SCIENCE CHINA
Information Sciences

• RESEARCH PAPERS •

December 2010 Vol. 53 No. 12: 2441–2451
doi: 10.1007/s11432-010-4100-z

Anisotropic resizing and deformation preserving geometric textures

CHEN Lin* & MENG XiangXu

School of Computer Science and Technology, Shandong University, Jinan 250101*, China*

Received July 7, 2009; accepted April 28, 2010

Abstract Model resizing is a useful technique for model reuse. Non-homogeneous model resizing introduced by Kraevoy et al. 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 to be extremely difficult for resizing 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 reconstructed 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. Our method can also easily be used in general deformation process.

Keywords geometric texture, anisotropic resizing, texture transfer

Citation Chen L, Meng X X. Anisotropic resizing and deformation preserving geometric textures. Sci China Inf Sci, 2010, 53: 2441–2451, doi: 10.1007/s11432-010-4100-z

1 Introduction

Digital 3D models have 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 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 in several orthogonal directions or dimensions.

Resizing by simply applying a global scaling usually causes distortions of various parts and features. Preserving important geometric features while resizing have been put forward. However, many practical objects contain various geometric textures. For these kinds of models, such methods alone cannot produce reasonably scaled models because the regions with geometric textures will significantly affect the estimation of vulnerability, while the overall perceived shape is similar. Our method explicitly deals with this problem: geometric textures are separated from the underlying surfaces using segmentation and finally resynthesized to produce visually similar results after non-homogenous scaling. By utilizing the natural correspondence before and after scaling, our method improves the non-homogeneous scaling of models with various geometric textures.

*Corresponding author (email: chenlin@mail.sdu.edu.cn)

This paper is an extended version of our previous work [1] which was accepted as a short paper at a conference. Compared with our previous paper, the results are improved in several important places. First, in texture segmentation process, we avoid the tedious user interactive work except a parameter to be specified. Our new method uses Gaussian curvature and mean curvature to estimate more appropriate local geometric attributes as segmentation criterion. The second improvement is that we encode geometric textures based on Laplacian coordinates instead of displacement map used previously, thereby preserving the geometric feature better. Thirdly, in texture synthesis process, we deal with the extracted geometric textures not only by flattening and resampling but also applying an inverse texture synthesis process, which improves both the quality and speed of the texture synthesis process. Lastly, we extend our method to general deformation process, and results show that our method greatly preserves the geometric textures. We briefly summarize various related techniques in section 2. An overview of our algorithm and details and experimental results are given in section 3. Conclusions and discussions of the future work are given in section 4.

2 Related works

Content-aware resizing was initially proposed and studied under the background of image and video processing. Content-aware image resizing was first proposed by Avidan and Shamir [2] based on a seam carving algorithm. Huang et al. [3] presented a more efficient algorithm for seam based content-aware image resizing, which supports real-time image resizing. Many algorithms such as that in [4] had considered a generalized problem of video resizing/retargeting.

However, studies on optimal resizing of geometric models are much fewer. Kraevoy et al. [5] first proposed nontrivial resizing of 3D geometric 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. Wang et al. [6] gave a method based on surface deformation. They measured per-edge sensitivity and computed the adjusted position for each vertex to guide non-homogeneous resizing. These methods 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, and thus cannot be scaled properly. The shape and the relative scale of the texture region will be greatly changed and such models are not suitable for reuse.

Researches on geometric textures or details have been carried on for many years, and some of them are related to our work. For texture segmentation, an extremely fast algorithm had been proposed in [7]. Lai et al. [8] presented a clustering-based top-down hierarchical segmentation algorithm, which particularly utilized integral invariants and their statistics for segmentation of shapes with significantly varied geometric textures. However, their statistic approach is relatively simple and unable to deal with very complicated textures. Yamauchi et al. [9] developed a mesh segmentation method that combines a procedure for clustering and adopts mean shift technique for clustering scattered data. For texture synthesis, Wei et al. [10] presented a method which created a small rectangular neighborhood with each vertex and searched a sample texture for similar neighborhoods. Lai et al. [11] proposed an algorithm to transfer geometric textures using geometry images. The method given by Han et al. [12] synthesized textures by discrete optimization. They combined texture optimization and synthesis by neighborhood sampling to achieve controllable, frame-coherent texture animation over general input textures and output meshes.

