Optimal Boundaries for Poisson Mesh Merging

Xiaohuang Huang[1,2 *†] Hongbo Fu[2 ‡] Oscar Kin-Chung Au[2 ‡] Chiew-Lan Tai[2 ‡]
[1]Zhejiang University [2]Hong Kong University of Science and Technology

Abstract

Existing Poisson mesh editing techniques mainly focus on designing schemes to propagate deformation from a given boundary condition to a region of interest. Although solving the Poisson system in the least-squares sense distributes the distortion errors over the entire region of interest, large deformation in the boundary condition might still lead to severely distorted results. We propose to optimize the boundary condition (the merging boundary) for Poisson mesh merging. The user needs only to casually mark a source region and a target region. Our algorithm automatically searches for an optimal boundary condition within the marked regions such that the change of the found boundary during merging is minimal in terms of similarity transformation. Experimental results demonstrate that our merging tool is easy to use and produces visually better merging results than unoptimized techniques.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Boundary representations;

Keywords: optimal boundaries, Poisson mesh merging

1 Introduction

Directly modeling 3D geometric objects from scratch is often difficult and time-consuming. Instead, mesh editing techniques aim to create models by modifying existing ones, usually obtained from 3D scanners. Mesh merging, as one of the most popular mesh editing tools, produces new meshes by composing parts of existing models. For example, user can merge the body of a women model with the tail of a fish to create an interesting mermaid model. Mesh merging is achieved either by blending details of meshes through an intermediate surface or by deforming the merging boundaries of meshes as well as the meshes themselves and stitching the merging boundaries together. Our method falls into the second category.

In recent years, several differential mesh editing techniques have been proposed (see the latest surveys in [Sorkine 2006] and [Huang et al. 2006]). Besides their easy implementation, these techniques support intuitive user interface: they allow the user to simply manipulate parts of a surface, called handles, and the deformation of the rest surface is computed by solving the Poisson equation subject to boundary condition from the handles.

By regarding the merging boundaries as the boundary condition, differential techniques are directly applicable to mesh merging [Sorkine et al. 2004; Yu et al. 2004]. Without loss of generality, in this paper we consider the merging problem as the deformation problem of a source mesh when the source merging boundary is deformed to the corresponding merging boundary on a target mesh, which is never deformed.

Regardless of the specific differential representation (e.g. the Laplacian coordinates [Sorkine et al. 2004; Lipman et al. 2004] or gradient field [Yu et al. 2004]) used in these techniques, we call all the merging tools based on these techniques *Poisson mesh merging*, as all of them need to solve a set of Poisson equations subject to the Dirichlet boundary condition. After merging, the errors from the change of the boundary condition in the source mesh are distributed over the region of interest in the least-squares sense. However, if the boundary condition undergoes large distortion, the merging procedure may still produce seriously distorted results. The distortion is especially large near the boundary condition (Figure 1c), as it provides soft or hard constraints to the deformation optimization [Sorkine et al. 2004]. Therefore the effectiveness of the existing Poisson mesh merging techniques is highly dependent on how carefully the user specifies the merging boundaries (the boundary condition).

A similar problem exists in Poisson image editing. Poisson image editing [Pérez et al. 2003] may generate bad image composition results, especially when the boundary conditions on the source and target images severely conflict with each other. To address the problem, Jia et al. [2006] propose to compute an optimized boundary condition for Poisson image editing: a boundary condition is optimal if it undergoes only a translation transformation in $\{r, g, b\}$ color spaces during composition.

Motivated by [Jia et al. 2006], we present an algorithm for easy Poisson mesh merging. It finds an optimal merging boundary within the regions casually marked by the user. A new objective function is proposed to find a boundary condition under an (unknown) similarity transformation during merging in the least-squares sense. Unlike images, meshes often have irregular sampling. We incorporate an edge-based weighting scheme to alleviate the influence of irregular sampling. Similar to [Jia et al. 2006], we use an alternating optimization method to solve the resulting complicated nonlinear optimization problem. After obtaining the optimized boundary condition, we apply one of the state-of-the-art differential mesh editing techniques, dual Laplacian mesh editing [Au et al. 2006], to deform and merge the source mesh to the target mesh.

Compared with existing Poisson mesh merging techniques, our system supports much easier user interface: the user only needs to *casually* mark the region to be cut on the source mesh and the desirable region to be pasted on the target mesh. Without the user's fine tuning of the merging boundaries, the optimal boundary condition leads to visually good merging results (Figure 1d).

