Anisotropic Resizing of Model with Geometric Textures

Lin Chen
School of Computer Science and Technology,
Shandong University
chenlin@mail.sdu.edu.cn

Xiangxu Meng
School of Computer Science and Technology,
Shandong University
mxx@sdu.edu.cn

ABSTRACT
Model resizing is a useful technique for model reuse. Introduced by Kraevoy et al., non-homogeneous model resizing is able to preserve important geometric features during anisotropic scaling. However, many practical objects contain various geometric textures. Different from similar objects without geometric textures, such objects seem as if extremely difficult to resize even if the underlying surfaces are relatively simple. In this paper, we present an automatic model resizing method based on geometric texture transfer. Geometric textures are separated from the underlying surfaces and reproduced on the non-homogenously scaled surfaces using geometric texture synthesis. By utilizing the natural correspondence between the surfaces before and after resizing, surfaces with multiple geometric textures can be resized and geometric texture recovered automatically. Experimental results show that our method effectively and automatically preserves geometric textures during the model resizing process.

Keywords
Geometric texture, Anisotropic resizing, Texture synthesis

1. INTRODUCTION
Digital 3D models are widely used in many applications such as entertainment, design, and engineering. To avoid the tedious tasks of creating new models from scratch, a trend towards reuse of existing models, parts, or designs has emerged. In order to reuse such models in new assemblies or scenes, some reshaping operation may be needed. One of the principal ways of reshaping objects is through the resizing operation, i.e. scaling or stretching the object along several orthogonal directions or dimensions.

Resizing by simply applying a global scaling usually causes distortions of various parts and features. Kraevoy et al. [7] presented a non-homogeneous model resizing method which can preserve important geometric features at the cost of changing the relative scale of the geometric features in the resized model. However, many practical objects contain various geometric textures. For these kinds of models, the method in [7] alone can not produce reasonably scaled models because the regions with geometric textures will significantly affect the estimation of vulnerability, while the overall perceived shape is similar.

2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling (SPM '09), October 4-9, 2009, San Francisco, CA.

Our method processes an input model by extracting geometric textures, resizing the base mesh, and resynthesizing geometric textures, to give an output model: scaled model which preserves the geometric features along with the relative scale of the geometric textures. The basic steps are shown in Figure 1. We briefly summarize this idea and various related techniques in Section 2. An overview of our algorithm and details are given in Section 3 and experimental results are provided in Section 4. Conclusions and discussions of the future work are given in Section 5.

2. RELATED WORKS
Our approach is related to mesh segmentation, mesh resizing and geometric texture synthesis. We will discuss the most related papers in the following subsections.

2.1 Geometric texture segmentation
Various previous works deal with segmentation of mesh and separation of details from a base mesh. An extremely fast algorithm has been proposed in [10], for more methods on segmentation, We refer readers to [15] for a recent survey.

Unlike most mesh segmentation methods, our work needs to segment regions with different geometric textures. Lai et al. [11] present a clustering-based top-down hierarchical segmentation algorithm, which particularly utilizes integral invariants and their statistics for segmentation of shapes with significantly varied geometric textures. However, their statistic approach is relatively simple and can not deal with very complicated textures. Yamauchi et al. [18] develop a mesh segmentation method that combines a procedure for clustering and adopts mean shift technique for clustering scattered data. Our method is also based on mean shift segmentation; however, we use different features to allow segmentation of regions of varied geometric textures.

2.2 Geometric texture synthesis and transfer
Kobbelt et al. gave an approach to extract a base surface in [6] by mesh smoothing; Biermann et al. [3] studied this based on cut-and-paste editing, using a method based on multi-resolution subdivision surfaces; further such work can be found in the references in their paper.

Various recent work considers the transfer of geometric textures or details. Sorkine et al. [16] proposed an approach in the context of

the properties of Laplacian coordinates. They establish the correspondence by parameterization and warping under a few specified constraints. Their approach deals with geometric detail transfer, rather than statistical textures. Zelinka et al. use a polar sampling pattern called geodesic fans to extend the similarity-based image synthesis method to geometric surfaces in [21]. This approach is similar to a generalized displacement mapping. The idea of image analogies is extended to surfaces by the approach in [2] which synthesizes similar geometric textures from examples. however, their approach requires representation conversion if the input model is a mesh because their method is based on volume analogies. As in the case of image analogies, their approach requires as input a pair of example models with and without texture. Lai et al. [9] proposed an algorithm to transfer geometric textures using geometry images. Our method is different from previous approaches in that we deal with texture preservation in the scaling process: natural correspondence for the models before and after resizing can be automatically obtained, even if the models contain multiple kinds of geometric textures.