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

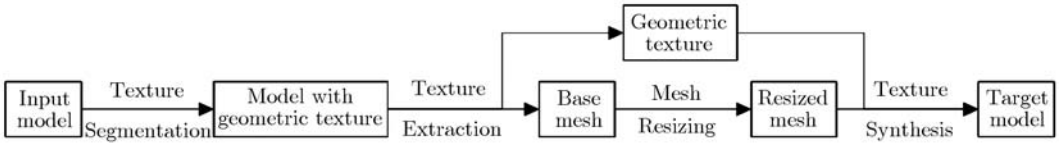

Figure 1 Algorithm overview.

surfaces before and after resizing, our method, under the premise of keeping the relative scale of the geometric texture part in the model, preserves the details of the geometric texture as far as possible. Our method is outlined in Figure 1. The whole process is as follows:

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 regions of the model. We first calculate integral invariants to estimate geometric attributes of mesh vertices. Then, we compute the expectation and variance of geometric attributes in the n-ring neighbor of each vertex as scattered data, and employ a mean shift clustering procedure to segment different parts of geometric textures.

Next, we apply a texture extraction process to the segmented regions with different geometric textures. 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. After obtaining the base mesh, we then encode the geometric textures based on Laplacian coordinates.

Now, the base mesh can be resized directly without affecting geometric texture. In our implementation, we employ Kraevoy's non-homogenous resizing method.

We then synthesize the geometric texture to the base mesh. Similar to Han's synthesis method, our method is also an optimization process. In our method, input texture is not texture sample but extracted geometric texture. Specifically, to get the texture sample, a flattening and resampling operation and an inverse texture synthesis process should be applied to the extracted geometric textures. The solving of both processes is algorithmically similar to expectation-maximization (EM) [13].

3.2 Texture segmentation

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

Like our previous work, we still use the mean shift clustering method to segment the geometric textures. An improvement in this extended work is that the user does not need to adjust the region of texture parts tediously. Instead, user can only specify the size of the sample texture which at least comprises more than one texel. The size of sample texture also can be used in texture resynthesis process for improvement. Our segmentation process is detailed as follows.

We should first decide which surface descriptors can be used to segment textures. Actually, according to different kinds of geometric textures, different attributes will be more sensitive and appropriate. In this extended work, we choose vertex coordinates combined with local Gaussian curvature and mean curvature as the attributes of our segmentation criterion instead of two principal curvatures in our previous work, because these two kinds of curvatures can differentiate geometric textures and geometric features better. Curvatures can be estimated by integral invariant computed by the method in [14]. Curvatures of each vertex cannot be used in segmentation directly, and local geometric attributes should be estimated as clustering criterion. In our method, we calculate the expectation $E(x_i)$ and variance $D(x_i)$ of curvature

x_i of vertex i in its n-ring neighborhood $N_n(i)$:

$$E(x_i) = \frac{1}{m} \sum_{j \in N_n(i)} x_j, \tag{1}$$

$$D(x_i) = \frac{1}{m} \sum_{j \in N_n(i)} (x_j - E(x_i))^2, \tag{2}$$

where x_j is principal curvature of vertex j in n-ring neighborhood of i, and $m = \#N_n(i)$ is the number of the neighbors (valence) of i, $n = \lceil (\sqrt{s} - 1)/2 \rceil$ and s is the size of sample texture specified by the user.

To implement clustering (segmentation), we employ mesh shift, a powerful general purpose procedure for non-parametric clustering of scattered data. 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.

We still use separable kernel density estimation method (Parzen-window estimation) to estimate the probability density function:

$$\widehat{f}(x) = \frac{1}{N h_p^{d_p} h_q^{d_q} h_t^{d_t}} \sum_{i=1}^{N} K_1\left(\frac{p - p_i}{h_p}\right) K_2\left(\frac{q - q_i}{h_q}\right) K_3\left(\frac{t - t_i}{h_t}\right), \tag{3}$$

where h_p, h_q, h_t are the size of each window, p, q, t are the center of each window, and h_p, h_q, h_t can be adjusted by users to improve the effect of segmentation. Additionally, window size can also be used to control patch number segmented. The larger the window size, the more segmented patches.

In our implementation, we choose Epanechnikov kernel which has a simple profile:

$$k_E(x) = \begin{cases} 1 - x, & 0 \leqslant x < 1, \\ 0, & x > 1. \end{cases}$$

We run our segmentation process over each vertex v_i of Mesh M to find its convergent point. The convergent point determines the sort of geometric textures it belongs to. Specifically, we first initialize the window center attributes by assigning $p = p_i, q = q_i, t = t_i$, and then we detect all the points until they are convergent or exceed an appointed iterative times. Figure 2 shows the segmented texture of the frog model. 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 curvature expectation counterpart q_i satisfies $\|q_i - q\| < h_q$.
3. For each detected vertex q_i, we check whether its local 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.