2 Related Work

Poisson Image Editing. Poisson image editing is a powerful tool for image composition [Pérez et al. 2003]. This technique seamlessly blends two images by solving the Poisson equations with

*This work was done while Xiaohuang Huang was an exchange student at the Hong Kong University of Science and Technology.

†e-mail: huangxiaohuang@cad.zju.edu.cn

‡e-mail: {fuhb, oscarau, taicl}@cse.ust.hk

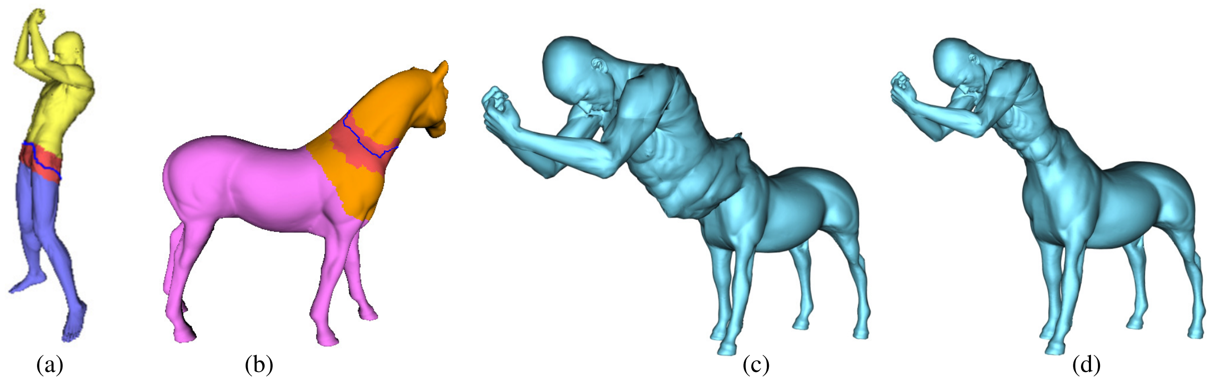

Figure 1: *The effectiveness of Poisson mesh merging is highly dependent on the choice of the boundary conditions. (a) Source mesh. (b) Target mesh. (c) and (d) are the merging results with the user-specified (unoptimized) boundary condition (i.e. the boundary separating the region in red and the region in blue) and the optimal one found by our algorithm (in blue), respectively.*

guidance fields from the source image and a boundary condition from the target image. However, the effectiveness of Poisson image editing is dependent on how the user carefully specifies the boundary condition. To make Poisson mesh editing easier to use, Jia et al. [2006] propose to optimize the boundary condition within a region roughly marked by the user.

Differential Mesh Editing. The idea of Poisson image editing has been successfully extended to 3D mesh editing domain [Yu et al. 2004; Sorkine et al. 2004]. Differential mesh editing techniques either use gradient fields [Yu et al. 2004] (similar to the guidance fields in image editing) or Laplacian coordinates [Lipman et al. 2004; Sorkine et al. 2004] to represent the original mesh. A deformed surface is reconstructed from these differential representations by solving a set of Poisson equations subject to the user-specified boundary condition,

$$\Delta\mathbf{x} = \mathbf{T}\delta, \qquad \mathbf{x}|_{\partial\Omega} = \mathbf{x}_0|_{\partial\Omega}, \tag{1}$$

where $\mathbf{x}$ is an unknown scalar function representing x, y or z value of vertices of the deformed surface, δ is the differential representation of the undeformed surface, and $\mathbf{x}_0$ provides the desirable values on the boundary $\partial\Omega$. Unlike Poisson image editing, appropriate transformation $\mathbf{T}$ is needed to transform δ before reconstruction, as δ is not rotation-invariant [Lipman et al. 2004; Yu et al. 2004].

To obtain natural deformed results, the transformation $\mathbf{T}$ is required to be as-rigid-as-possible, or as close to a similarity transformation as possible if uniform scaling is needed. As 3D rotation transformations are nonlinearly dependent on vertex positions, differential mesh editing is essentially nonlinear. In most of early solutions, for fast computation, this transformation problem is approximately formulated as linear least-squares minimization problems. According to whether or not the formulation of $\mathbf{T}$ depends on the (unknown) deformed surface, these linear techniques can be classified as implicit [Sorkine et al. 2004; Fu et al. 2006] or explicit ones [Zayer et al. 2005; Yu et al. 2004; Zhou et al. 2005; Shi et al. 2006]. However, all of them only partially solve the transformation problem; they cannot handle either distortion caused by large angle rotation or distortion from a pure translation of the boundary condition. To completely address the problem, several nonlinear solutions have been proposed recently [Huang et al. 2006; Botsch et al. 2006; Au et al. 2006]. We use the dual Laplacian editing system [Au et al. 2006] to deform the source mesh after the boundary condition is changed.