2.3 Mesh resizing

Kraevoy et al. [7] first proposed nontrivial resizing of 3D geometric models which can be considered as a generalization of content-aware image resizing (e.g. [1]) to 3D models. In their method input mesh was first embedded into a protected grid, and then the vulnerability of each cube was evaluated in order to optimize the directional scaling to obtain the desired size of each cube after scaling. The method can produce content-protected scaling results, with significant features preserved and errors distributed mostly to less important regions. However, for models contain geometric texture, vulnerability of texture region will be much different from the underlying surfaces, thus can not be scaled properly. The shape and the relative scale of the texture region will be greatly changed and such models do not suitable for reuse.

3. ALGORITHM

3.1 Overview

Given an input model represented as a triangular mesh, and a few user specified parameters, our method can resize the model in a more rational way. According to the natural correspondence between the surfaces before and after resizing, our method can under the premise of keeping the relative scale of the geometric texture part in the model, preserve the details of the geometric texture as far as possible. Our method is outlined as follows, see Figure 1.

1. Detecting and segmenting geometric textures of the model.
2. Extracting geometric textures, getting base mesh of the model.
3. Resizing the base mesh.
4. Synthesizing extracted geometric textures to the resized base mesh.

Given an input model, the first stage of processing is segmenting the geometric texture region of the model. We first calculate integral invariants to estimate geometric attributes of mesh vertices with the method in [20]. Then, we compute the expectation and variance of geometric attributes in the 1-ring neighbor of each vertex as scattered data, and employ a mean shift clustering procedure similar to [18] to segment different parts of geometric textures.

Next, we apply a texture extraction process to the segmented regions with different geometric textures. We take the geometric texture as the displacement of each vertex coordinate in its normal direction. We first perform a mesh fairing operation on the input model to estimate the shape of the base mesh, and then filter out the geometric textures. We set different parameters on geometric regions and other regions in our mesh fairing method in order to avoid filtering out feature details. After obtaining the base mesh, we then calculate geometric textures by projecting the difference of each vertex coordinates between the original mesh and the base mesh to its normal direction. Details will be described in Section 3.3.

Now, the base mesh can be resized directly without the affecting of geometric texture. In our implementation, we employ the method depicted in [7].

We next should synthesis the geometric texture to the resized base mesh. We make some modifications to the original approach described in [5]. In our method, input texture is not texture sample but extracted geometric texture. Specifically, to get the texture sample, a flattening and resampling operation should be applied to the extracted geometric texture. The solving of the synthesis process is algorithmically similar to Expectation-Maximization (EM) [13].

3.2 Texture segmentation

A 3D model often is covered more than one kind of geometric texture. Different geometric textures should be segmented and disposed respectively. The main idea behind our approach can be summarized as clustering of local geometric attributes associated with vertices approximated by a given mesh.

We first should decide which surface descriptor can be used to segment textures. In our implementation, we choose vertex coordinates combined with local principal curvatures as the attributes of our segmentation criterion. Principle curvatures can be estimated by integral invariant computed by the method in [20]. we estimate local geometric attributes by calculating the expectation $E(x_i)$ and variance $D(x_i)$ of principal curvature x_i of vertex i in its 1-ring neighborhood $N_1(i)$.

To implement clustering (segmentation), We employ mesh shift, a powerful general purpose procedure for non-parametric clustering of scattered data. Our process is similar to [18]. Given a triangle mesh, we consider the vertex coordinates c_i equipped with local geometric attributes $E(x_i)$ and $D(x_i)$ of vertex i as scattered data

$$\chi = (p_i, q_i, t_i) = (c_i, E(x_i), D(x_i))$$

in R^7 (principle curvatures in two directions).

