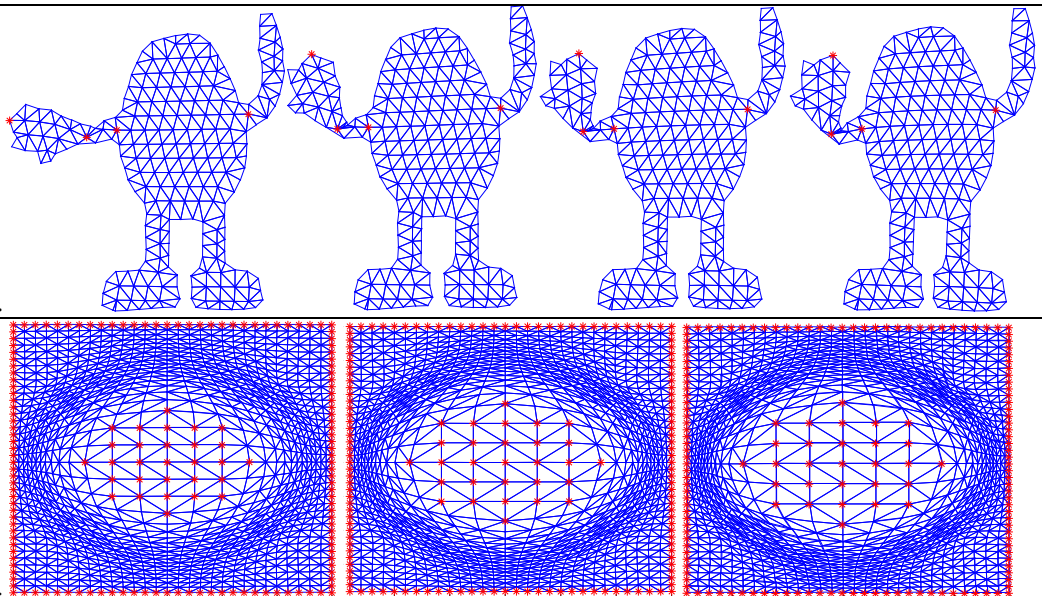


Figure 5-4. The iterative results of examples in Fig.5-2 and Fig.5-3 by our approach. Red stars indicate the positional constraints.

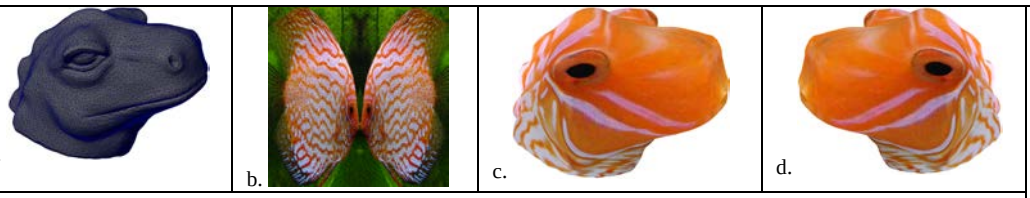


Figure 5-5. Illustration of texture mapping. a) original 3D model; b) matching 2D parameterization of 3D model with texture image; c,d) mapping fish image to 3D model.

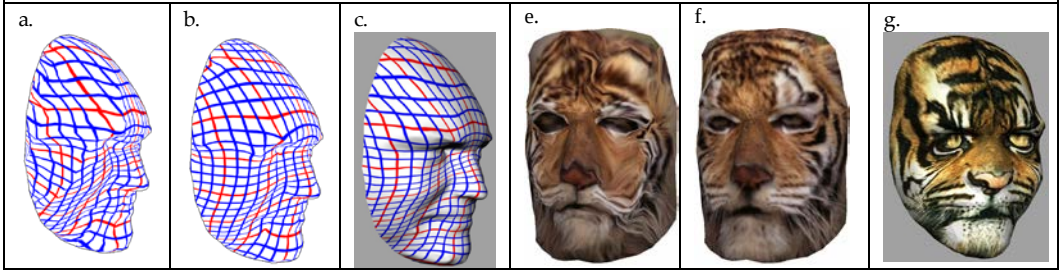


Figure 5-6. Comparison of smoothness. a and b) chessboard mapping results before and after smoothing using [Lee et al. 2008]; c) chessboard mapping result using our method; e and f) texture mapping results before and after smoothing using [Kraevoy et al. 2003]; g) texture result using our method. Figures of 6e and 6f are from [Kraevoy et al. 2003].

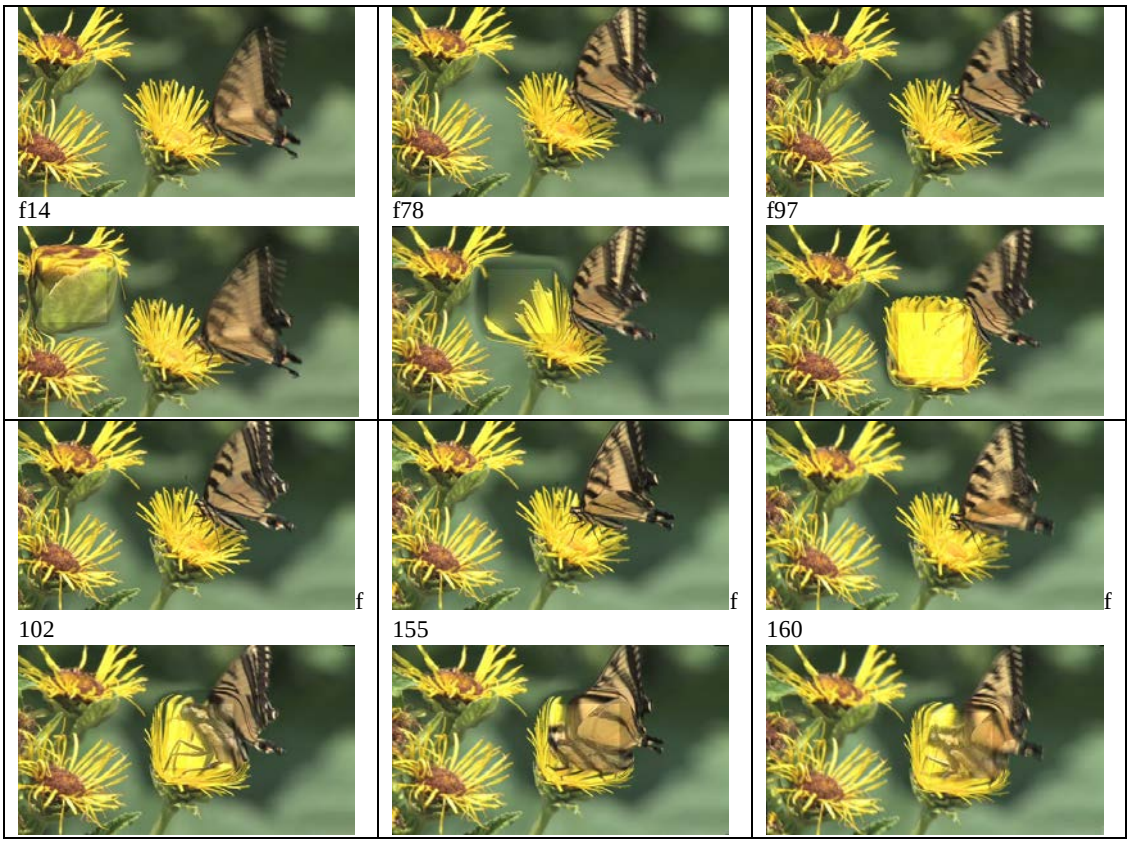
Comparing Fig.5-4a with Fig.5-2a and 5-2b, it can be noted Eq.5.4 in the local step can effectively solve the elbow-collapse problem shown in Fig.5-2a and 5-2b. Moreover, the intermediate interpolated shapes show little distortion. Our approach does not need any post-processing for reducing distortion, such as the scale-adjust presented in [Igarashi et al. 2005]. Thus, it is useful in producing 2D animation. Figure 5-4b shows that the amplified region can keep its original aspect without distortion during mesh deformation. Our approach satisfies the specified positional constraints iteratively given any tolerance.

Image warping or mesh deformation is also the basis for texture mapping. Readers are referred to [Lee et al. 2008] for a state of the art review on texture mapping. One usually first deforms the parameterization of the 3D model to match the given texture image with some specified positional constraints, and then maps the texture image to the 3D model according to the resulting (/deformed) re-parameterization. Any foldovers arisen will lead to unpleasing artefacts in the final textured model. In Figure 5-5, we perform our foldover-free constrained deformation algorithm on the parameterization of the snake head model to match the image of a tropical fish. Positional constraints are placed on the eye, nose and mouth. The mapped texture is well fit, which can be seen from the smoothly spread strips on the snake head model.

Existing texture mapping methods employ a smoothing process for reducing distortion after aligning the positional constraints. Because of the feasible subspace with the vector consistency, our method can yield a smooth re-parameterization without any post-processing. To highlight this advantage, in Fig.5-6, we compare our

deformation method with two state-of-the-art methods published in [Lee et al. 2008] and [Kraevoy et al. 2003]. Figure 5-6a, 6b and 6c show the chessboard texture mapping results by using the algorithm of [Lee et al. 2008] and our method. Figure 5-6d, 6e and 6f show the results by [Kraevoy et al. 2003] and ours. Among the three methods, ours produces the most smooth results. The experiments are undertaken using Matlab on an Intel Pentium 4 3.2GHz PC with 1 Gbyte of RAM. The running time of our method is 1.91 sec.

In the implementation of our image and video magnifier, we paste the super-resolution image patches onto the amplified ROIs within a deformed view field to avoid any potential distortion. The ROI is a square region. We performed our proposed magnifier presented in section 3.4 on a video as shown in Fig.5-7. The upscaling factor is 4 for SR patches. The SR algorithm is able to upscale an image patch sized under 50×50 around 1 second on average.



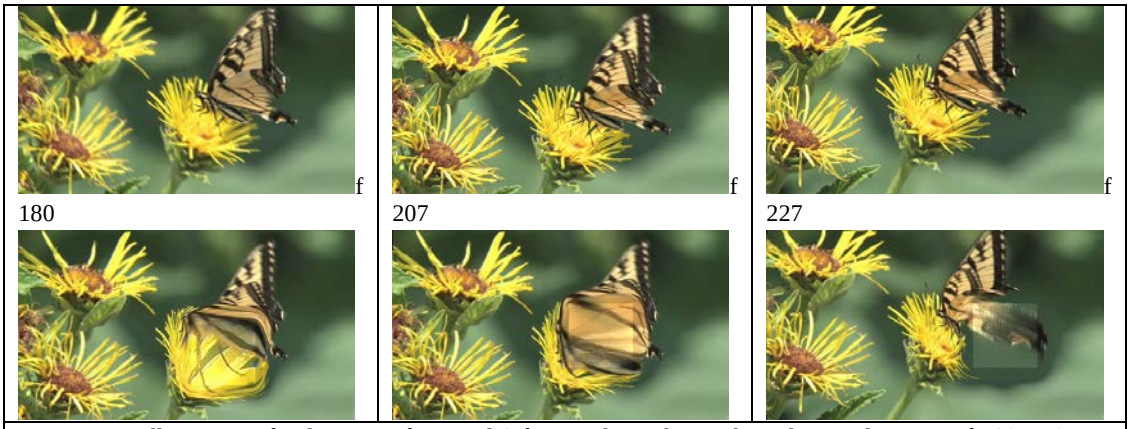


Figure 5-7. Illustration of video magnifier, total 9 frames shown here. The video resolution is of 480x270. For comparison, we put the original frames (upper) and their amplified versions (lower) together. The amplified ROIs are embedded by the super-resolution patches.

5.5 CHAPTER SUMMARIZATION

This chapter addresses the 2D shape deformation with the positional constraints and rigid constraint into consideration. Unlike the previous chapters, the rigid constraint is taken into account beside the usual positional constraints here. This is indeed to seek a tradeoff of the positional constraints and minimum distortion in terms of various applications. To highlight this issue, we further develop an image/video magnifier that is requested to both satisfy the positional constraints and cause as-rigid-as-possible deformation.

The next chapter will focuses on the rigid constraint rather than the positional constraints based on the mesh editing applications.

CHAPTER 6

WELL-SHAPED AND WELL-SPACED MESH

6.1 PROBLEM FORMULATION

This chapter addresses an application based on the surface parameterization with constraint of minimal distortion. Unlike the previous chapters, this chapter focuses on the rigid constraint instead of the positional constraints. The challenge is to generate a quality mesh after deformation so as to preserve the details with the minimal distortion. Inspired by the application of the skin sliding simulation, the goal is to develop a pre-processing procedure for low-distortion parameterization. For clarity, we address this application in details as below.