Mesh Merging and Surface Pasting. Cut-and-paste editing is ubiquitous in text and image processing applications. It has been extended to 3D mesh domain to compose new models from parts of existing models.

Kanai et al. [1999] present a mesh merging technique based on local 3D metamorphosis. This method allows details from the source and target meshes to be smoothly blended together in the final merging result. Later, the idea of transferring details is extended to multiresolution framework [Biermann et al. 2002] and differential framework [Sorkine et al. 2004]. These methods need to build one-to-one correspondence between the whole source and target regions of interest, thus requiring their topologies the same. To remove this requirement of topology, Fu et al. [2004] uses the base surfaces of the source and target regions of interest for correspondence building.

Recently, merging techniques based on Poisson mesh editing have been proved effective. Poisson mesh merging [Sorkine et al. 2004; Yu et al. 2004] deforms the meshes while deforming the source and target merging boundaries to be stitched together. As no surface parameterization is involved for correspondence building, Poisson mesh merging is applicable to regions with nonzero genus. However, if the merging boundary is not well chosen, the merging result might still be bad. In this paper, we present an automatic algorithm to find an optimal merging boundary within the merging regions that are roughly specified by the user.

The merging results with most existing merging techniques [Sorkine et al. 2004] largely depend on the well-adjusted relative positions of source and target meshes. However, precisely adjusting relative positions of models in 3D space is a difficult task, even for experienced users. To ease user's effort, Fu et al. [2006] propose a configuration-independent merging that produces the same merging result given the same boundary correspondence, regardless of the relative positions of models. Sharf et al. [2006] present another intuitive mesh merging technique, with which the user only needs to roughly adjust the relative positions of models until there is a significant overlap between them, then the source mesh is automatically snapped and merged to the target mesh. Hassner et al. [2005] introduce a part-in-whole model alignment method to aid the user in positioning the models. After the model alignment, they find a minimal cut on the graph respecting both the source and target models to simultaneously cut and stitch the models.

3 System Overview

We give a system overview in this section. Our goal is to find an optimal merging boundary on the target mesh to be used as the boundary condition to deform the part of the source mesh containing the features to be pasted onto the target mesh. The target mesh remains undeformed. Our system contains the following main steps (Figure 2):

1. The user casually marks a region of interest Ω_0 on the source mesh. This region should be large enough to cover the features $\Omega_{feature}$ (i.e. $\Omega_{feature} \subset \Omega_0$) that the user really wants to paste onto the target mesh. Intuitive cutting techniques, e.g. easy mesh cutting [Ji et al. 2006], can be used to identify $\Omega_{feature}$. To avoid having the optimal boundary $\partial\Omega$ cutting into $\Omega_{feature}$, we constrain $\partial\Omega$ to be within the region $\Omega_0 \setminus \Omega_{feature}$. On the target mesh, the user roughly chooses a region Ω_1 onto which the features from the source mesh are to be pasted. We assume that Ω_1 does not contain complex features; otherwise we simply remove the features before the pasting so as to reduce distortion in the next parameterization step.

2. For each vertex in $\Omega_0 \setminus \Omega_{feature}$, we find the corresponding position on Ω_1 (Figure 4). Unlike 2D image editing, there is no explicit correspondence between the source and target meshes in 3D. We use one of state-of-the-art surface parameterization methods, least squares conformal maps [Lévy et al. 2002], to build the correspondence.
   Unfortunately, the band shape of $\Omega_0 \setminus \Omega_{feature}$ often leads to large parameterization distortion, which might defeat the gain from having an optimal boundary condition. To reduce parameterization distortion, we first fill the hole induced by boundary $\partial\Omega_{feature}$ through an optimal triangulation that minimizes the total triangle area [Barequet and Sharir 1995]. We then parameterize the surface $(\Omega_0 \setminus \Omega_{feature}) \cup \Omega_{filled}$, where Ω_{filled} is the region resulting from the boundary triangulation. Replacing $\Omega_{feature}$ with Ω_{filled} for the purpose of correspondence building has the following advantages. First, additional distortion would not be introduced from parameterizing $\Omega_{feature}$, which could be of complicated geometric shape. Second, as $\Omega_{feature}$ is not used in surface parameterization, this region can be of complex topology (e.g. with nonzero genus).
   To obtain a meaningful correspondence, the user needs to manually translate, scale and rotate the parameterization of $\Omega_0 \setminus \Omega_{feature}$ with respect to the parameterization of Ω_1. The relative positions of the source and target models are roughly fixed once the correspondence is determined. The subsequent algorithm only fine tunes the final orientations and scalings. Therefore, the user can anticipate the composition effect when specifying the correspondence.