Like method in [18], we also use separable kernel density estimation method (Parzen-window estimation) to estimate the probability density function:

$$\widehat{f}(x) = \frac{1}{N h_p^{d_p} h_q^{d_q} h_t^{d_t}} \sum_{i=1}^{N} K_1(\frac{p - p_i}{h_p}) K_2(\frac{q - q_i}{h_q}) K_3(\frac{t - t_i}{h_t}) \quad (1)$$

where h_p, h_q, h_t are the size of each window, p, q, t are the center of each window.

In our implementation, we choose Epanechnikov kernel which has a simple profile. We run our segmentation process over each vertices v_i of Mesh M to find its convergent point. The convergent

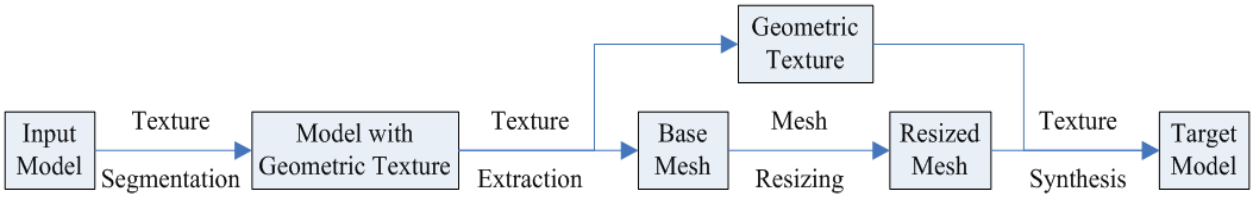


Figure 1: Algorithm overview

point determines the sort of geometric textures it belongs to. Specifically, we first initialize the window center attributes by assign $p = p_i, q = q_i, t = t_i$, then we detect all the points until they are convergent or exceed an appointed iterative times. The detection process consists of the following three steps:

1. Given a query (p, q, t), we detect all vertices p_i such that $\|p_i - p\| < h_p$.
2. For each detected vertex p_i, we detect whether its local principal curvature expectation counterpart q_i satisfies $\|q_i - q\| < h_q$.
3. For each detected vertex q_i, we check whether its local principal curvature variance counterpart t_i satisfies $\|t_i - t\| < h_t$.

3.3 Texture extraction

The purpose of texture extraction is to extract geometric details from the original model, giving a representation suitable for synthesis and transfer. In our method, we first apply an implicit fairing process to the given mesh to estimate the shape of the base mesh and filter out the geometric textures. Then we extract the geometric textures by projecting displacement of each vertex coordinate to its normal direction.

An implicit fairing method is presented in [4] which using a scale-dependent Laplacian operator to improve the diffusion process. This method is adapted to us because of the features volume preservation, and hard and soft constraints on positions of vertices of the mesh. The implementation of this method is based on *backward Euler method* and extension of simulation of heat equation as follows:

$$(I - \lambda dtL)X^{n+1} = X^n \tag{2}$$

$$L(x_i) = \frac{2}{E} \sum_{j \in N_1(i)} \frac{x_j - x_i}{|e_{ij}|}, \quad with\ E = \sum_{j \in N_1(i)} |e_{ij}|. \tag{3}$$

where λdt is a constraint factor, X^n is vertex coordinates in iteration n, $|e_{ij}|$ is the length of edge e_{ij} of mesh.

In order to preserve the volume of the original model, all the vertex positions are multiplied by $\beta = (V^0/V^n)^{1/3}$, where V^0 is the initial model volume and V^n is the model volume in iteration n. Let us denote x_k^1, x_k^2, and x_k^3 are the three vertices of the kth triangle, the volume can be computed according to [12] as follows:

$$V = \frac{1}{6} \sum_{k=1}^{nbFaces} g_k \cdot N_k \tag{4}$$

where $g = (x_k^1 + x_k^2 + x_k^3)$ and $N_k = \overrightarrow{x_k^1 x_k^2} \wedge \overrightarrow{x_k^1 x_k^3}$.

Obviously, geometric features may also be filtered during implicit fairing. In our implementation, for the sake of preserving important features of the model, we set large λdt on geometry texture regions and small one to other regions, specifically 10 and 0 respectively, see Figure 2. The fairing operation iterates until the altering of vertices coordinates of texture regions is less than a threshold or the number of iteration exceeds a specified times.