In human skin simulation, the initially created model normally has a very high quality mesh configuration, in order to present delicate details and beautiful textures. However along with the movement of the underlying skeleton, large distortions are

frequently observed on the skin surface. This greatly reduces the realistic appearance. From an anatomical perspective, the skin surface, as an elastic layer covering multiple anatomy structures, should preserve its own tension during deformation rather than over-stretches and compression that only happen in some local areas (e.g. joint). In practice, due to large-scale deformation and lack of precise control, the appearance of many skinny triangles is unavoidable [Yang et al. 2009, Zhang et al. 2011]. This further results in many visible artifacts or distortions on the surfaces. For realistic purposes, skin sliding is presented to deal with this challenging issue. Figure 6-1 illustrates the intuitional geometric meaning of skin sliding. Roughly speaking, skin sliding is a variant of remeshing for removing skinny triangles, which is resampling the 3D coordinates of vertices while preserving the mesh connection relationship. Some efforts have been made to implement this real life effect into character animation. For example, in the movie *Hellboy* [Stinson and Thuriot 2004], the system employed a spring network, where all the vertices of the skin mesh are individually connected to the muscles or the bones using spring forces. This allows skin surface “slides” above the underlying structures instead of fixing solidly to certain points on bones or muscles. However in this system, since the elasticity of the skin mesh itself was not considered, some distortions can still be seen on the skin surface. One possible solution is to replace the edges of the skin mesh with springs as well. This would incur a drastic increase in the computation burden. Moreover, tuning so many spring parameters would also be a big challenge to animators. To avoid such a complicated control system, some interpolation techniques were applied to simulate the elastic property of the skin surface instead of the spring network. The mesh before deformation is regarded as a reasonable skin mesh, while after deformation it is always largely distorted. Hence, skin sliding is applied to the mesh

after deformation for removing distortion. To improve the reality, it is further proposed to,

- Keeping the geometry and topology attributes of skin mesh as close as possible to the mesh before deformation, which is assumed to have the desired features of the skin surface. The geometry attributes of a mesh refer to the relative positions between vertices, while the topology attributes refer to the connection relationship of vertices. It is desired to transfer the attributes of skin mesh onto the mesh after deformation;

- Increasing smoothness. Intuitively, it is always desired to generate a smooth "skin surface";

- Keeping the skin mesh as close as possible to the shape after deformation, which is assumed to be the desired shape created by animators. Usually, animators pay attention to the global shape of the mesh after deformation rather than the local skinny triangles, since the shape results from underlying anatomical structures. It is expected that the skin surface at least overlays multiple layers of the underlying anatomical structures without any unexpected bulge or shrinkage.

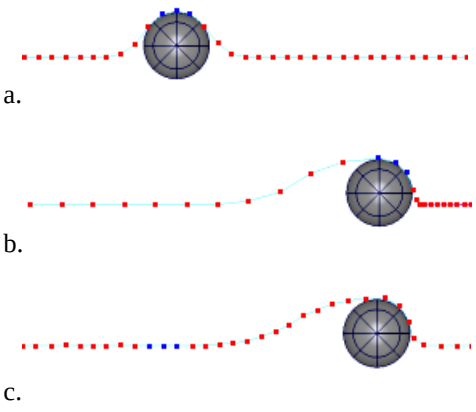


Figure 6-1. Schematic diagram of the effect of skin slide over an underlying structure. (a) The three blue points denote points on the skin surface over the underlying anatomical structure. (b) Incorrect sliding, the blue points shift along with the underlying structure, and the skin is stretched on one side and compressed on the other side. (c) Correct sliding, the blue points remain static thus making the skin appear to slide over the internal structure. (from [Yang et al. 2009])

To this end, [Yang et al. 2009] applied a linear interpolation method to skin sliding simulation based on 2D parameterizations. Because of linear interpolation, the computation is very simple. However, the deficiencies are distinct. The main challenge is that there is no guarantee of preserving the shape and feature of the original surface. There is a tradeoff between the computational complexity and realism on this issue. It is inevitable that linear interpolation incurs a big loss of realism. These issues will be further addressed later.

In pursuit of a realistic appearance, we focus on two aspects of skin sliding in terms of the above-mentioned goals, that is, partitioning the meshes and preserving the details of the mesh before deformation. This is due to the following observation. Skin sliding is always performed on a set of distorted patches instead of the whole mesh domain. Hence, removing distortion from a skin mesh implicitly demands two procedures, one is to partition meshes into a set of patches (e.g. distorted and non-distorted patches) for further resampling, and the other is to preserve the original features or details of the skin mesh. We further reformulate the skin sliding implementation by introducing the graph Laplacian framework [Sorkine 2006]. The main contributions include, partitioning a whole mesh into a set of patches, transferring and preserving features and shape without any unexpected bulge and shrinkage, which will be addressed later. Moreover, our proposed method can be integrated into the developed animation pipeline [Yang et al. 2009] without any interference with the traditional methods of skinning, giving the animator the freedom to control and design actively during the skinning phase.

6.2 METHOD OVERVIEW

We first recap the skin sliding method presented in [Yang et al. 2009] briefly, and then point out some existing challenges. After that, our proposed algorithm will be outlined.

In [Yang et al. 2009], skin sliding method is usually applied to a specified region of interest (ROI) instead of the whole mesh domain. The input is a pair of patches M_0 and M_d before and after deformation. The correspondence between them is established by embedding M_0 and M_d into a common 2D domain. In terms of the resulting embedding maps U_0 and U_d, one can find out the facets of U_d where the vertices of U_0 fall in, that is, the correspondence between the facets of U_d and vertices of U_0. Moreover, the vertices of M_d are updated by the vertices interpolated in the facets of M_d accordingly. The resulting patch is finally embedded back into the deformed mesh domain.

It is worth noting that these two patches M_0 and M_d before and after deformation are matched by overlapping the 2D embeddings U_0 and U_d. Intuitively, this is to build up a sampling mesh over the patch M_d. As a result, the resampling is fulfilled by a linear interpolation accordingly. A simple assumption arises here, that is, the vertices of M_d moving to their updated locations can be mirrored in the embeddings U_d and U_0, just like the effect of the vertices moving across the surface. This indeed risks a big distortion. All of the existing 2D embedding approaches inevitably introduce large distortion, such as the cotangent weighted Laplacian coordinates [Desbrun et al. 1999], bar-net embedding [Yang et al. 2009] and Mean Value

Coordinates [Floater 1997]. The resulting sampling mesh based on 2D embeddings might lead to the worst case that sparse sampling is performed on a high-frequency region while dense sampling on a low-frequency region. To deal with this challenge, the skin sliding method proposed in [Yang et al. 2009] was performed on small patches that were manually cropped in advance. This is because a small patch is easily flattened with little distortion. Moreover, skin deformation always appears at some local regions rather than the whole surface. Desirably, skin sliding methods should be performed on the skinning regions instead of the whole surface. The challenging issues raised here are how to cut the surface into a set of patches and how to remove the visible seams on the reconstructed surface. Our first contribution in this chapter is to present an approach for automatically detecting and segmenting skinning regions of a skin surface.

Another point worth noting is that interpolation takes place on the facets of the patch M_d rather than others. This guarantees that the interpolated mesh complies with the deformed patch M_d. However, the deficiency is also clear, that is, there is no guarantee of transferring the details of M_0 to the target skin mesh. Moreover, because of linear interpolation, the noticeable drawback is that if two adjacent patches are resampled individually and then embedded back to the original deformed mesh domain, there are visible seams across the patch boundaries on the reconstructed surface. Additionally, there is the visible shrinkage of the skin mesh, since the over-stretch of the local triangles misses out some prominent parts. Our second contribution is to overcome these challenges in this chapter.

Compared to the previous approaches, our proposed algorithm reformulates the skin sliding implementation based on the graph Laplacian framework. The distinct advantage is that this framework covers two important applications, spectral

clustering [Shi and Malik 2000] and differential coordinates [Sorkine 2006]. The former has been applied to machine learning and computer vision fields. Recent work further demonstrates that it can be applied to mesh parameterization [Mullen et al. 2008], segmentation [Liu and Zhang 2007] and compression [Karni and Gotsman 2000] as well. The latter can be viewed as a combinatorial analog of Laplace-Beltrami operator defined on a manifold. It can effectively approximate the differential properties of a smooth surface and further results in the desired detail-preserving manipulations. Our skin sliding algorithm is outlined as follows.

Graph Laplacian based skin sliding algorithm

(1) Partitioning a given pair of meshes by Normalized cut algorithm;

(2) Smoothing specified patch pairs;

(3) Resampling by Laplacian coordinates;

(4) Embedding the resampled patches back to the original mesh domain for the reconstructed surface.

The proposed algorithm can provide a global, shape and detail-preserving solution. In the following sections, we will further address the implementation of each step.

6.3 PARTITION BY NORMALIZED CUTS

To deform a skin surface, the following can be applied over the whole mesh domain: stretching, compression and non-deformation. In terms of the given mesh pair before and after deformation, the goal of the surface partition step is to identify

these three types of Regions Of Interest (ROIs). It is always expected that the partitioned patch boundaries can maintain more natural shapes with genus-0, such as quadrangle, triangle and circle-like boundaries, in order to reduce distortion in the following step. This is because the following step will embed the cropped patches into a common 2D parametric domain for further resampling. Our algorithm utilizes one of the spectral clustering approaches, i.e. normalized cut algorithm [Shi and Malik 2000], to partition the given meshes. Although it has been applied in mesh segmentation [Liu and Zhang 2007], we further extend it to smooth the resulting cluster boundaries by using the min-cut/max-flow algorithm [Boykov and Kolmogorov 2004] here. The starting point is the graph Laplacian matrix, which is application-dependent. In the partition step, we construct the graph Laplacian as follows.

Let M_0 and M_d be the meshes before and after deformation with the definition as $M = \{V,E,F\}$, i.e. vertex set denotes $V = \{v_i \,|\, i = 1,...,n\}$, edge set denotes $E = \{e_i = (v_{i_1}, v_{i_2}) \,|\, i = 1,...,m\}$ and facets set denotes $F = \{ f_i = (v_{i_1}, v_{i_2}, v_{i_3}) \,|\, i = 1,...,h\}$. Our goal is to partition the mesh into a set of patches in terms of the three categories, stretch, compression and non-deformation. Consider a dual graph $G=\{F,E\}$ with the facets as nodes. For each pair of the corresponding triangles $f_i^{(0)}$, $f_i^{(d)}$ from M_0 and M_d, we can separately add the 4th vertex to form a pair of corresponding tetrahedrons as used in [Sumner and Popović 2004] and then compute the affine transformation A_i between them. Further consider the following identity,

$$\det(A_i) = Vol_d / Vol_0 ,$$

where *Vol* denotes the volume of the f_i. The volume ratio has the three possible values, i.e. greater than 1, equal to 1 and less than 1, corresponding to the three deformations of the f_i respectively, i.e. stretch, non-deformation and compression. We employ the volume ratio as the attribute of the f_i here. Moreover, for a pair of the adjacent f_i and f_j, we define the distance between them as $dist(f_i, f_j) = \det(A_i) - \det(A_j)$. The edge connecting the f_i and f_j in the dual graph *G* is further weighted as,

$$w_{ij} = \exp\left(-\frac{dist(f_i, f_j)^2}{\sigma^2}\right), \qquad (6.1)$$

otherwise $w_{ij} = 0$ for the nonadjacent facets. The kernel width σ is the standard deviation of pairwise distance between facets. It does not obviously influence the partition results, as long as it is not too small.

In terms of such weighted adjacency matrix, we can further construct the graph Laplacian of *G* as,

$$L = D - W, \qquad (6.2)$$

where *D* is a diagonal matrix with the degrees, $d_i = \sum_{i=1}^{n} w_{ij}$ on the diagonal.

Moreover, we apply the normalized cuts to the graph Laplacian of Eq.6.2 for partition purposes. Herein, the resulting graph Laplacian *L* is very sparse, large and symmetric. To exploit these properties of our eignsystem, Lanczos algorithm [Golub and Van 1996] is thus employed.

Note that each eigenvector **x** may take on *h* real values, and the elements of **x** individually correspond to the facets of the mesh. Consequently, *h* facets can be

mapped into the subspace (also called eigenspace) spanned by the t eigenvectors associated with the first t smallest eignevalues. This mapping is regarded as a t-D embedding of the vertex set. Accordingly, the normalized cut algorithm iteratively approximates the desired solution by thresholding the resulting t-D embedding. Although the weighted adjacency matrix W is employed in Eq.6.2, the t eigenvectors can smoothly distribute over the eigenspace rather than the distinct gaps inbetween them. At each iteration, we have to choose a splitting point to bipartition the given facet set into two parts in our implementation. To search for such splitting point, the normalized cut employs the following metric,

$$\begin{cases} Ncut(P_1,...,P_k) = \sum_{i=1}^{k} \dfrac{cut(P_i,\bar{P}_i)}{assoc(P_i)} \\ cut(P,Q) = \sum_{i\in P, j\in Q} w_{ij} \\ assoc(P) = \sum_{i\in P} d_i \end{cases}, \qquad (6.3)$$

where $\bar{P}$ is the complement of the set P and k denotes the number of means (Refer to [Shi and Malik 2000] for details). The terminating criterion here is defined by measuring the cluster divergence in the eigenspace. To this end, we first construct the covariance matrices for the individual clusters in the current eigenspace, and then carry out the SVD decomposition on them to obtain their individual ratios between the minimum and maximum singular values. The greater the cluster divergence, the smaller the ratio will be. In our experiments, we found that simple thresholding on the ratio can effectively alleviate the over-segmentation. We set that threshold value at 0.25 in our experiments. In addition, post-processing is still needed to deal with the resulting fragments.