3. We search for a closed path $\partial\Omega$ within region $\Omega_0 \setminus \Omega_{feature}$ as the optimal boundary condition through an iterative optimization algorithm (Section 4). To guarantee that $\partial\Omega$ encloses $\Omega_{feature}$, we cut across the ring-like region $\Omega_0 \setminus \Omega_{feature}$ and search for a boundary that begins and ends at this cut.

4. We move the vertices on $\partial\Omega$ to their corresponding target positions on Ω_1 and perform Poisson mesh merging to deform the region enclosed by $\partial\Omega$ (containing $\Omega_{feature}$). We choose to use the dual Laplacian editing framework [Au et al. 2006], as it completely solves the transformation problem of differential-based deformation.

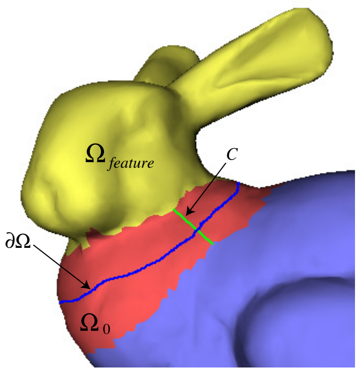

Figure 2: *An illustration of different types of boundaries and regions on the source mesh. Ω_0 is the region of interest (in red) casually marked by the user. $\Omega_{feature}$ contains the features to be merged (in yellow). The optimal boundary $\partial\Omega$ (in blue) lies in the region $\Omega_0 \setminus \Omega_{feature}$. Cut C (in green) breaks the ring of $\Omega_0 \setminus \Omega_{feature}$.*

4 Optimal Boundary

Our boundary condition optimization for Poisson mesh merging is inspired by [Jia et al. 2006]. However, extending the boundary condition optimization algorithm from Poisson image pasting [Jia et al. 2006] to Poisson mesh merging is not straightforward.

First, the optimization problem in mesh merging becomes more complicated, with more unknowns introduced. In 2D image editing [Jia et al. 2006], the resulting composite has the best quality when the difference between the source and target boundary conditions is a constant (i.e., corresponding to a pure translation of the boundary condition in color spaces). In Poisson mesh merging, the desirable scenario is when there exists a similarity transformation (consisting of a rigid transformation and uniform scaling) between the source and target merging boundaries, as there would be no shearing or stretching distortion in the reconstructed meshes [Fu et al. 2006]. However, such desirable transformation does not exist in most merging scenarios, thus we search for a least-squares solution: the change of the boundary condition in the source mesh during merging is minimal in terms of an unknown similarity transformation.

Second, irregular sampling in meshes makes the objective function in [Jia et al. 2006] inapplicable to Poisson mesh merging. For images, whether or not sampling factor is considered in the objective function is insignificant due to the regular structure. However, sampling consideration is crucial when designing the objective function to optimize the boundary condition in 3D.

4.1 Boundary Energy Minimization

Following the above discussion, an optimal boundary for Poisson mesh merging is a boundary $\partial\Omega$ in $\Omega_0 \setminus \Omega_{feature}$ such that the transformation between $\partial\Omega$ and its corresponding boundary $\partial\Omega^*$ on Ω_1 is as close as possible to an unknown similarity transformation $\mathbf{T}$. We formulate the objective function to be minimized as follows:

$$E(\partial\Omega, \mathbf{T}) = \sum_{e \in \partial\Omega} \|\mathbf{T}e - e^*\| \cdot length(e), \quad \partial\Omega \subset \Omega_0 \setminus \Omega_{feature}, \quad (2)$$

where e is any edge on $\partial\Omega$, i.e., a vector with its endpoint positions as the starting and ending points, e^* is the corresponding edge of e