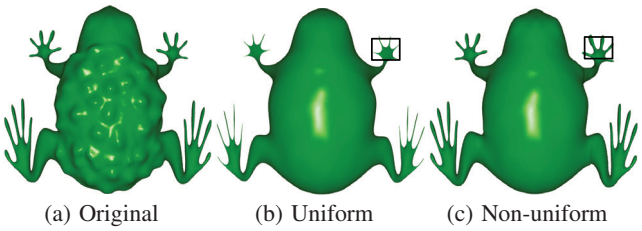


Figure 2: Smooth input model, uniformly and non-uniformly.

Because the local coherence of mesh before and after resizing in this paper, we only need to represent geometric textures as displacement affecting mesh vertices. Specifically, we represent the geometric texture t as:

$$t = (\overrightarrow{v_o} - \overrightarrow{v_b}) \cdot \overrightarrow{n} \tag{5}$$

where $\overrightarrow{v_o}$ is vertex coordinates vector of original mesh, $\overrightarrow{v_b}$ is vertex coordinates vector of base mesh, and $\overrightarrow{n}$ is vertex unit normal vector.

3.4 Texture resynthesis over resized base model

The extracted base mesh can be resized without the affection of geometric texture. In our paper we apply non-homogeneous resizing method in [7] to process the base model. Their method is based on optimization process. Details can be found in their paper.

For the sake of eliminating the distortion of geometric texture, a texture synthesis process should be adopted to assign the extracted texture back to the resized base mesh. According to the coherence of topology before and after resizing, the region to which geometric texture should be synthesized is the separated textured region before. Our texture Synthesis process is much simpler due to the natural correspondence before and after resizing. First, the direction of the texture do not have to be appointed due to the consistency of mesh before and after resizing. Secondly, The order of vertices synthesized is in accordance with that of vertices displacements extracted.

Our texture synthesis method in this paper is based on [5] which can be considered as an extension to 3D of texture synthesis method in [8]. Different from those texture synthesis method based on greedy heuristics, they synthesis textures by optimization. In their method, they first assign each mesh vertex a random color, then flatten and resample each output neighborhood into a regular grid and determine each output vertex color by minimize an energy function

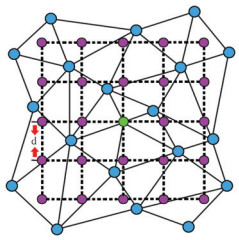


Figure 3: Neighborhood template. The green circle indicates the current vertex to be synthesized, the blue mesh indicates the flattened patch of the mesh, and the purple grid indicates neighborhood template.

$E(\mathbf{x})$:

$$E(\mathbf{x}) = E_t(\mathbf{x}; \mathbf{z}_p) + \lambda E_c(\mathbf{x}; \mathbf{u}) \tag{6}$$

where E_t measures local neighborhood similarity across the current active subset $X^\dagger$ of the output, E_c imposes color constrains (details can be found in Section 4 of [8]), and λ weighs these two different energy terms according to user preference. E_t can be calculated by:

$$E_t(\mathbf{x}; \mathbf{z}_p) = \sum_{p \in X^\dagger} |\mathbf{W}_p \mathbf{x} - \mathbf{z}_p|^2 \tag{7}$$

where $\mathbf{z}_p$ indicates the most similar input neighborhood to each flatten neighborhood $\mathbf{W}_p\mathbf{x}$, $\mathbf{W}_p$ is a per-vertex sparse interpolation matrix relating resampled regular grid vertices $\mathbf{x}_p$ to mesh vertices $\mathbf{x}$.

Differently from their implementation, our input texture sample is 3D geometric texture extracted before, so extracted texture sample must be disposed firstly.