However, applying the Normalized cuts algorithm to the graph Laplacian L yields a coarse partition, since the cluster boundaries tend to be jagged. These irregular

boundaries usually incur large distortion in the subsequent 2D parameterization procedure. Further refinement is thus necessary to identify accurate and sound boundaries. To this end, we employ the min-cut/max-flow algorithm [Boykov and Kolmogorov 2004] here, that is, to find a cut that has the minimum cost among all cuts based on a graph. Since our goal is to refine the boundaries of the resulting clusters, we can define a narrow band enveloping the boundaries and perform the min-cut/max-flow algorithm on it instead of the whole mesh domain. In our implementation, the neighborhood of the boundaries typically covers the 4 rings of the vertices on the boundaries. We thus construct the desired dual graph in terms of the selected narrow band. In the context of the graph cut based minimization techniques, the min-cut/max-flow algorithm essentially solves the shortest path problem on a weighted graph. The edge weights need to involve the cue of the edge length. Accordingly, the edge weight between the facets f_i and f_j is re-defined as,

$$w'_{ij} = \left|e_{ij}\right| \exp\left(-\frac{dist(f_i, f_j)^2}{\sigma^2}\right), \qquad (6.4)$$

where e_{ij} denotes the edge shared by the facets f_i and f_j in the mesh M_0. As a result, this refinement makes the boundaries smoother. One can compare the effects of the partition procedure before and after refinement processing in the following experiment. Our partition procedure is summarized as follows,

Partition Procedure

(1) Given a pair of meshes, set up a weighted adjacency matrix W and degree diagonal matrix D;

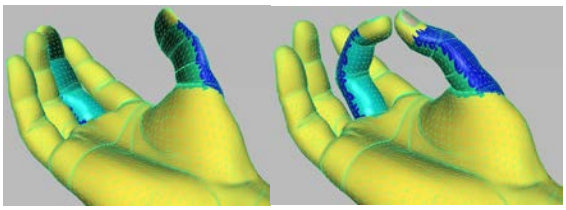
(2) Solve the generalized eigensystem $L\mathbf{x} = \lambda D\mathbf{x}$ for eigenvectors $\mathbf{x}$ with the smallest eigenvalues λ;

(3) Use the t eigenvectors with the first t smallest eigenvalues to bipartition the meshes, e.g. k-means clustering algorithm;

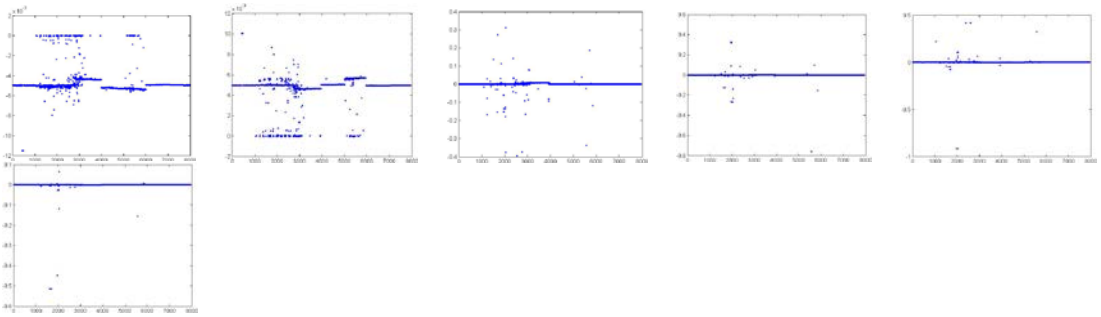
(4) Recursively re-partition the segmented parts if necessary;

(5) Employ min-cut/max-flow algorithm to refine the partition boundaries.

a.



b.



c.

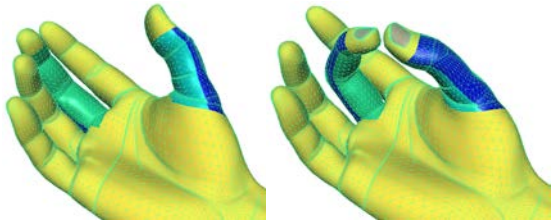


Figure 6-2. The partition result of a hand skin surface. (a) shows the partition results of the fingers before and after deformation without refinement; (b) plots the eigenvectors corresponding the 2nd smallest eigenvalue to the 7th smallest one; (c) shows the refinement results.

Figure 6-2 shows the partition results of a hand model that contains 7997 vertices and 15855 triangular facets. In the original normalized cut algorithm, only the eigenvector with the 2nd smallest eigenvalue is used, whereas the next few smallest eigenvectors indeed contain useful partition cues as well. This can be observed in the plots of the 2nd, 3rd and 4th smallest eigenvectors in Fig.6-2c. The other eigenvectors (i.e. 5th, 6th and 7th smallest eigenvectors in Fig.6-2c) almost have the same distribution. Obviously, it is enough to span the eigenspace by using the 2nd, 3rd and 4th eigenvectors here. Additionally, it can be noted that the skinning regions (i.e.

stretch and compression ROIs) appear around joints while others belong to non-deformation regions. The stretch (compression) ROIs maintain quadrangle-like boundaries in Fig.6-2c. In skin surface deformation, we pay attention to stretch ROIs rather than others. The compression ROIs usually come along with self-intersection. From the perspective of an animator, the outward appearance of a deformed model receives more attention than inside.

6.4 SMOOTHING AND RESAMPLING

Our basic idea consists of: (1) firstly generating the base of the original mesh, (2) then extracting the details accordingly, (3) further transferring them to a specified surface. For clarity, we first address the Laplacian Coordinates briefly and then present our smoothing and resampling steps separately.

6.4.1 DIFFERENTIAL SURFACE REPRESENTATION

The graph Laplacian has long been viewed as a combinatorial version of the Laplace-Beltrami operator and applied to geometry processing. Unlike the spectral clustering, it works directly on a mesh rather than a dual graph of the mesh. For completeness, we first address how to represent a surface by changing the weights of the graph Laplacian matrix Eq.6.2, i.e. Laplacian Coordinates, and then return to the smoothing and resampling procedures. Consider a triangle mesh $M = \{V,E,F\}$. The

graph Laplacian of Eq.6.2 is viewed as a discretization of the continuous Laplace-Beltrami operator as follows,

$$L = I - D^{-1}W\,, \tag{6.5}$$

where I denotes an identity matrix. Usually, we simply assume that the mesh M is a piecewise-linear approximation of a smooth surface. The Laplacian operator at vertex v_i is written as,

$$L(v_i) = \sum_{j \in N(i)} w_{ij} \left(v_j - v_i \right), \tag{6.6}$$

where $N(i)$ denotes the 1-ring neighborhood of the vertex v_i, and w_{ij} is the weight of edge e_{ij} subject to $\sum_{j \in N(i)} w_{ij} = 1$. The basic geometric meaning of Laplacian operator is that the relative location of vertices is encoded in δ-coordinates (or called Laplacian coordinates) as follows,

$$\delta_i = L(v_i)\,. \tag{6.7}$$

Intuitively, the details of the mesh M are preserved in δ-coordinates. To perform surface modeling by using Laplacian coordinates, one can fix the absolute position of several vertices as Dirichlet boundary conditions and further solve a sparse linear system, $L\mathbf{v} = \boldsymbol{\delta}$, derived from Eqs.6.5 and 6.7 for restoring a global solution $\mathbf{v}$ of Cartesian coordinates of the vertices.

6.4.2 SMOOTHING STEP

The cropped patches need to be smoothed. The smoothed version of a patch is viewed as its base so that the details of the patch can be represented by the offsets. An efficient smoothing approach is the curvature flow smoothing algorithm proposed in [Desbrun et al. 1999]. The distinct advantage is speed. But there is also an obvious deficiency that the curvature flow may shrink the volume of a mesh substantially. Moreover, for irregular connectivity meshes, there is a lack of sufficient control over global behaviour during smoothing, such as large inaccuracies for irregular meshes. This implies that the sharp features will be smoothed out firstly by the curvature flow. In our skinning applications, large deformation always yields many skinny triangles, and hence the smoothing step is expected to generate a well spaced and well shaped mesh. For all the above-mentioned purposes we would like, we present a smoothing method based on the tangent plan of a surface in this chapter.

From a differential geometry perspective, Laplacian coordinates of Eq.6.5 can be viewed as the discrete version of the mean curvature normal of a surface by using cotangent weights instead of uniform ones [Meyer et al. 2003]. The curvature flow drives the surface shrink or expansion along the normal direction of the surface. In essence, the curvature flow reduces curvatures and ends up removing them everywhere. Inspired by the anisotropic diffusion techniques in image processing, our basic idea is to smooth the surface along some tangent direction of the surface rather than the normal ones here. This can preserve the underlying structures of the surface as much as possible during smoothing. Our method exploits the tangent planes and restricts the updated vertices within the tangent planes.

Consider the vertex v_i of a given mesh *M*. The normal of the tangent plane at v_i can be determined by covariance matrix as follows. Let $C_i = \sum_{j \in N(i)} \left(v_j - v_i\right)\left(v_j - v_i\right)^T$, where $N(i)$ denotes the 1-ring neighborhood of v_i. Taking SVD on the real and symmetric matrix C_i yields three orthogonal bases of the local frame at v_i. The eigenvector associated with the minimum singular value is namely the normal N_i of the tangent plane at v_i.

Once the normal N_i at v_i available, the tangent plane is determined accordingly. Vertex v_i will be updated within it. The neighbours in $N(i)$ are projected onto the tangent plane. Our goal here is to determine a proper location for updating v_i on the tangent plane in terms of these projections. It is clear that v_i will move within the tangent plane instead of the normal direction of the mesh.

Choosing a proper location for updating v_i should tend to alleviate the issue of skinny triangles and improve the triangle quality as much as possible. To this end, we weight the projections $\hat{v}_j$ of the neighbors $v_j, j \in N(i)$ of the v_i by the areas of the adjacent triangles instead of the centroid of the projections $\hat{v}_j$. This is illustrated in Fig.6-3a.

For each projection $\hat{v}_j$, there exist two adjacent triangles $\Delta(v_{j-1}v_jv_i)$ and $\Delta(v_jv_{j+1}v_i)$ sharing the edge $\hat{v}_jv_i$ on the tangent plane. We weight the $\hat{v}_j$ by,

$$w_j = \frac{A_{j-1} + A_j}{2\sum_{j \in N(i)} A_j}, \tag{6.8}$$

where A_j denotes the area of the *j*th adjacent triangle. As a result, the v_i is updated within the tangent plane by,

$$v_i^{new} = v_i + \lambda \sum_{j \in N(i)} w_j \left(v_j - v_i \right). \qquad (6.9)$$

where λ denotes the iterative step length. Moreover, for the entire mesh *M*, Eq.6.9 yields a highly nonlinear system. To deal with such nonlinear optimization problem, we utilize a Gauss-Seidel style iteration to smooth the mesh with a series of local optimization steps instead of solving this problem globally. The proposed smoothing procedure is summarized as follows.

Tangent Plane Based Smoothing Method

Input: the current mesh *M*;
Repeat
(1) Choose a vertex v_i and compute its normal vector N_i by the 1-ring neighborhood;
(2) Project all the neighbors onto the tangent plane of v_i;
(3) Compute the weights and update the v_i by Eq.6.9 within the tangent plane;
Until numerical convergence.

Figure 6-3c to 3d show the smoothing results achieved using our tangent plane based smoothing method and the curvature flow algorithm respectively. It is obvious that the proposed smoothing method can effectively preserve the sharp features and areas with large curvatures compared to the curvature flow algorithm. Moreover, it can also be observed that our method can alleviate the issue of skinny triangles and effectively improve the mesh quality. However, the proposed tangent plane based smoothing method essentially minimizes the curvature of a surface in a subspace of

the tangent space. Much iteration may still degrade the order of accuracy. In addition, the stability criterion requires the iterative step length of Eq.6.9 $\lambda<1$. If this criterion is not satisfied, ripples appear and further result in serious oscillations of growing magnitude over the entire surface. In our implementation, let $\lambda<0.25$.