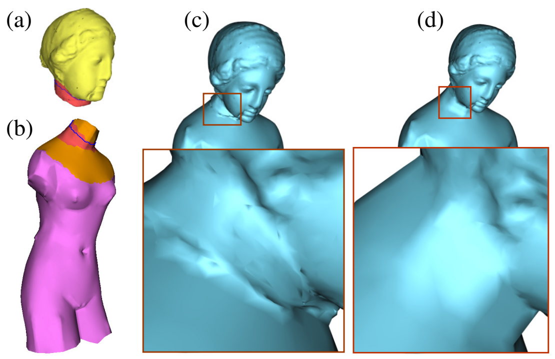

Figure 3: *(a) Source. (b) Target. (c) and (d) are the merging results using the unoptimized and optimal boundary conditions, respectively.*

on Ω_1, and $\|\cdot\|$ denotes L2 vector norm. As meshes often have irregular sampling of geometry, we add the term $length(e)$ to prevent the optimal boundary from bypassing regions with dense sampling. Another desirable effect of adding the weighting factor is that the length of the optimal boundary will be as short as possible, pushing it to approach $\partial\Omega_{feature}$.

4.2 Iterative Optimization

Since the optimal boundary might contain all the vertices in $\Omega_0 \setminus \Omega_{feature}$, minimizing $E(\partial\Omega, \mathbf{T})$ to solve for the optimal merging boundary and the transformation simultaneously is intractable. Similar to [Jia et al. 2006], we use an alternating method to solve the minimization problem iteratively. Mainly, it contains the following steps:

1. Initialize $\partial\Omega$ as $\partial\Omega_0$.
2. Given the current boundary $\partial\Omega$ on the source mesh and its corresponding boundary $\partial\Omega^*$ in Ω_1 on the target mesh, we compute the optimal similarity transformation $\mathbf{T}$. Specifically, given the corresponding sets of points on $\partial\Omega$ and $\partial\Omega^*$, we use the algorithm in [Horn 1987] to compute a rigid motion. The uniform scaling factor is defined as the ratio of the average edge length of $\partial\Omega^*$ to that of $\partial\Omega$.
3. Given the current transformation $\mathbf{T}$, we optimize the boundary $\partial\Omega$.
4. Repeat steps 2 and 3 until the change of the energy $E(\partial\Omega, \mathbf{T})$ converges or it reaches a prescribed maximum number of iterations.

Given $\mathbf{T}$, solving for $\partial\Omega$ by minimizing the boundary energy $E(\partial\Omega, \mathbf{T})$ is equivalent to finding a shortest path in $\Omega_0 \setminus \Omega_{feature}$. However, we have an additional requirement here: $\partial\Omega$ should enclose $\Omega_{feature}$. The path found by a standard shortest path problem (e.g. Dijkstra algorithm [Dijkstra 1959]) is very likely not the one we need. To fulfill the requirement, we first break the ring-like region $\Omega_0 \setminus \Omega_{feature}$ by adding a cut C, as shown in Figure 2, and then find a shortest path that starts and ends at C. After cutting, each original vertex on the cut C is split into two vertices, on different sides of C.

To achieve better performance, we want a cutting path C with minimal number of vertices. A zigzag cut C may make the found shortest path $\partial\Omega$ intersects C more than once, thus leading to a non-optimal boundary condition [Jia et al. 2006]. Straightening the cut

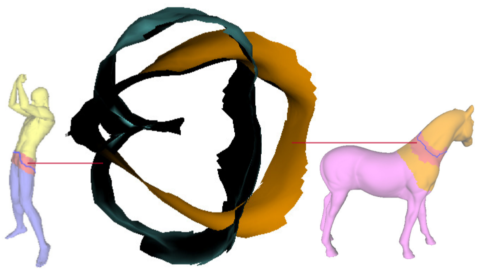

Figure 4: *Region $\Omega_0 \setminus \Omega_{feature}$ on the source mesh and its corresponding region on the target mesh.*

can greatly reduce the possibility of multiple intersections. Therefore we find the shortest path with a source vertex on $\partial\Omega_0$ and a sink vertex on $\partial\Omega_{feature}$ as the cut C. The shortest path is computed using Dijkstra algorithm [Dijkstra 1959] with edge lengths as the weighting costs.