Both the geometric texture and the base mesh should be flattened and resampled to construct neighborhood in regular grid in order to perform pixel-wise comparison. Here we utilize the same method in [17] to construct neighborhood template (Figure 3). For a vertex p to be synthesized, We first orthographically project the triangles adjacent to p onto its local texture coordinate system, and then grow the flattened patch by adding triangles one at a time in order of increasing distance form p until the neighborhood template is fully covered. The distance between adjacent sample points d is determined separately by triangles in different texture regions. In our implementation, we use $d = \sqrt{2 \times A}$ to adapt the size of each geometric texture region, where A is average triangle area of each region, see Figure 3. Thus, we modify E_t as follows:

$$E_t(\mathbf{x}; \mathbf{z}) = \sum_{p \in X^\dagger} |\mathbf{W}_1 \mathbf{x} - \mathbf{W}_2 \mathbf{z}|^2 \tag{8}$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ are relative per-vertex sparse interpolation matrix of output mesh vertices $\mathbf{x}$ and extracted texture vertices $\mathbf{z}$ as discussed before.

We also make a slight modification of solving in [5] in our method which is also algorithmically similar to Expectation-Maximization. In M-step we use approximate nearest neighbor (ANN) matching [14] to accelerate the matching process, and in E-step we use least square to optimize texture appearance.

4. EXPERIMENTAL RESULTS

We demonstrate the application of our method to several models as follows. We compare simply scale, Kraevoys′ method and our method respectively.

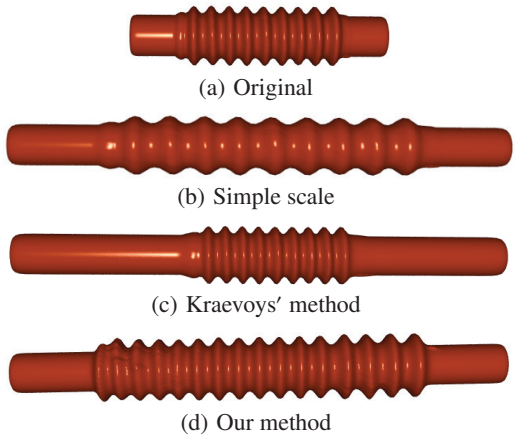


Figure 4: Canal (7778 vertices). The shape and the relative scale of the texture are well preserved by our method.

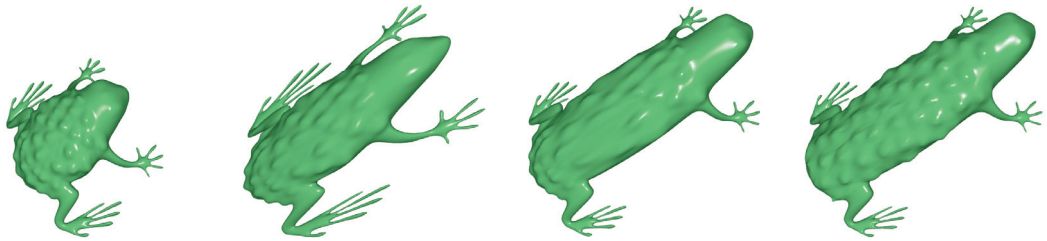
The first example involves twisting geometric texture, see Figure 4. Figure 4(b) shows simply scale greatly changed the size of each screw thread. Figure 4(c) shows Kraevoys′ method preserves the size of each screw thread, but the relative size of screw thread part in the model is greatly changed. Figure 4(d) shows our method not only preserve the size of each screw thread, but also guarantee the relative size of screw thread part.

The third example in Figure 5 shows a model which contains two kinds of textures. Figure 5(d) shows our method retains the shape and relative scale of both kinds of textures of the model. Simply scale in Figure 5(b) and Kraevoys′ method in Figure 5(c) still can not get a rational result.

The forth example in Figure 6 shows a frog model which has a relatively complicated base mesh. Simple scale in Figure 6(b) can not preserve features of toes and head parts of the frog; Kraevoys′ method in Figure 6(c) preserves features better, but because the vulnerability of the texture on back is relatively smaller, the shape of the texture is greatly distorted. Our method in Figure 6(d) preserves features and relatively retains the appearance of the texture.