Before proceeding further, we assume that the partition step has cut off a pair of ROI patches respectively from the skin meshes before and after deformation, and the smoothing step has further produced the smoothed versions of these patches. We assume that these two patches share the same connectivity. Hence the correspondence is readily given. To define the mapping between these two patches, we apply the 2D embedding to them as described in [Yang et al. 2009]. Since the cropped patches usually maintain the quadrangle-like boundaries from the partition procedure, we identify corresponding boundary vertices and fix them at the same boundaries of a unit square domain in our implementation. However, we also have to point out a special case, i.e. surface twist. Although it seldom appears in skin deformations, we still encounter this challenge, for example, turning the head. It can be observed that the stretched and compressed triangles mix together in the region of neck. Thus, such skinning region is viewed as an entire ROI patch instead of as stretch and compression ROI patches individually. Additionally, to speed up the correspondence between the 2D embeddings, we apply the approximate nearest neighbour (ANN) search [Merkwirth et al. 2009] to the 2D embeddings here.

Remark

We hope to point out that the distinct advantages of the partition and smoothing steps, that is, smooth partition boundaries and well-shaped and well-sapced meshes. In the following 2D parameterization, smooth boundaries can reduce distortion

compared to the jagged boundaries. Well-shaped and well-sapced meshes can further yield a plausible correspondence by overlapping the two parameterizations.

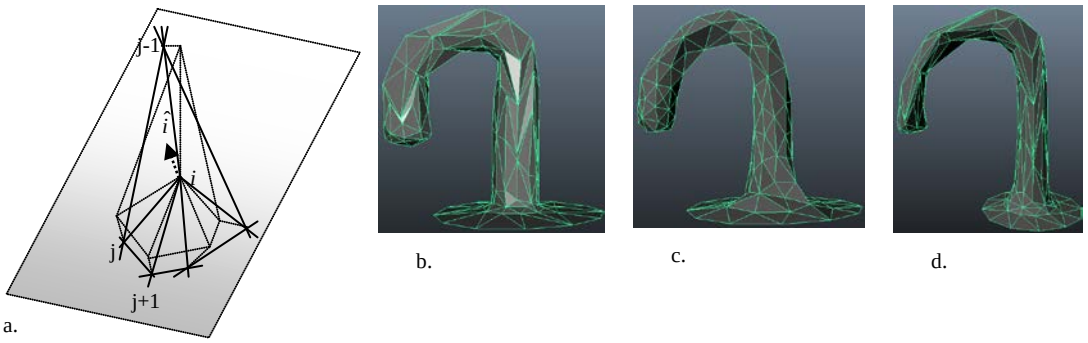


Figure 6-3. Comparison of mesh smoothing. a) The dashed lines show the projections of 1-ring neighbourhood on the tangent plane at the *i*th vertex; b) shows the original bent tube model; c) shows the smoothing result produced by our proposed smoothing method; d) shows the smoothing result produced by the curvature flow algorithm.

6.4.3 RESAMPLING STEP

Our goal is to resample the coordinates of the vertices based on the above-mentioned mapping between the cropped ROI patches. To this end, Eq.6.7 is employed for extracting and transferring the details of the skin surface. However, the challenging issue is that δ-coordinates are usually not invariant to rotation and scaling. Indeed, these differential coordinates are sensitive to linear transformations [Sorkine 2006]. Roughly speaking, Laplacian coordinates are encoded in a global coordinate system while deformations are always local and non-rigid. For example, if a surface were bent, preserving the δ-coordinates in their original orientation with respect to the global coordinate system would lead to the distorted orientation of the local details. Thus, δ-coordinates need to be properly reoriented for deformation purposes as shown in Fig.6-4a. This seems to be a typical “chicken and egg”

problem. To reorient these δ-coordinates, we need to know the geometry of the unknown deformed model and vice versa. We attempt to solve this challenge through two matched ROI patches.

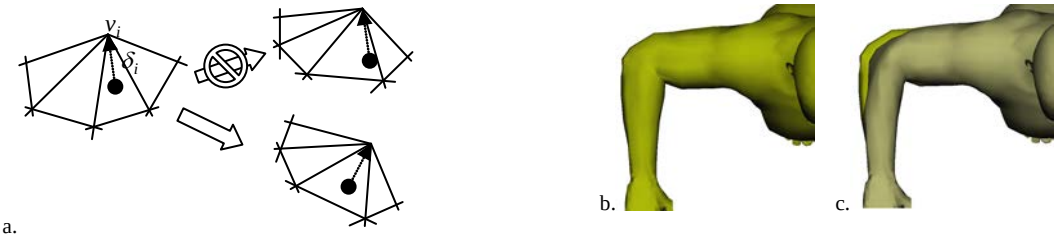


Figure 6-4. Illustration of reorienting δ-coordinates. a) During deformation, δ-coordinate is reoriented within 1-ring neighborhood of v_i; b) shows the original M_d; c) shows the shrinkage of overlapping M_d and M_d'.

Reorienting δ-Coordinates

It is natural to apply an appropriate local transformation to each vertex for the rotation-invariant representation of the details similar to [Sorkine et al. 2004]. Nevertheless, the main difference here is that in our skin sliding application, the local transformation for the individual vertices can be obtained explicitly through the two matched patches rather than an implicit optimization procedure as used in [Sorkine et al. 2004]. For clarity, we consider the rotation and scaling of the local transformation separately here.

Let $\tilde{M}_0$ and $\tilde{M}_d$ be the smoothed versions of the given ROI patches M_0 and M_d respectively that have been well matched in advance. The local details of meshes are encoded into the δ-coordinates by Eq.6.7, i.e. $\delta_i = L(v_i), i = 1,...,n$. In terms of the mapping between the smoothed versions $\tilde{M}_0$ and $\tilde{M}_d$, a vertex in $\tilde{M}_0$ can be mapped to some arbitrary point inside a triangular facet in $\tilde{M}_d$. This is indeed a resampling point in $\tilde{M}_d$ that we can determine by using the Barycentric coordinates,

that is, computing the Barycnetric coordinates in 2D embeddings and applying them to 3D surface accordingly. We can therefore set up a resampling mesh over the entire $\tilde{M}_d$ and further replace $\tilde{M}_d$ with this resampling mesh (that still denotes it as $\tilde{M}_d$ thereafter). Moreover, for each pair of corresponding vertices between $\tilde{M}_0$ and $\tilde{M}_d$, we can define their individual affine transformations in terms of the 1-ring neighborhood.

Usually, the polar decomposition is employed here for shear and rotation decomposition [Alexa et al. 2000]. However, this decomposition may yield a reflection matrix instead of rotation. To overcome this drawback, we develop a rotation-scale-shear decomposition as below, i.e. to decompose the affine A_i of the *i*th pair of the corresponding vertices into the shear Sh_i, scaling Sc_i and rotation R_i matrices $A_i = Sh_i Sc_i R_i$.

Without loss of generality, let *A* be a given affine transformation of size 3-by-3. We will decompose it into an orthogonal matrix *R* and an upper triangle matrix *U*, and hence denote the row vectors of *R* and *A* as $R = (r_1, r_2, r_3)^T$ and $A = (a_1, a_2, a_3)^T$ respectively. In terms of the Gram-Schmidt orthonormalization, we have,

$$\begin{cases} r_3 = \dfrac{1}{\|a_3'\|} a_3 \\ r_2 = -\dfrac{\langle a_2, r_3 \rangle}{\|a_2'\|} r_3 + \dfrac{1}{\|a_2'\|} a_2 \\ r_1 = -\dfrac{\langle a_1, r_3 \rangle}{\|a_1'\|} r_3 - \dfrac{\langle a_1, r_2 \rangle}{\|a_1'\|} r_2 + \dfrac{1}{\|a_1'\|} a_1 \end{cases}, \qquad (6.10)$$

where $\langle , \rangle$ denotes the inner product, and the row vectors of *A* are orthonormalized by,

$$\begin{cases} a_3' = a_3 \\ a_2' = a_2 - \langle a_2, r_3 \rangle r_3 \\ a_1' = a_1 - \langle a_1, r_3 \rangle r_3 - \langle a_1, r_2 \rangle r_2 \end{cases}.$$

We can yield the orthogonal matrix R accordingly. Moreover, we further construct the upper triangle matrix U as follows,

$$U = \begin{pmatrix} \|a_1'\| & \langle a_1, r_2 \rangle & \langle a_1, r_3 \rangle \\ & \|a_2'\| & \langle a_2, r_3 \rangle \\ & & \|a_3'\| \end{pmatrix}. \tag{6.11}$$

It can be observed that the diagonal items are positive here. U is therefore positive definite. We define the scaling transform as a diagonal matrix, i.e.

$$Sc = \begin{pmatrix} \|a_1'\| & & \\ & \|a_2'\| & \\ & & \|a_3'\| \end{pmatrix}, \tag{6.12}$$

and the shear transform as an upper triangle matrix, i.e.

$$Sh = \begin{pmatrix} 1 & \langle a_1, r_2 \rangle & \langle a_1, r_3 \rangle \\ & 1 & \langle a_2, r_3 \rangle \\ & & 1 \end{pmatrix}. \tag{6.13}$$

As a result, factoring A yields,

$A = Sh \cdot Sc \cdot R$. End of the proof.

We refer to it as the rotation-scale-shear decomposition. Compared to the polar decomposition, the proposed rotation-scale-shear decomposition can explicitly give out the rotation, scaling and shear components of a given affine transformation. Note that multiplying the shear with the scaling yields an upper triangle matrix that is positive definite. For the resulting R_i, there is no guarantee that R_i is definitely a rotation. Taking the Eign-Value Decomposition on R_i, it can be noted that the eigenvalues are of $e^{-i\theta}, e^{i\theta}, +1/-1$ respectively. Obviously, when the eigenvalues

involve -1, R_i is a reflection matrix. Hence, we can utilize a trick to deal with this issue here. If R_i is a reflection matrix, let $B_i = \begin{pmatrix} 1 & & \\ & 1 & \\ & & -1 \end{pmatrix} A_i$ (assume that the 3rd eigenvalue is of -1 here). The resulting B_i is positive definite. Applying our proposed rotation-scale-shear decomposition to B_i updates the shear Sh_i, scaling Sc_i and rotation R_i. Then, we can further yield,

$$A_i = \begin{pmatrix} 1 & & \\ & 1 & \\ & & -1 \end{pmatrix} Sh_i Sc_i R_i . \qquad (6.14)$$

This guarantees that the updated R_i is a rotation matrix rather than others. The scaling matrix might be negative definite. In general, the scaling Sc_i is of a diagonal matrix and the shear Sh_i satisfies $\det(Sh_i) = 1$. Thus, the shear Sh_i can be omitted here.

Moreover, to compare the influence of rotation and scaling, we first consider the rotation R_i here, i.e. to apply the resulting R_i to the δ-coordinates. Applying Eq.6.7 to the smoothed and unsmoothed versions of the patch M_0 respectively yields their individual δ-coordinates, $\tilde{\delta}_i$ and δ_i. Moreover, the details of M_0 can be obtained by their difference, i.e. $\xi_i = \delta_i - \tilde{\delta}_i$. To transfer the details to the smoothed version of the patch M_d, we can update the δ-coordinates of M_d by,

$$\delta_i^{(d)} = \tilde{\delta}_i^{(d)} + R_i \xi_i , \qquad (6.15)$$

where $\tilde{\delta}_i^{(d)}$ is the δ-coordinates of the smoothed version $\tilde{M}_d$. Obviously, the reoriented $\delta_i^{(d)}$ can be computed explicitly and individually here.

We can further solve the resampling vertices v_i' in M_d by minimizing the following error function,

$$E(V') = \sum_{i=1}^{n} \left\| \delta_i^{(d)} - L(v_i') \right\|^2 + \sum_{j \in N(c)} \left\| v_j' - v_j^c \right\|^2 , \qquad (6.16)$$

where V' denotes the set of the unknown vertices v_i' , v^c denote the given constrained vertices, and $N(c)$ denotes the set of the constrained vertices. The constrained vertices refer to the specified boundary of the patch M_d here. The resulting patch denotes as $M_d' = \{V', E, F\}$, that preserves the original topology. Fitting the Laplacian coordinates of the unknown geometry V' to the given δ-coordinates yields an over-determined and sparse linear system. As a result, we can obtain a global, detail-preserving and non-iterative solution V'. However, since Eq.6.15 just takes into account rotation rather than scaling, it can be observed that the Laplacian coordinate framework leads to some geometric distortion, such as shrinkage as shown in Fig.6-4c.