Given a cut C, we show how to compute a closest shortest path that begins and ends at a vertex on C as the boundary $\partial\Omega$. We associate each edge e with cost $\|\mathbf{T}e - e^*\| \cdot length(e)$. The accumulated cost of a path is defined as the summation of the costs of all edges on the path. For each vertex u on one side of the cut C, we use Dijkstra algorithm to compute the shortest path $path(u)$ with minimal cost to the vertex v which is originally split from the same vertex as u. The optimal boundary $\partial\Omega$ is set as the one with the minimum cost from the set $\{path(u) \mid u \in C\}$.

Like ours, the algorithm proposed by Hassner et al. [2005] finds a merging boundary respecting both the source and target models. Their solution consists of two main steps: model alignment (to find an appropriate transformation) followed by a minimal cut. In a sense, their solution only corresponds to one iteration of ours and thus is not optimal.

5 Examples and Discussion

In this section, we demonstrate that optimal boundary conditions lead to merging results with less distortion (i.e., less shearing and stretching) than those reconstructed using user-specified unoptimized boundary conditions Ω_0. We do not compare the change of the global shapes of the source features when using the two different types of boundary conditions, as it is dependent on the scale factor. Instead, we compare the local distortions of the merged source meshes. For the optimal or unoptimized boundary condition, we use the ratio of the average length of the source and target merging boundaries to uniformly scale the Laplacian coordinates of the source mesh to account for the difference in the sizes between the source and target boundaries.

When the source and target merging boundaries are of very different shapes, the deformed source mesh inevitably exhibits local distortion. The distortion is more noticeable near the merging boundary, as the merging boundary serves as soft or hard constraint to the deformation optimization. For the source and target models in Figure 1, the region $\Omega_0 \setminus \Omega_{feature}$ on the source mesh and its corresponding region on the target mesh are of very different shapes (Figure 4). Using the user-specified boundary condition, the local distortion, especially near the boundary (i.e. the waist region), is large. In contrast, the optimal boundary condition leads to a much better merging result. Better merging results are also demonstrated by the examples in Figures 3, 5, 6 and 7, when optimal merging boundaries are used.

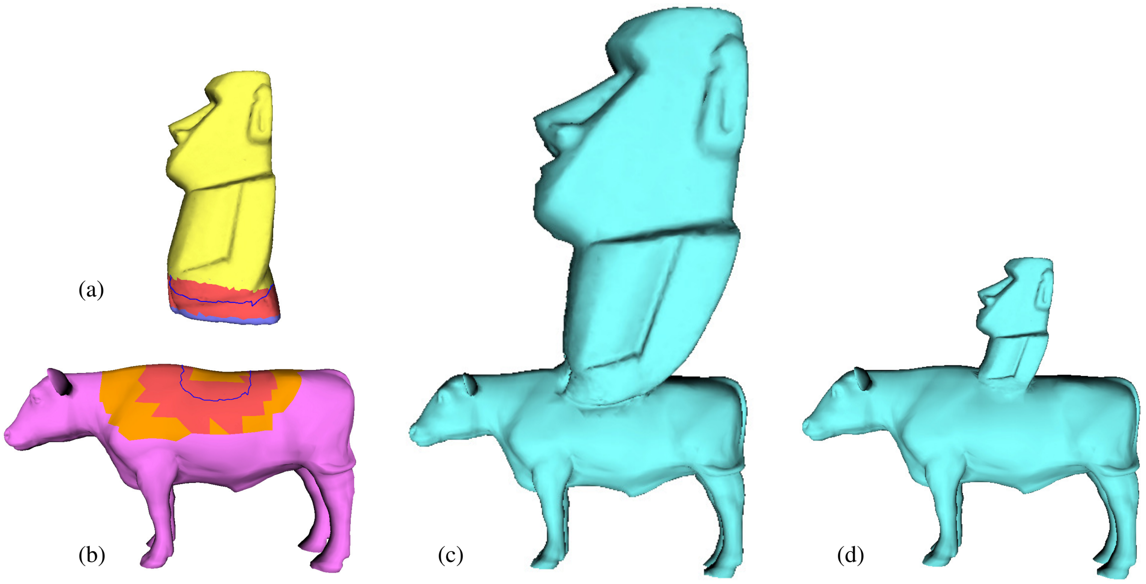

Figure 5: *(a) Source. (b) Target. (c) and (d) are the merging results using the unoptimized and optimal boundary conditions, respectively.*