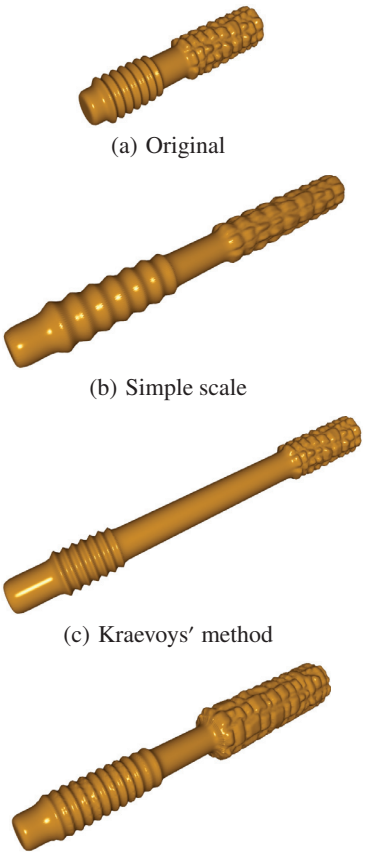
The fifth example in Figure 7 shows a crocodile model. Figure 7(b) shows simple scale distorts features of legs and head parts. Kraevoys′ method in Figure 7(c) shows that because the vulnerability of texture part is relatively larger, the head and the tail parts of the crocodile are scaled much longer, even the displacement of texture is slight. Figure 7(d) shows our method preserves features and textures much better.

We implemented our algorithm on a PC with a Core2 3.00GHz CPU using C++. The example models are about 10K vertices and texture parts are about 1K-4K vertices. The texture segmentation time is less than 1 second, the texture extraction process is about 1-2 seconds, not including the time spent on flattening and resampling textures. Due to different mesh structures and vertex numbers, the cost of flattening and resampling process is greatly different. In our implementation, textured meshes about 3K vertices can be flattened and resampled in 2-3 seconds, and base models about 10K vertices need 10-20 seconds differently. Each E-M synthesis step takes 2-5 seconds due to the resolution of texture sample, and generally 25-30 steps can generate an acceptable result.



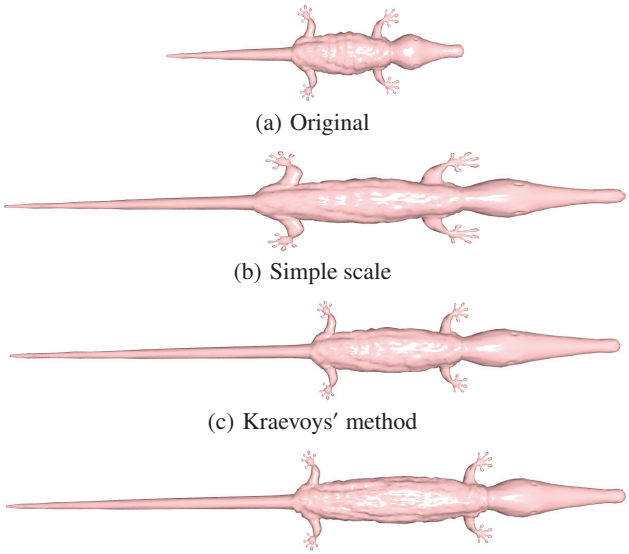
(a) Original (b) Simple scale (c) Kraevoys′ method (d) Our method

Figure 6: Frog (23248 vertices). Features and the shape of the texture are well preserved by our method.



(a) Original (b) Simple scale (c) Kraevoys′ method (d) Our method

Figure 5: Model with two kinds of textures (15588 vertices). Both kinds of textures are well preserved by our method.



(a) Original (b) Simple scale (c) Kraevoys′ method (d) Our method

Figure 7: Crocodile (8176 vertices). Features and relative scale of head and tail parts of the crocodile are well preserved by our method.

5. CONCLUSIONS

Resizing provides an effective mechanism for creating rich geometric content. In this paper, we have presented an efficient algorithm to resize models with geometric textures, which requires very little user intervention. Our method can preserve geometric features along with the shape and relative scale of the geometric textures of the model. Our method augments the set of tools that simplify the creation of 3D geometry, enabling reuse and redesign of existing models.

Our method combines mesh segmentation, geometric texture transfer and mesh resizing technology, and the effect of our method can be improved with the amelioration of each technology. Firstly, we use expectation and variance of curvature in 1-ring neighborhood as features for mesh segmentation, and this scheme could be extended to other feature measures to improve the segmentation. Secondly, one of the limitations is our method is more suitable for dealing with elaborated models. In case the triangles of a geometric feature is coarse and the scale of resizing is large, some unnatural result may appear. If topology is allowed to be changed, a subdivision pretreatment can be applied firstly. Finally, our anisotropic resizing of models can be further extended for more application, such as skeleton-based deformation [19] with geometric texture. We will

investigate these problems as future works.