Shape and Feature Preservation

Our goal is to transfer the desired details to the surface of another specified model without any noticeable artifacts. Shape and feature preservation depends on the smoothed version $\tilde{M}_d$ and the details ξ_i of M_0. Assume that $\tilde{M}_d$ is smooth enough here. Appropriately reoriented and scaled details ξ_i will not lead to unexpected distortions. In some scenarios, improper scaling may lead to visible artifacts. Our basic idea is to take scaling into account in the transformation Eq.6.15 between the smoothed versions $\tilde{M}_0$ and $\tilde{M}_d$ as well as rotation for shape and feature

preservation. Similar to rotation, the scaling has been obtained by our rotation-scale-shear decomposition. Accordingly, the δ-coordinates of M_d can be updated by the revised version of Eq.6.15 as follows,

$$\delta_i^{(d)} = \tilde{\delta}_i^{(d)} + Sc_i R_i \xi_i . \qquad (6.17)$$

In general, compared to rotation, the influence of the scaling seems fairly imperceptible. This is because Eq.6.16 gives out a solution of the resampling vertices V' in a least square sense, which can alleviate the errors from the scale disagreement. However, the least squared solution is not always valid in some scenarios. To emphasize this issue, we deliberately twist a tube and transfer the protuberant characters from on another tube to it as show in Fig.6-5. Since the tube is a regular geometry model, it is easy to extract the character details by simply subtracting the estimated base of the tube in Fig.6-5a. This helps us to exclude the effects from the previous steps, particularly the smoothing step which cannot yield a smooth base just like the tube without protuberant characters. Moreover, it can be observed that the topologies of the models in Fig.6-5a and 5b are different. We thus specify some positional constraints (e.g. red lines) manually and further deform the 2D embeddings to match them. This guarantees that the characters can be properly mapped to the desired positions. For comparison, we employ Eq.6.15 and 6.17 to Eq.6.16 respectively to illustrate the effects of the detail transferring with and without scaling constraints in Fig.6-5c and 5d. Note that the twisted tube in Fig.6-5b contains both rotation and shrinkage. Due to a lack of scaling constraint, it can be observed that the resulting mesh in Fig.6-5c has visible artifacts. This is due to the fact that the volume change from Fig.6-5a to Fig.6-5b (i.e. middle part of tube) is too large to overcome the scale disagreement of the characters and the twisted tube

effectively in the least squares sense. When further investigating the local mesh, it can be observed that there are many self-intersection triangles in Fig.6-5c.

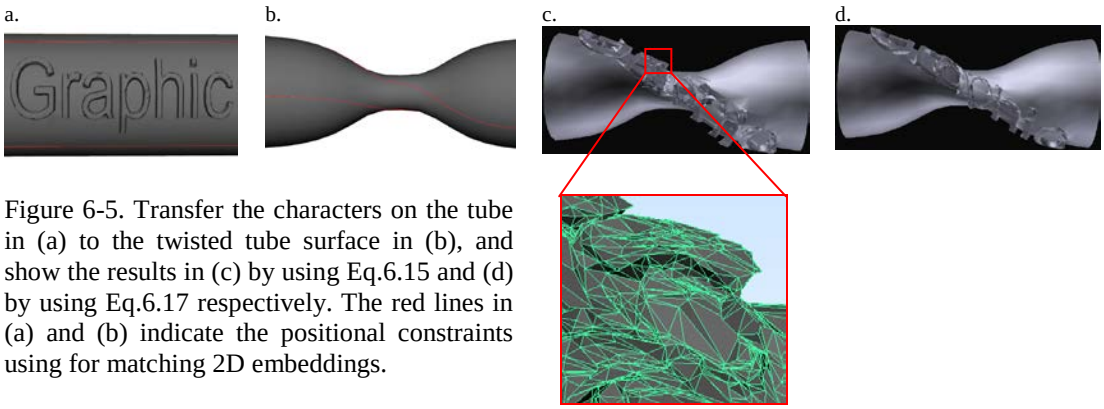
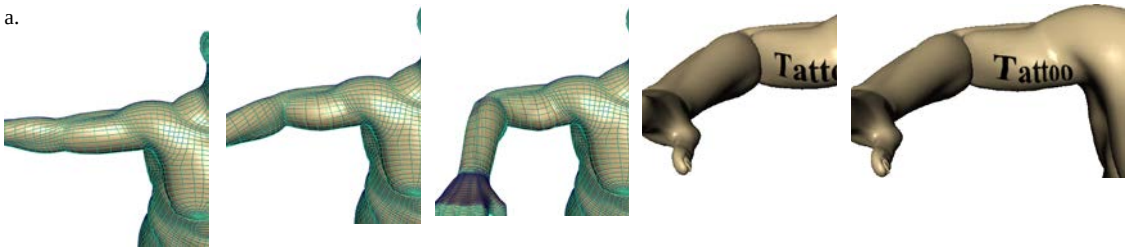


Figure 6-5. Transfer the characters on the tube in (a) to the twisted tube surface in (b), and show the results in (c) by using Eq.6.15 and (d) by using Eq.6.17 respectively. The red lines in (a) and (b) indicate the positional constraints using for matching 2D embeddings.

6.5 IMPLEMENTATION AND ANALYSIS

In our proposed skin sliding algorithm, the partition step needs to solve a generalized eigensystem with a large, sparse and symmetric matrix by using Lanczos algorithm. Figure 6-2 indicates that we just need to compute the first t eigenvectors of the graph Laplacian instead of all eigenvectors. We thus utilized a freely available code--eigifp() at [Money and Ye 2005] here, which implements an inverse free preconditioned Krylov subspace projection method. The cropped skinning patches (i.e. stretch regions in our experiments) can usually maintain quadrangle-like boundaries.



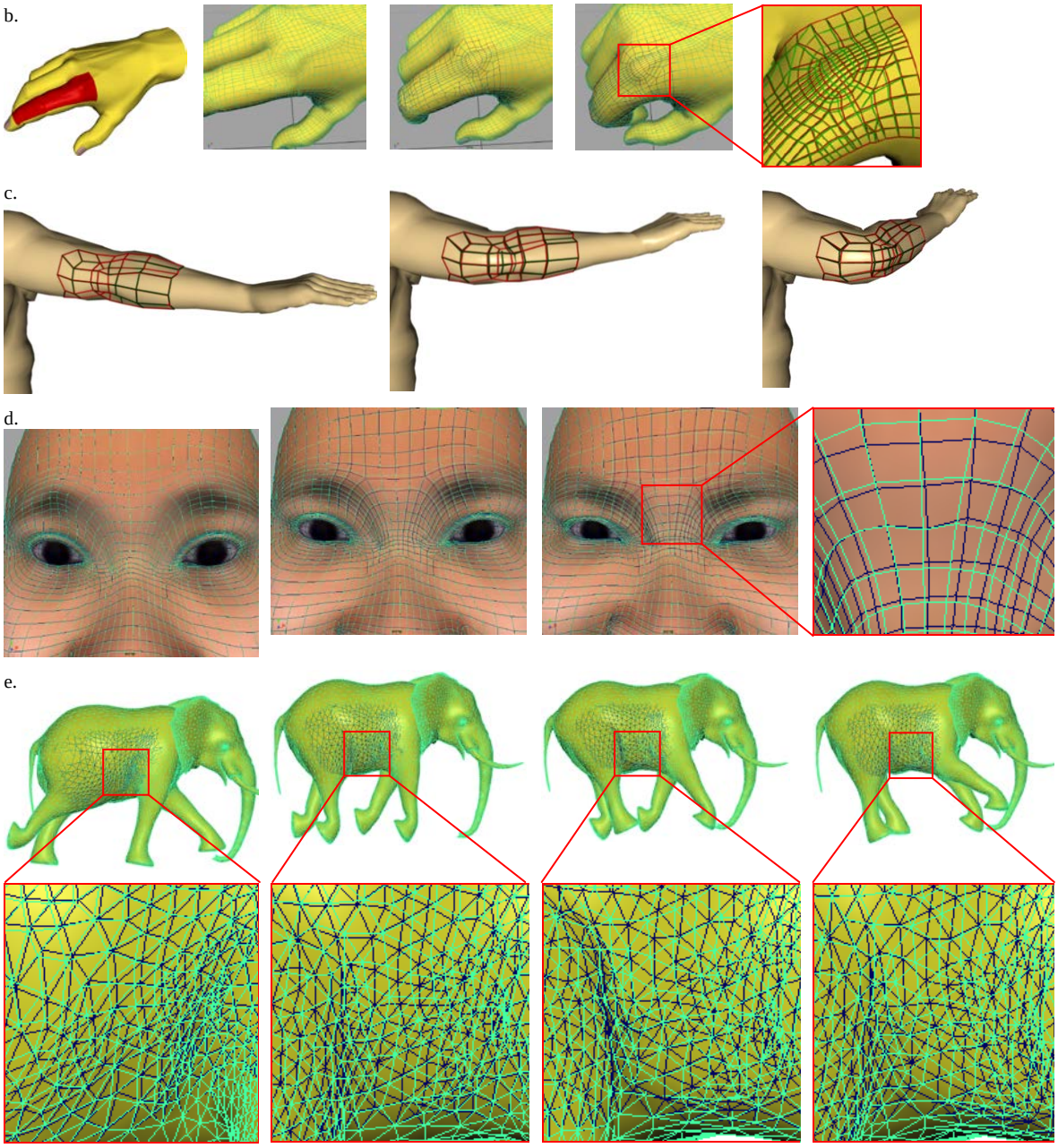


Figure 6-6. (a) shows the arm muscle bulge. The first 3 images demonstrate the mesh change during arm bending. We highlight the results by our method in blue while the result by [Yang et al. 2009] in red. The last 2 images show the difference of the character locations due to skin sliding by our method (the 4th image) and the linear interpolation [Yang et al. 2009] (the 5th image) respectively. (b) shows the skin sliding effect of clenching the fist. The red mesh is maded by our method while the blue one by [Yang et al. 2009]. (c) shows the skin sliding effect at the elbow during arm bending. The red mesh is maded by our method while the blue one by [Yang et al. 2009]. (d) shows the skin sliding effect of frowning expression. The blue mesh is made by our method while the red one by [Yang et al. 2009]. (e) shows the skin sliding effect of elephant running. The blue mesh is made by our method while the red one by [Yang et al. 2009]. Since the cropped compression and stretch ROIs are adjacent to each other here, we set them as a whole skinning patch for further processing.

In some rare scenarios, it is possible to encounter some large skinning patches. The small patches are easier to flatten into such a 2D rectangular parametric domain

with low distortion. We thus have to cut them into a set of adjacent small patches manually. However, in the resampling step, such adjacent patches are not processed individually but are stacked together into Eq.6.16 for a quasi global solution. Moreover, it is possible to stack all of the adjacent and nonadjacent patches together into Eq.6.16 for a global solution. However, due to the inverse of the large Laplacian matrix, we still processed the nonadjacent patches individually in our implementation.

In the following experiments, we first performed our proposed skin sliding algorithm on a set of human skinning regions in Fig.6-6. To demonstrate the efficacy of our proposed method in comparison to the existing method [Yang et al. 2009], we overlapped the resulting meshes by using our algorithm and [Yang et al. 2009] respectively together.

In the above experiments, the skin surfaces are too smooth. To demonstrate the feature preservation and transferring, we performed our proposed graph Laplacian based skin sliding algorithm on the tail of a seahorse model. The seahorse tail was bent up here. Because of the large stretch and bend, the bumpy surface of the tail became both very flat and very smooth, that is, it lost many concave convex details (see Fig.6-7b). Our goal is to recover the details of the tail mesh as much as possible rather than its volume before deformation. The skin sliding result is shown in Fig.6-7c. For the comparison purposes, we also show the result using linear interpolation method [Yang et al. 2009] in Fig.6-7d. It can be observed that the result achieved using the linear interpolation method in Fig.6-7d is smoother than that achieved using our proposed skin sliding algorithm in Fig.6-7c. This implies that the linear interpolation method cannot transfer details to the surface of the target model. Moreover, one can observe that the seahorse tail after bending in Fig.6-7b is very

close to the interpolation result in Fig.6-7d. Since the tail surface after bending is too smooth in Fig.6-7b, performing any interpolation technique on it without any extra cues of the surface will not bring about any new details or features. This further justifies that the linear interpolation methods cannot transfer the features to the target surface.