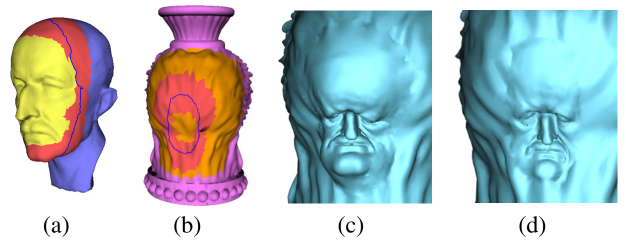

Figure 6: *(a) Source. (b) Target. (c) and (d) are the merging results using the unoptimized and optimal boundary conditions, respectively.*

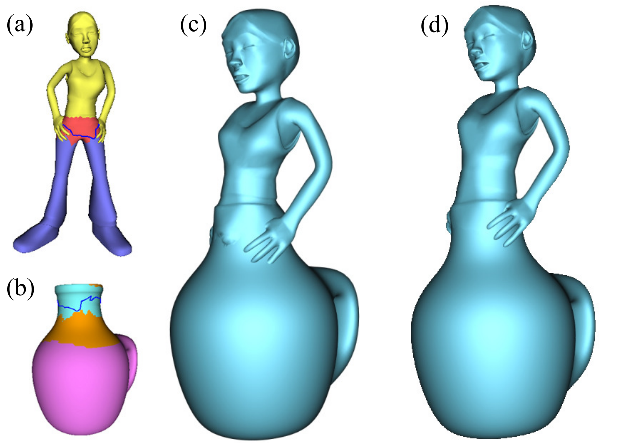

Figure 7: *(a) Source. (b) Target. (c) and (d) are the merging results using the unoptimized and optimal boundary conditions, respectively.*

We need a planar surface parameterization to build the correspondence of $\Omega_0 \setminus \Omega_{feature}$ between the source and target meshes. However, as $\Omega_{feature}$ itself is not involved in the parameterization step, our system does not require the topology of $\Omega_{feature}$ to be homeomorphic to a disk. This is demonstrated by the example in Figure 8.

It is hard to theoretically prove the convergence of the proposed iterative method. However, experiments show that the iterative process has no convergence problem. Although the iterative process may fall into a local minimum, the resulting boundaries always lead to better merging results than those with unoptimized boundaries. As the time complexity of the Dijkstra algorithm is $O(N \log N)$, for each iteration, the overall computational complexity of finding an optimal boundary is $O(MN \log N)$, where M and N are the number of vertices on the cut C and in $\Omega_0 \setminus \Omega_{feature}$, respectively. For example, given a region $\Omega_0 \setminus \Omega_{feature}$ with about 10K vertices, it takes about 1 minutes to compute the final optimal boundary condition. Performing the Laplacian deformation is very efficient [Au et al. 2006].

6 Conclusion

We present an easy-to-use Poisson mesh merging tool without requiring careful user-specified merging boundaries. Our algorithm automatically finds an optimal boundary based on the information casually provided by the user and produces visually better merging results.

The distortion introduced in the correspondence building step using surface parameterization definitely influences the final merging results. For models with complex shapes at $\Omega_0 \setminus \Omega_{feature}$, the parameterization distortion might defeat the gain from the optimal boundary. As a future work, we will explore other registration methods, e.g. iterative closest point (ICP) algorithm [Besl and McKay 1992] or its variants, instead of using direct surface parameterization techniques.

The change of the local frames at each vertex of the boundary condition before and after merging can be used to increase the smoothness across the merging boundary [Yu et al. 2004]. We plan to incorporate the local rotations or similarity transformations into the boundary condition optimization formulation to find optimal

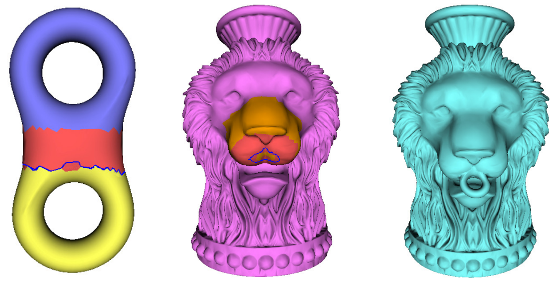

Figure 8: *Our merging is applicable to a region of interest with nonzero genus.*

boundaries that will lead to better smoothness across the merging boundary.

Currently, we search for the shortest paths on the graph of the original mesh, which restricts the found optimal boundary to be composed of the mesh edges. Getting rid of this constraint may further improve the quality of merging results.