6. ACKNOWLEDGMENTS

This work was done when Lin-Chen was working as a visiting student at Visual media Research center, Tsinghua University. Furthermore, we appreciate the suggestions of the anonymous reviewers. This work is supported by National Key Basic Research and Development Plan (2006CB303102) and National Natural Science Foundation (60703028).

7. REFERENCES

[1] S. Avidan and A. Shamir. Seam carving for content-aware image resizing. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 10, New York, NY, USA, 2007. ACM.

[2] P. Bhat, S. Ingram, and G. Turk. Geometric texture synthesis by example. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 41–44, New York, NY, USA, 2004. ACM.

[3] H. Biermann, I. Martin, F. Bernardini, and D. Zorin. Cut-and-paste editing of multiresolution surfaces. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 312–321, New York, NY, USA, 2002. ACM.

[4] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 317–324, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[5] J. Han, K. Zhou, L.-Y. Wei, M. Gong, H. Bao, X. Zhang, and B. Guo. Fast example-based surface texture synthesis via discrete optimization. *Vis. Comput.*, 22(9):918–925, 2006.

[6] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 105–114, New York, NY, USA, 1998. ACM.

[7] V. Kraevoy, A. Sheffer, A. Shamir, and D. Cohen-Or. Non-homogeneous resizing of complex models. *ACM Trans. Graph.*, 27(5):1–9, 2008.

[8] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. Texture optimization for example-based synthesis. *ACM Transactions on Graphics, SIGGRAPH 2005*, August 2005.

[9] Y.-K. Lai, S.-M. Hu, D. X. Gu, and R. R. Martin. Geometric texture synthesis and transfer via geometry images. In *SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling*, pages 15–26, New York, NY, USA, 2005. ACM.

[10] Y.-K. Lai, S.-M. Hu, R. R. Martin, and P. L. Rosin. Rapid and effective segmentation of 3d models using random walks. *Computer Aided Geometric Design, article in Press*, 2009.

[11] Y.-K. Lai, Q.-Y. Zhou, S.-M. Hu, and R. R. Martin. Feature sensitive mesh segmentation. In *SPM '06: Proceedings of the 2006 ACM symposium on Solid and physical modeling*, pages 17–25, New York, NY, USA, 2006. ACM.

[12] S.-l. Lien and J. T. Kajiya. A symbolic method for calculating the integral properties of arbitrary nonconvex polyhedra. In *IEEE CG&A*, pages 35–41, New York, NY, USA, October, 1984. IEEE.

[13] G. J. McLachlan and T. Krishnan. The em algorithm and extensions. Wiley series in probability and statistics, 1997.

[14] D. Mount and S. ARYA. Ann: A library for approximate nearest neighbor searching. 2006.

[15] A. Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6):1539–1556.

[16] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184, New York, NY, USA, 2004. ACM.

[17] L.-Y. Wei and M. Levoy. Texture synthesis over arbitrary manifold surfaces. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 355–360, New York, NY, USA, 2001. ACM.

[18] H. Yamauchi, S. Lee, Y. Lee, Y. Ohtake, A. Belyaev, and H.-P. Seidel. Feature sensitive mesh segmentation with mean shift. In *SMI '05: Proceedings of the International Conference on Shape Modeling and Applications 2005*, pages 238—245, Washington, DC, USA, 2005. IEEE Computer Society.

[19] H.-B. Yan, S. Hu, R. R. Martin, and Y.-L. Yang. Shape deformation using a skeleton to drive simplex transformations. *IEEE Transactions on Visualization and Computer Graphics*, 14(3):693–706, 2008.

[20] Y.-L. Yang, Y.-K. Lai, S.-M. Hu, and H. Pottmann. Robust principal curvatures on multiple scales. In K. Polthier and A. Sheffer, editors, *SGP 2006: 4th Eurographics Symposium on Geometry processing*, pages 223–226. Eurographics Association, 2006.

[21] S. Zelinka and M. Garland. Similarity-based surface modelling using geodesic fans. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 204–213, New York, NY, USA, 2004. ACM.