Further Extension

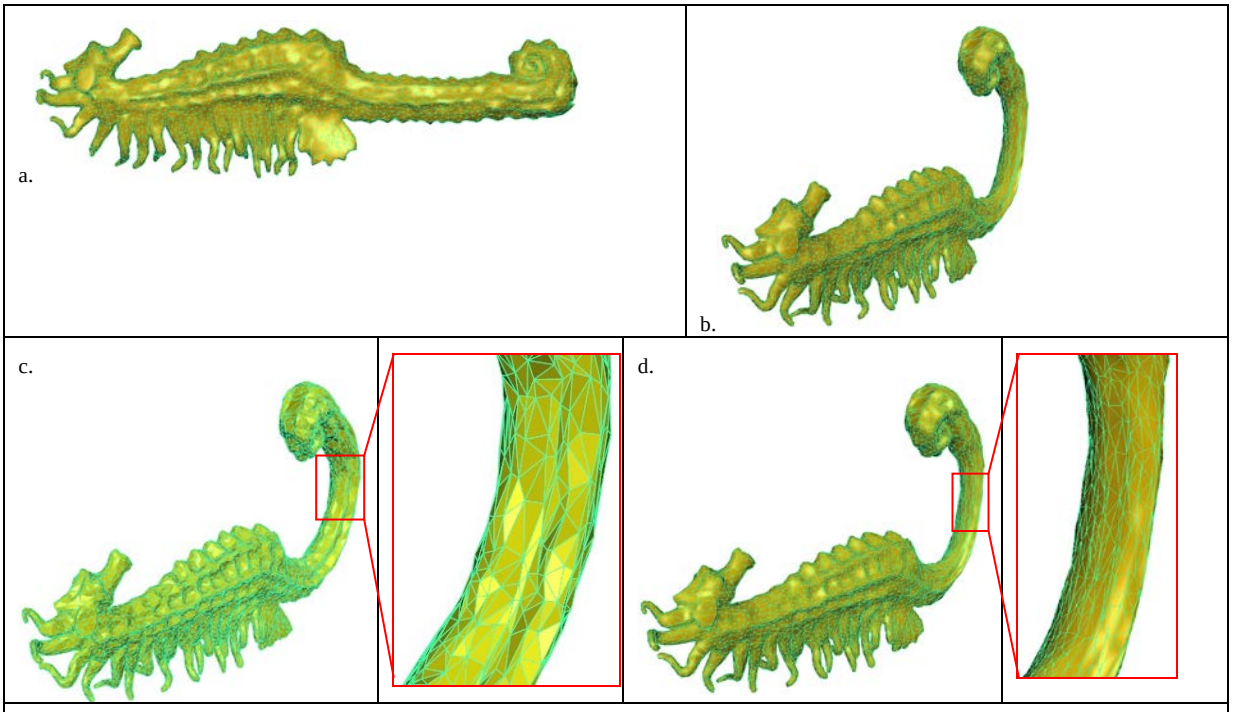


Figure 6-7. The original seahorse models before and after bending its tail are shown in (a) and (b) respectively. Since the tail is bent up and stretched, the surface is too smooth. The result produced by our proposed skin sliding algorithm is shown in (c). For comparison, the result achieved using the linear interpolation method in [Yang et al. 2009] is shown in (d). Our method is performed on the tail part instead of the whole seahorse mesh. Thus, the entire tail was first cropped manually as the input of the partition step, containing vertices 3153 and triangles 6774. Then the partition step yielded two patches, stretch and compression ROIs. Since they were adjacent, we viewed them as an entire skinning patch containing vertices 765 and triangles 1467.

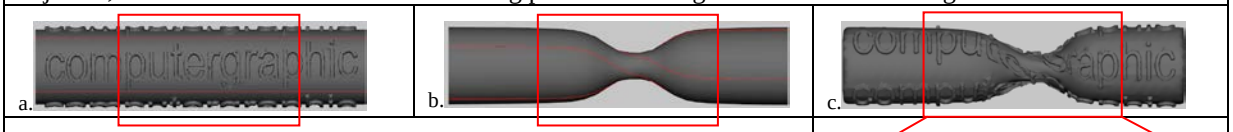


Figure 6-8. Illustration of transferring the characters to a twisted tube surface. The source and target models are shown in (a) and (b) respectively. (c) shows the result of our proposed method. Since the partition step only determines the skinning region rather than stretching and compressing regions correspondingly, we performed our method on a skinning part of tube (i.e. highlighted by red rectangle) instead of the whole tube. The cropped mesh contains the vertices 33170 and triangles 66191.

In the above skin sliding applications, the details/features are transferred between the source and target meshes, which share the same topology. In partice, the detail transferring is usually requested to perform between arbitrary surfaces with the different topologies as well. This causes a big challenge, that is, how to establish the correspondence between the source and target meshes. The following experiment shows that our proposed skin sliding algorithm can deal with such challenging issue well. We performed our method on the character transferring test, that is, transferring 4 sets of the protuberant characters—"computergraphic" from a tube to another twisted tube surface, as shown in Fig.6-8. It can be observed that the twisted tube maintains high distortion in the middle part in Fig.6-8b, i.e. large rotation (rotation angle up to 135°) and shrinkage. This implies that the Laplacian coordinates encounter both rotation and scaling challenges here. Moreover, unlike the seahorse surface, the character geometry is both regular and complicated in general. Any distortion is prone to cause visible artifacts. This is in favorable in relation to evaluating the performance of our proposed algorithm. Since the topologies of the source and target models in Fig.6-8a and Fig.6-8b are different to each other, the 2D embedding maps need to be deformed with some positional constraints for further establishing the correspondence between them. This is indeed the well known topic of the 2D parameterization with positional constraints. We employed the previous method—RBF based re-parameterization method here [Yu et al. 2011]. The red lines in Fig.6-8a and 8b indicate the positional constraints. In addition, since the twisted regions were obvious and the meshes were too large, we cropped the skinning patches manually. Using our proposed method, the transferred characters almost retain their shapes, giving the impression of underlying motion.

Moreover, we also show the computational times in Table 6-1. The computer configuration for these experiments was an Intel Pentium 4 CPU 2.8GHz with 2 GB RAM and Nvidia GeForce 8400 GS. All the codes were not optimized in MatLab in our implementation. Since the human skinning regions in Fig.6-6 are very small compared to the hand in Fig.6-2, the seahorse tail in Fig.6-7 and the twisted tube in Fig.6-8, Table 1 only shows the computational times of each step of the hand, seahorse tail and twisted tube respectively. Note that the partition step performs on the whole mesh domain, while the following steps perform on the resulting patches. Thus, the running time of the partition step is obviously much more compared to the following steps.

Table 6-1 Running Time

	Vertices/triangles	Partition	Vertices/triangles (per patch)	Smoothing	Corresponding	Re-sampling
Hand	7997/15855	126.21 sec	191/358	0.12 sec	0.06 sec	0.03 sec
Seahorse tail	3153/6774	52.18 sec	765/1467	2.91 sec	2.16 sec	0.67 sec
Twisted tube	--	--	33170/66191	97.27 sec	17.12 sec	7.98 sec

In the twisted tube test, we cropped the ROI by manual instead of the partition step.

6.6 CHAPTER SUMMARIZATION

This chapter addresses the application of skin sliding based on the graphic Laplacian framework. The foundation of this mesh editing system is to overlap the 2D parameterization of the two skinning patches before and after deformation for preserving and transferring mesh features. Thus, in order for the small distortion caused by parameterizations, the proposed pre-procedure, including mesh partition

and mesh smoothing, is performed on the meshes in advance to generate the well-shaped and well-spaced skinning patches. This is essentially to satisfy the rigid constraint for the following parameterization procedure.

Unlike the methods presented in the previous chapters, the rigid constraint is carried out in a pre-processing procedure instead of the parameterization procedure here.

CHAPTER 7

CONCLUSIONS

This thesis focuses on the constraints underlying various surface parameterization applications. Usually, the specified constraints include positional constraints, area or angle etc. Moreover, minimum distortion is desired as well, which is called as the rigid constraint in this thesis. These constraints are usually application-dependent. For different application purposes, the emphasis of these constraints may be changeable accordingly, that is, positional constraints, tradeoff of positional-rigid constraint, and rigid constraint. The main challenges addressed in this thesis include the triangle flipping (or called foldover) and the suitable boundary mapping. This thesis explores the mentioned challenges by the implementations of the following applications, texture maps, image warping and mesh editing.

7.1 SUMMARY

Simple positional constraints.

This kind of constraints is most popular in surface parameterization applications. The challenges include one-to-one mapping (also called topology preservation) and minimal distortion. In this research, we focus on the topology preservation rather than others. It is frequently observed that many deformations might result in self-intersecting meshes. This issue is also called mesh foldover. The first contribution of my work is to formulate a mathematic condition, called the foldover-free condition that guarantees the connection relationship of a mesh is preserved during deformation. Furthermore, incorporating with this foldover-free condition, a foldover-free iterative mechanism is presented. Inspired by the application of texture mapping, the proposed iterative mechanism is employed and an RBF-based re-parameterization method with positional constraints is presented for texture mapping 3D models from 2D texture images. The second contribution of my work is that the proposed RBF-based method provides the estimate of the scaling factor as the iterative step-length.

Compared to the existing techniques, the proposed method does not require predefined restrictions in the implementation, such as fixing the boundaries of the mesh. This adds greater freedom for reducing distortion than do other methods [Kraevoy et al. 2003, Lee et al. 2008], and the resulting 2D parameterization has smaller distortion. In addition, the method is a mesh-free approach, allowing direct treatment of multiborder topology. Complexity analysis suggests a low computation cost, which depends mainly on the number of the vertices.

Moreover, foldover is not only observed in the triangular meshes but also in the high dimensional datasets, e.g. polyhedral meshes and volume data. The third contribution of my work is to extend the proposed RBF-based re-parameterization method to the high dimensional datasets. The high dimensional deformable fields usually require a high-dimensional dataset to be optimally deformed in terms of the constraints of both the geometry and the topology. It is straightforward to apply the surface parameterization techniques to high dimensional space. Therefore, we further focused on the high dimensional deformable fields with internal hard constraints and revised the RBF-based reparameterization method to solve this particular issue. Our experiments suggest our method works very well even in situations where there exist a large number of constraints.

Tradeoff of positional-rigid constraint.

This kind of constraints is usually employed to image deformation applications. Large distortions inevitably bring about the noticeable artifacts. This is unacceptable in such applications. The main contribution of my work is to introduce the feasible subspace approach to the deformation field. As a result, applying the reconstructed displacement fields to the deformation can approximate the positional constraints with minimal distortion. Our proposed framework maintains two distinct advantages. The first is to enforce the topology preservation constraints on a given displacement field. This allows us to develop a foldover-free constrained deformation approach. The second is the ability to easily incorporate the rigid constraint, e.g. As-Rigid-As Possible deformation. It guarantees the resulting distortions are as small as possible. Experimental results further validate the effectiveness of our proposed feasible subspace based deformation method. Moreover, another important application is the

volume data registration. The challenging issue is to estimate the divergence and curl fields in 3D space. Unfortunately, this still remains open.

Rigid constraint.

This kind of constraints is always requested in various mesh editing applications. Essentially, it requests a suitable boundary mapping to reduce distortion caused by parameterization. In this research, we present a pre-processing procedure for the surface parameterization with minimum distortion, i.e. the rigid constraint is carried out in the proposed pre-processing procedure instead of in the following surface parameterization procedure. Our application aims at the skin sliding simulation. Modeling skin sliding realistically is a notoriously difficult task due to the complex anatomical dependencies of the outer layers of the human body. Although the mass spring system may yield realistic results, it is very expensive, both in terms of computation time and the difficulty of specifying a highly sophisticated system. It is an appropriate alternative to resampling the target mesh with some specified shape and feature constraints. However, in practice there are many challenging issues, e.g. missing out features of skin surface and smoothness issues etc. This greatly reduces the realism of the appearance. Essentially, skin sliding assumes that the shape can be preserved and the features of skin surface can be transferred to the target skin mesh.

The key step of our skin sliding approach is to parameterize the regions of interest in 2D domain and overlap them for matching. Due to the distortion caused by 2D parameterization procedure, a challenging issue arise, that is, the shape/details preservation and transfer raise challenge. We focus on these issues here and further reformulated the implementation of skin sliding based on the graph Laplacian

framework, including mesh partition, shape and feature preservation. The elements worth mentioning are the mesh partition and mesh smoothing. The former is a new application of the active research issue: mesh segmentation. We apply the Normalized cuts to its implementation here. Compared to the existing mesh segmentation approaches, the novelty is that the partition boundaries can be smoothened effectively by the min-cut/max-flow method. The latter is a new method for mesh smoothing that is implemented using our proposed tangent plane based smoothing method. These two procedures can effectively improve the mesh quality.

7.2 FUTURE WORK

Because the constraints of the constrained surface parameterization are application-dependent, this thesis focuses on the three kinds of constraints based on the different applications. This further incurs more challenging issues in these parameterization applications.

In texture mapping application, we employ the triangle subdivision technique for some extreme cases in our presented RBF-based reparameterization algorithm. This leads to the addition of redundant vertices. Thus, new methods to reduce the redundant vertices with a minimum loss of smoothness will be developed in a future work. In addition, the analysis of the time complexity indicates that running time could rapidly increase when adding a number of constraint vertices. The primary performance cost is to compute the inverse of the large symmetric matrix. In future, a GPU-based algorithm will be developed to speed up the computation.

Furthermore, to extend the RBF-based reparameterization algorithm to high dimensional datasets, the main challenge is to save running time. Thus, developing a GPU-based algorithm will be our future work.

Compared the RBF-based deformation algorithms presented in Chapter 3 and 4 with the feasible subspace based deformation approach presented in Chapter 4, the former has infinite support while the latter maintains more rigidity. Thus, for some scenarios with large deformation, the feasible subspace based deformation algorithm may become invalid. However, it is able to reduce distortion as much as possible. This is the distinct advantage against the other deformation methods. In future, it is natural to extend this algorithm to medical volume data, since it is vital to reduce the distortion caused by any local deformation in medical imaging.