Acknowledgment

We thank the anonymous reviewers for the helpful comments. The models are courtesy of Stanford University, 3D CAFE, Cyberware and AIM@SHAPE Shape Repository. This work was supported by a grant from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project No. HKUST6295/04E).

References

AU, O. K.-C., TAI, C.-L., LIU, L., AND FU, H. 2006. Dual Laplacian editing. *IEEE Transaction on Visualization and Computer Graphics 12*, 3, 386–395.

BAREQUET, G., AND SHARIR, M. 1995. Filling gaps in the boundary of a polyhedron. *Computer Aided Geometric Design 12*, 2, 207–229.

BESL, P. J., AND MCKAY, N. D. 1992. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 14*, 2, 239–256.

BIERMANN, H., MARTIN, I., BERNARDINI, F., AND ZORIN, D. 2002. Cut-and-paste editing of multiresolution surfaces. *ACM Transaction on Graphics 21*, 3, 312–321.

BOTSCH, M., PAULY, M., GROSS, M., AND KOBBELT, L. 2006. PriMo: coupled prisms for intuitive surface modeling. In *Symposium on Geometry Processing*, 11–20.

DIJKSTRA, E. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik 1*, 1, 269–271.

FU, H., TAI, C.-L., AND ZHANG, H. 2004. Topology-free cut-and-paste editing over meshes. In *Geometric Modeling and Processing 2004*, 173–182.

FU, H., AU, O. K.-C., AND TAI, C.-L. 2006. Effective derivation of similarity transformations for implicit Laplacian mesh editing. *Computer Graphics Forum*. To appear.

HASSNER, T., ZELNIK-MANOR, L., LEIFMAN, G., AND BASRI, R. 2005. Minimal-cut model composition. In *SMI*, 72–81.

HORN, B. K. P. 1987. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America 4*, 4, 629–642.

HUANG, J., SHI, X., LIU, X., ZHOU, K., WEI, L.-Y., TENG, S.-H., BAO, H., GUO, B., AND SHUM, H.-Y. 2006. Subspace gradient domain mesh deformation. *ACM Trans. Graph. 25*, 3, 1126–1134.

JI, Z., LIU, L., CHEN, Z., AND WANG, G. 2006. Easy mesh cutting. *Computer Graphics Forum 25*, 3, 283–291.

JIA, J., SUN, J., TANG, C.-K., AND SHUM, H.-Y. 2006. Drag-and-drop pasting. *ACM Trans. Graph. 25*, 3, 631–637.

KANAI, T., SUZUKI, H., MITANI, J., AND KIMURA, F. 1999. Interactive mesh fusion based on local 3D metamorphosis. In *Graphics interface '99*, 148–156.

LÉVY, B., PETITJEAN, S., RAY, N., AND MAILLOT, J. 2002. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph. 21*, 3, 362–371.

LIPMAN, Y., SORKINE, O., COHEN-OR, D., LEVIN, D., RÖSSL, C., AND SEIDEL, H.-P. 2004. Differential coordinates for interactive mesh editing. In *Proceedings of Shape Modeling International*, 181–190.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Transactions on Graphics 22*, 3, 313–318.

SHARF, A., BLUMENKRANTS, M., SHAMIR, A., AND COHEN-OR, D. 2006. SnapPaste: an interactive technique for easy mesh composition. *The Visual Computer 22*, 9, 835–844.

SHI, L., YU, Y., BELL, N., AND FENG, W.-W. 2006. A fast multigrid algorithm for mesh deformation. *ACM Trans. Graph. 25*, 3, 1108–1117.

SORKINE, O., LIPMAN, Y., COHEN-OR, D., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *Symposium on Geometry Processing*, 179–188.

SORKINE, O. 2006. Differential representations for mesh processing. *Computer Graphics Forum 25*, 4, 789–807.

YU, Y., ZHOU, K., XU, D., SHI, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2004. Mesh editing with Poisson-based gradient field manipulation. *ACM Trans. Graph. 23*, 3, 644–651.

ZAYER, R., RÖSSL, C., KARNI, Z., AND SEIDEL, H.-P. 2005. Harmonic guidance for surface deformation. *Computer Graphics Forum 24*, 3, 601–609.

ZHOU, K., HUANG, J., SNYDER, J., LIU, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2005. Large mesh deformation using the volumetric graph Laplacian. *ACM Trans. Graph. 24*, 3, 496–503.