In skin sliding application, we focus on the rigid constraint rather than the positional constraints. The proposed pre-processing procedure, involving mesh partition and smoothing procedures, can effectively reduce distortion caused by the following parameterization procedure. However, for large meshes, it is necessary for our skin sliding algorithm to speed up the partition and smoothing steps. An alternative is to implement our algorithm with GPU. The challenging issues include, computing preconditioners in a parallel way to solve the generalized eigensystem in the partition step and using the Jacobi style iteration instead of the Gauss-Seidel iteration in the smoothing step. It is worth exploring a more effective implementation of these specific challenges in our future work.

BIBLIOGRAPHY

Alexa, M., Cohen-Or, D. and Levin, D., (2000) As-rigid-as-possible shape interpolation, in Proc. of ACM SIGGRAPH'00.

Allen, B., Curless, B. and Popovic, B., (2003) "The space of human body shapes: Reconstruction and parameterization from range scans", in Proc. of ACM SIGGRAPH'03, pp.587–594.

Alliez, P. and Gotsman, C., (2003) "Recent advances in compression of 3D meshes", in Proc. of the Symposium on Multiresolution in Geometric Modeling'03, pp. 3-26.

Anguelov, D., Srinivasan, P., Koller, D., Thrun, S., Rodgers, J. and Davis, J., (2005) "SCAPE: Shape completion and animation of people," in Proc. of ACM SIGGRAPH'05.

Arfken, G.B. and Weber, H.J., (1995) Mathematical Methods for Physicists (4th edition), Academic Press: San Diego, pp.91-94.

Ashikhmin, M., (2001) Synthesizing natural textures, ACM Symposium on Interactive 3D Graphics, pp.217–226, March 2001.

Ashburner, J. and Friston, K.J., (2000) "Voxel-Based Morphometry—The Methods", NeuroImage, Vol.11, No.6, pp.805–821.

Avidan, S. and Shamir, A., (2007) Seam carving for content-aware image resizing, in Proc. of ACM SIGGRAPH 2007.

Boykov, Y. and Kolmogorov, V., (2004) An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.23, No.11, pp.1124-1137.

Beg, M.F., Miller, M.I., Trouve, A. and Younes, L., (2005) Computing Large Deformation Metric Mappings via Geodesic Flows of Diffeomorphisms, International Journal of Computer Vision, Vol.61, No.2, pp.139-157.

Bennis, C., Vezien, J.-M. and Iglesias, G., (1991) “Piecewise surface flattening for nondistorted texture mapping”, in Proc. of ACM SIGGRAPH’91, pp.237–246.

Biermann, H., Martin, I., Bernardini, F. and Zorin, D., (2002) “Cut-and-paste editing of multiresolution surfaces,” in Proc. of ACM SIGGRAPH’02, pp.312–321.

Cabral, M., Lefebvre, S. and Dachsbacher, C., (2009) “Structure-preserving reshape for textured architectural scenes”, Proc. of EUROGRAPHICS2009, pp.469-480.

Carr, J.C., Beatson, R.K., Cherrie, J.B., Mitchell, T.J., Fright, W.R., McCallum, B.C. and Evans, T.R., (2001) “Reconstruction and Representation of 3D Objects with Radial Basis Functions”, in Proc. of ACM SIGGRAPH’01, Los Angeles, CA, pp.67-76, August 2001.

Chadwick, J.E., Haumann, D.R. and Parent, R.E., (1989) Layered Construction for Deformable Animated Characters, in Proc. of ACM SIGGRAPH’89, pp. 243-252.

Catmull, E.E., (1972) A system for computer generated movies, In Proc. of ACM Annual Conference’72, pp.422-431.

Cootes, T.F., Edwards, G.J. and Taylor, C.J., (1998) "Active Appearance Models", in Proc. of European Conference on Computer Vision, Vol.2, pp.484-498.

Daniel, G. and Chen, M., (2003) Video Visualization, in Proc. of the 14th IEEE Visualization 2003 (VIS'03).

Dong, W., Zhou, N., Paul, J. and Zhang, X., (2009) Optimized image resizing using seam carving and scaling, in Proc. of ACM SIGGRAPH Asia 2009.

Desbrun M., Meyer M., and Alliez P., (2002) "Intrinsic Parameterizations of Surface Meshes", Proc. Eurographics'02/Computer Graphics Forum, Vol.21, No.3, pp.209-218.

Desbrun, M., Meyer, M., Schröder P. and Barr. A.H., (2002) Discrete Differential-Geometry Operators for Triangulated 2-Manifolds, in Proc. of VisMath'02, Berlin, Germany.

Desbrun, M., Meyer, M., Schröder, P. and Barr, A.H., (1999) Implicit fairing of irregular meshes using diffusion and curvature flow, in Proc. of ACM SIGGRAPH'99, pp.317-324.

Dong, S., Bremer, P.-T., Garland, M., Pascucci, V. and Hart, J.C., (2006) "Spectral surface quadrangulation", ACM Transactions on Graphics, Vol. 25, Issue 3.

Eckstein, I., Surazhsky, V. and Gotsman, C., (2001) "Texture Mapping with Hard Constraints", Proc. of Eurographics'01/Computer Graphics Forum, Vol.20, No.3, pp.95-104.

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

Efros, A. and Leung, T., (1999) "Texture synthesis by nonparametric sampling", in Proc. of ICCV'99, pp.1033–1038.

Efros, A. and Freeman, W.T., (2001) Image quilting for texture synthesis and transfer, in Proc. of ACM SIGGRAPH'01, pp.341–346.

Floater, M.S. and Hormann, K., (2005) "Surface parameterization: a tutorial and survey", in Advances in Multiresolution for Geometric Modeling (Eds. N. A. Dodgson, M. S. Floater and M. A. Sabin), Springer, pp.157-186.

Floater M.S., (1997) "Parameterization and Smooth Approximation of Surface Triangulations", Computer Aided Geometric Design, Vol.14, No.3, pp.231-250.

Floater, M.S. and Hormann, K., (2002) Parameterization of triangulations and unorganized points, in Tutorials on Multiresolution in Geometric Modelling, A. Iske, E. Quak and M. S. Floater (eds.), Springer-Verlag, Heidelberg, pp.287-315.

Fujimura, K. and Makarov, M., (1998) "Foldover-free image warping", Graphical Models and Image Processing, Vol.60, No.2, pp.100-111.

Gorla, G., Interrante, V. and Sapiro, G., (2001) Growing fitted textures, in Proc. of ACM SIGGRAPH'01, Sketches and Applications, pp.191.

Golub, G.H. and Van Loan, C.F., (1996) "Matrix Computations", Baltimore: Johns Hopkins University Press, pp.320.

Gu, X., Gortler, S. and Hoppe, H., (2002) "Geometry images", in Proc. of ACM SIGGRAPH'02, pp.356–361.

Guo, Y., Wang, J., Sun, H., Cui, X., and Peng, Q., (2005) "A novel constrained texture mapping method based on harmonic map", Comput. Graph. Vol.29, No.6, pp.972-979.

Guenter, B., Grim, C., Wood, D., Malvar, H., and Pighin, F., (1998) Making Faces, *Proc. of ACM SIGGRAPH'98*, pp.55-66.

Guskov, I., Vidimce, K., Sweldens, W. and Schroder, P., (2000) “Normal meshes”, in Proc. of ACM SIGGRAPH’00, pp.95–102.

Haber, E. and Modersitzki, J., (2007) Image Registration with guaranteed displacement regularity, International Journal of Computer Vision, Vol.71, pp.361-372

Haber, E., Horesh, R. and Modersitzki, J., (2010) Numerical Methods for constrained image registration, Numerical linear algebra with applications, Vol.17, pp.343-359.

Hagenlocker, M. and Fujimura, K., (1998) “CFFD: a tool for designing flexible shapes”, The Visual Computer, Vol.14, pp.271-287.

Haker, S., Angenent, S., Tannenbaum, A., Kikinis, R., Sapiro, G., and Halle, M., (2000) “Conformal Surface Parameterization for Texture Mapping”, IEEE Transactions on Visualization and Computer Graphics, Vol.6, No.2, pp.181-189.

Heeger, D.J. and Bergen, J.R., (1995) “Pyramid-based texture analysis/synthesis”, in Proc. of ICIP’95, pp.III:648–651.

Hoppe, H. and Praun, E., (2005) “Shape compression using spherical geometry images,” in Advances in Multiresolution for Geometric Modeling’05, pp.27–46.

Igarashi, T., Moscovich, T. and Hughes, J.F., (2005) As-rigid-as-possible shape manipulation, ACM Trans. on Graphics, Vol.24, No.3.

Joshi, S.C. and Miller, M.I., (2000) Landmark matching via large deformation diffeomorphisms, IEEE Trans. on Image Process, Vol.9, No.8, pp.357-70.

James, D.L. and Twigg, C.D., (2005) Skinning mesh animations, ACM Trans. on Graphics, Vol.24, No.3.

Joshi, P., Meyer, M., DeRose, T., Green, B. and Sanocki, T., (2007) Harmonic coordinates for character articulation, ACM Trans. On Graph., Vol.26, Issue 3.

Karni, Z., Freedman, D. and Gotsman, C., (2009) Energy-based image deformation, in Proc. of the Symposium on Geometry Processing'09.

Karni, Z. and Gotsman, C., (2000) Spectral compression of mesh geometry, in Proc. of ACM SIGGRAPH'00.

Kavan, L., Collins, S., Žára, J. and O'Sullivan, C., (2007) Skinning with dual quaternions, in Proc. of the ACM Symposium on interactive 3D Graphics and Games'07, April 30-May 02, 2007.

Karacali, B. and Davatzikos, C., (2004) Estimating topology preserving and smooth displacement fields, IEEE Trans. On Med. Imaging, Vol.23, No.7, pp.868-80.

Khodakovsky, A., Litke, N. and Schroder, P., (2003) "Globally smooth parameterizations with low distortion", in Proc. of ACM SIGGRAPH'03, pp.350–357.

Kirk, W.A. and Khamsi, M.A., (2001) "An Introduction to Metric Spaces and Fixed Point Theory", John Wiley, New York.

Knupp, P.M., (2001) Algebraic Mesh Quality Metrics, SIAM J. on Scientific Computing, Vol.23, Issue 1, pp.193-218.

Kovar, L. and Gleicher, M., (2004) "Automated extraction and parameterization of motions in large data sets", ACM Trans. on Graphics, Vol.23, Issue 3, pp.559-568.

Kraevoy, V. and Sheffer, A., (2005) "Template based mesh completion," in Proc. of Symposium on Geometry Processing (SGP05).

Kraevoy V., Sheffer A., and Gotsman C., (2003) “Matchmaker: Constructing Constrained Texture Maps”, ACM Trans. Graphics, Vol.22, No.3, pp.326-333.

Krähenbühl, P., Lang, M., Hornung, A. and Gross, M., (2009) A system for retargeting of streaming video, in Proc of ACM SIGGRAPH Asia 2009.

Kwatra, V., Schodl, A., Essa, I., Turk, G. and Bobick, A., (2003) Graph-cut textures: Image and video synthesis using graph cuts, ACM Trans. on Graphics, Vol.22, No.3, pp:277–286.

Kwatra, V., Essa, I., Bobick, A. and Kwatra, N., (2005) “Texture optimization for example-based synthesis,” ACM Trans. on Graphics, Vol.24, No.3, pp.795–802.

Kybic, J. and Unser, M., (2003) Fast parametric elastic image registration, IEEE Trans. on Image Process., Vol.12, No.11, pp.1427-42.

Lee, A., Hoppe, H. and Moreton, H., (2000) “Displaced subdivision surfaces”, in Proc. of ACM SIGGRAPH’00.

Lee, S.Y. and Chwa, K.Y., (1995) "Image metamorphosis using snakes and free-form deformations", Proc. of SIGGRAPH'95, pp.439-448.

Lee, T.-Y. and Huang, P.H., (2003) “Fast and Institutive Polyhedral Morphing Using SMCC Mesh Merging Scheme”, IEEE Trans. on Visualization and Computer Graphics, Vol.9, No.1, pp.85-98.

Lee, T., Yen, S. and Yeh, I., (2008) “Texture Mapping with Hard Constraints Using Warping Scheme”, IEEE Transactions on Visualization and Computer Graphics, Vol.14, No.2, pp.382-395.

Lerios, A., Garfinkle, C.D. and Levoy, M., (1995) "Feature-based volume metamorphosis", in Proc. of ACM SIGGRAPH’95, pp.449-456.

Levy, B., (2001) “Constrained Texture Mapping for Polygonal Meshes”, Proc. of ACM SIGGRAPH’01, pp.417-424.

Levy, B., and Mallet, J.L., (1998) “Non-distorted Texture Mapping for Sheared Triangulated Meshes”, Proc. of ACM SIGGRAPH1998, pp.343-352.

Levy, B., Petitjean, S., Ray, N., and Maillot, J., (2002) “Least Squares Conformal Maps for Automatic Texture Atlas Generation”, ACM Transactions on Graphics, Vol.21, No.3, pp.362-371.

Levy, B., (2003) “Dual domain extrapolation,” in Proc. of ACM SIGGRAPH’03.

Lewis, J.P., Cordner, M. and Fong, N., (2000) Pose Space Deformation: a unified approach to shape interpolation and skeleton driven deformation, in Proc. of ACM SIGGRAPH’00, pp.165–172.

Liang, L., Liu, C., Xu, Y., Guo, B. and Shum, H.Y., (2001) Real-time texture synthesis using patch-based sampling, ACM Trans. on Graphics, Vol.20, No.3.

Liu, L., Zhang, L., Xu, Y., Gotsman, C. and Gortler, S.J., (2008) A local/global approach to mesh parameterization, Proc. of the Symposium on Geometry Processing, Copenhagen, Denmark, July 02-04, 2008.

Lipman, Y., Kopf, J., Cohen-Or, D. and Levin, D., (2007) GPU-assisted positive mean value coordinates for mesh deformations, in Proc. of the 5th Eurographics symposium on Geometry processing, pp.117-123.

Lipman, Y., Levin, D. and Cohen-Or, D., (2008) Green Coordinates, ACM Trans. on Graphics, Vol.27, Issue 3.

Liu, R. and Zhang, H., (2007) Mesh Segmentation via Spectral Embedding and Contour Analysis, Computer Graphics Forum, Vol.26, Issue 3, pp.385–394.

Magda, S. and Kriegman, D., (2003) Fast texture synthesis on arbitrary meshes, in Proc. of Eurographics Symposium on Rendering, June 2003.

Meisters, G.H. and Olech, C., (1963) “Locally one-to-one mappings and a classical theorem on schlicht functions”, Duke Math. J., Vol.30, No.1, pp.63-80.

Meyer, M., Desbrun, M., Schröder, P. and Barr, A.H., (2003) Discrete Differential-Geometry Operators for Triangulated 2-Manifolds, Visualization and Mathematics III, Hans-Christian Hege and Konrad Polthier (editors), Springer-Verlag, Berlin, pp.35-58.

Merkwirth, C., Parlitz, U., Wedekind, I., Engster, D. and Lauterborn, W., (2009) OpenTSTOOL package, available at http://www.physik3.gwdg.de/tstool/.

Merry, B., Marais, P. and Gain, J., (2006) Animation space: A truly linear framework for character animation, ACM Trans. On Graph., Vol.25, No.4, pp.1400-1423.

Mullen, P., Tong, Y., Alliez, P. and Desbrun, M., (2008) Spectral Conformal Parameterization, Computer Graphics Forum, Vol.27, Issue 5, pp.1487–1494.

Musse, O., Heitz, F. and Armspach, J.P., (2001) Topology preserving deformable image matching using constrained hierarchical parametric models, IEEE Trans. on Image Process., Vol.10, No.7, pp.1081-93.

Money, J.H. and Ye, Q., (2005) Algorithm 845: EIGIFP: a MATLAB program for solving large symmetric generalized eigenvalue problems, ACM Trans. on Mathematical Software, Vol.31, Issue 2.

Pérez, P., Gangnet, M. and Blake, A., (2003) Poisson image editing, ACM Trans. on Graphics, Vol.22, Issue 3.

Piponi, D., and Borshukov, G., (2000) “Seamless texture mapping of subdivision surfaces by model pelting and texture blending”, Proc. of SIGGRAPH2000, pp.471-478.

Praun, E., Finkelstein, A. and Hoppe, H., (2000) Lapped textures, in Proc. of ACM SIGGRAPH’00, pp.465–470.

Rohlfing, T., Maurer, C.R. Jr, Bluemke, D.A. and Jacobs, M.A., (2003) Volume-preserving nonrigid registration of MR breast images using free-form deformation with an incompressibility constraint, IEEE Trans. on Med. Imaging, Vol.22, No.6, pp.730-41.

Rubinstein, M., Shamir, A. and Avidan, S., (2008) Improved seam carving for video retargeting, in Proc. of ACM SIGGRAPH’08.

Sander P.V., Snyder J., Gortler S.J. and Hoppe H., (2001) “Texture Mapping Progressive Meshes”, Proc. ACM SIGGRAPH’01, pp.409-416.

Sander, P., Gortler, S., Snyder, J., and Hoppe, H., (2002) “Signal specialized Parameterization”, Proc. of Eurograph Workshop on Rendering.

Sander, P.V., Wood, Z.J., Gortler, S.J., Snyder, J. and Hoppe, H., (2003) Multi-chart geometry images, in Proc. of the Eurographics/ACM SIGGRAPH symposium on Geometry processing’03.

Scheepers, F., Parent, R.E., Carlson, W.E. and May, S.F., (1997) Anatomy-Based Modeling of the Human Musculature, in Proc. of ACM SIGGRAPH’97, pp.163-172.

Sederberg, T.W. and Parry, S.R., (1986) "Free-form deformation of solid geometric models", in Proc. of ACM SIGGRAPH’86, pp.151-160.

Shi, J. and Malik, J., (2000) Normalized cuts and image segmentation, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.22, Issue 8, pp.888-905.

Sheffer, A., Praun, E., and Rose, K., (2006) MESH PARAMETERIZATION METHODS AND THEIR APPLICATIONS, Foundations and Trends in Computer Graphics and Vision, Vol.2, No.2, pp.105-171.

Sheffer, A. and Sturler, E., (2001) "Parameterization of Faceted Surfaces for Meshing Using Angle Based Flattening", Engineering with Computers, Vol.17, No.3, pp.326-337.

Sheffer, A., Levy, B., Mogilnitsky, M., and Bogomyakov, A., (2005) "ABF++: fast and robust angle based flattening", ACM Trans. On Graph., Vol.24, No.2, pp.311-330.

Schaefer, S., McPhail, T. and Warren, J., (2006) Image deformation using moving least squares, in Proc. of ACM SIGGRAPH'06.

Sorzano, C.O., Thévenaz, P. and Unser, M., (2005) Elastic registration of biological images using vector-spline regularization, IEEE Trans. on Biomed. Eng., Vol.52, No.4, pp.652-63.

Sorkine, O. and Cohen-Or, D., (2004) "Least-Squares Meshes", in Proc. of Int'l Conf. on Shape Modeling and App.

Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rossl, C. and Seidel, H.P., (2004) Laplacian Mesh Editing, in Proc. of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing'04, pp.179-188.

Sorkine, O. and Alexa, M., (2007) As-rigid-as-possible surface modeling, Proc. of the 5th Eurographics symposium on Geometry processing, Barcelona, Spain, July 04-06, 2007.

Sorkine, O., (2006) Differential Representations for Mesh Processing, Computer Graphics Forum, Vol.25, No.4, pp.789--807.

Stinson, W.T. and Thuriot, P.G., (2004) Bulging Muscles and Sliding Skin: Deformation Systems for Hellboy, in Proc. of ACM SIGGRAPH'04 Sketches.

Sumner, R.W. and Popović, J., (2004) Deformation transfer for triangle meshes, ACM Trans. on Graphics, Vol.23, Issue 3.

Tang, Y., Wang, J., Bao, H.J. and Peng, Q.S., (2003) "RBF-based constrained texture mapping", Computers and Graphics, Vol.27, No.3, pp.415-422.

Tao, J., Schaefer, S. and Warren, J., (2005) Mean value coordinates for closed triangular meshes, in Proc. of ACM SIGGRAPH'05, pp.561-566.

Thalmann, M.N., Laperrière, R. and Thalmann, D., (1988) Joint-dependent local deformations for hand animation and object grasping, in Proc. of Graphics interface'88 (Canada), pp.26-33.

Tong, Y., Lombeyda, S., Hirani, A.N. and Desbrun, M., (2003) Discrete multiscale vector field decomposition, ACM Trans. on Graphics, Vol.22, Issue 3.

Tong, X., Zhang, J., Liu, L., Wang, X., Guo, B. and Shum, H.Y., (2002) Synthesis of bidirectional texture functions on arbitrary surfaces, ACM Trans. on Graphics, Vol.21, No.3, pp:665–672.

Turner, R. and Thalmann, D., (1993) The Elastic surface layer model for animated character construction, in Proc. of Computer Graphics International'93, Lausanne, Switzerland, pp.399–412.

Turk, G., (2001) Texture synthesis on surfaces, in Proc. of ACM SIGGRAPH'01, pp.347–354.

Tiddeman, B., Duffy, N. and Rabey, G., (2001) “A general method for overlap control in image warping”, Computer & graphics, Vol.25, No.1.

Ulrike von Luxburg, (2010) A tutorial on spectral clustering, STATISTICS AND COMPUTING, Vol.17, No.4, pp.395-416.

Urtasun, R., Glardon, P., Boulic, R., Thalmann, D. and Fua, P., (2004) "Style-Based Motion Synthesis", COMPUTER GRAPHICS FORUM, Vol.23, No.4, pp.799–812.

Venkatraman, K., Lodha, S. and Raghavan, R., (2005) A kinematic-variational model for animating skin with wrinkles, Computers & Graphics, Vol.29, Issue 5, pp.756-770.

Wang, X.C. and Phillips, C., (2002) Multi-weight enveloping: least squares approximation techniques for skin animation, in Proc. of ACM SIGGRAPH/Eurographics symposium on Computer animation’02, pp.129–138.

Wang, Y., Tai, C., Sorkine, O. and Lee, T.Y., (2008) Optimized scale-and-stretch for image resizing, in Proc. of ACM SIGGRAPH Asia 2008.

Weber, O., Sorkine, O., Lipman, Y. and Gotsman, C., (2007) Context-aware skeletal shape deformation, Computer Graphics Forum, Vol.26, Issue 3, pp.265-274.

Wang, Y.S., Fu, H., Sorkine, O., Lee, T.Y. and Seidel, H.P., (2009) Motion-aware temporal coherence for video resizing, ACM Transactions on Graphics, Volume 28, Issue 5.

Wang, Y., Lin, H., Sorkine, O. and Lee, T.Y., (2010) Motion-based video retargeting with optimized crop-and-warp, in Proc. of ACM SIGGRAPH’10.

Weber, O., Ben-Chen, M., and Gotsman, C., (2009) Complex barycentric coordinates with applications to image deformation, Computer Graphics Forum, Vol.28, No.2.

Wei, L.Y. and Levoy, M., (2001) Texture synthesis over arbitrary manifold surfaces, in Proc. of ACM SIGGRAPH'01, pp.355–360.

Wei, L.Y. and Levoy, M., (2000) "Fast texture synthesis using tree structured vector quantization", in Proc. of Siggraph'00, pp.479–488.

Weng, Y., Xu, W., Wu, Y., Zhou, K. and Guo, B., (2005) 2D shape deformation using nonlinear least squares optimization, The Visual Computer, Vol.22, Issue 9.

Winkler, T., Drieseberg J., Alexa, M. and Hormann, K., (2010) Multi-scale geometry interpolation, Computer Graphics Forum, Vol.29, No.2.

Xiong, Z., Sun, X. and Wu, F., (2010) Robust web image/video super-resolution, IEEE Trans. on Image Processing, Vol.19, Issue 8.

Xue, Z., Shen, D., and Davatzikos, C., (2006) "Statistical representation of high-dimensional deformation fields with application to statistically constrained 3D warping", Medical Image Analysis, Vol.10, Issue 5, pp.740-751.

Yang, X., Southern, R. and Zhang, J., (2009) Fast simulation of skin sliding, Computer Animation and Virtual Worlds, Vol.20, Numbers 2-3, pp.333-342.

Yang, J., Wright, J., Huang, T.S. and Ma, Y., (2010) Image Super-Resolution Via Sparse Representation, IEEE Trans. on Image Processing, Vol.19, Issue 11, pp.2861-2873.

Yoshizawa, S., Belyaev, A. and Seidel, H.P., (2004) “A Fast and Simple Stretch-Minimizing Mesh Parameterization”, Proc. of Int’l. Conf. on Shape Modeling and Applications (SMI'04).

Yu, H., Lee, T., Yeh, I., Yang, X., Li, W. and Zhang, J., (2011) RBF-Based Re-parameterization Method for Constrained Texture Mapping, IEEE Trans. On Visualization and Computer Graphics, available at: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5928343&tag=1

Zhang, E., Mischaikow, K. and Turk, G., (2005) “Feature-based surface parameterization and texture mapping”, ACM Trans. on Graphics, Vol.24, Issue 1.

Zhou K., Wang X., Tong Y., Desbrun M., Guo B. and Shum H.-Y., (2005) “TextureMontage: Seamless Texturing of Arbitrary Surfaces from Multiple Images”, ACM Trans. Graphics, Vol.24, No. 3, pp.1148-1155.

Zhang, S., Huang, J. and Metaxas, D.N., (2011) Robust mesh editing using Laplacian coordinates, Graphical Models, Vol.73, Issue 1, pp.10-19.

Zhang, J., Zhou, K., Velho, L., Guo, B. and Shum, H.Y., (2003) Synthesis of progressively variant textures on arbitrary surfaces, ACM Transactions on Graphics, Vol.22, No.3, pp:295–302.