We adapted the fairing method in [15] because of the features volume preservation, and hard and soft constraints on positions of vertices of the mesh. This method is based on backward Euler method and extension of simulation of heat equation as follows:

$$(I - \lambda dt L) X^{n+1} = X^n, \tag{4}$$

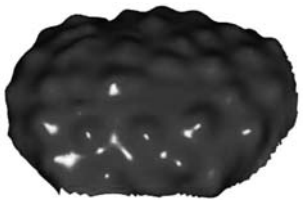

Figure 2 Segmented texture of the frog model.

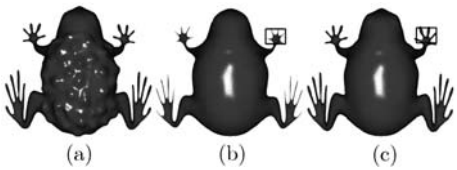

Figure 3 Smooth input model, uniformly and non-uniformly. (a) Original; (b) uniform; (c) non-uniform.

where λdt is a constraint factor, and X^n is vertex coordinates in iteration n.

Obviously, geometric features may also be filtered during uniformly implicit fairing, as Figure 3(b) shows. In our approach, for the sake of preserving important features of the model, we set a large λdt on geometry texture regions and a small one to other regions, specifically 10 and 0 respectively. Figure 3(c) shows the base mesh we get by our method. The fairing operation is iterated until the altering of vertices coordinates of texture regions is less than a threshold or the number of iteration exceeds a specified times. Apparently, this implicit fairing operation only modifies coordinates of model vertices. The topology of the model is unchanged. The one-to-one correspondence between the base mesh and the original mesh is well kept.

Different from our previous work, we encode the geometric textures based on Laplacian coordinates similarly to the work of coating transfer in [16] instead of displacement map. More precisely, we define geometric details as the differences of Laplacian coordinates between the model before and after smoothed. Laplacian coordinate vector approximates mean curvature normal better representing geometric textures. Besides, the least square solution of reconstruction process smoothly distributes the error across the whole domain, and thus provides a graceful appearance. As Figure 4 shows, texture extracted as Laplacian coordinate in Figure 4(b) looks more smooth than displacement map in Figure 4(a).

Let S be the input mesh, and $\tilde{S}$ be the base mesh. And let δ_i^s and $\tilde{\delta}_i^s$ be the Laplacian coordinates of the vertex i in S and $\tilde{S}$, respectively. We encode the geometric texture details at vertex i as follows:

$$\xi_i = \delta_i^s - \tilde{\delta}_i^s. \tag{5}$$

In this paper, the Laplacian coordinates are computed by

$$\delta^s = L^s V^s, \tag{6}$$

where L^s is the topological Laplacian of the source mesh. It can be defined by

$$(L)_{ij}^s = \begin{cases} 1, & i = j, \\ -1/d_i, & (i,j) \in E^s, \\ 0, & \text{otherwise}, \end{cases}$$

where d_i is the valence of vertex i.

3.4 Texture resynthesis

The extracted base mesh can be resized without affecting the geometric texture. We choose Kraevoy's non-homogeneous resizing method to process the base model. Our geometric texture preservation method is extended to deformation process in this paper. With regard to models with geometric textures, deformation process may also cause geometric texture distortion, especially when scaling is involved. Our method can be extended by applying deformation operation process to base mesh instead of resizing. In this paper, we adopt deformation method in [17].

To eliminate 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

Figure 4 Different representations of geometric texture. (a) Texture is extracted as displacement map; (b) texture is extracted as Laplacian coordinate.

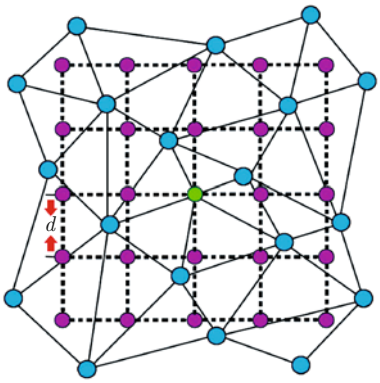

Figure 5 Neighborhood template.

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 does 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. The process of synthesizing textures back to resized model is detailed as follows.

In our approach, both the geometric texture and the base mesh should firstly be flattened and resampled to construct neighborhood in regular grid in order to perform pixel-wise comparison. Here we utilize the same neighborhood creating method as that of Wei's to construct neighborhood template, as Figure 5 shows, where 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. 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 according to its distance from p increasingly 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 5).

Instead of simply flattening and resampling the geometric texture, in this extended work we apply the inverse texture synthesis method discussed in [18] to get a sample texture. The inverse synthesis process can be cast as an optimization problem by minimizing the following function:

$$\Phi(\boldsymbol{W};\boldsymbol{x};\boldsymbol{z};\boldsymbol{w}) = \frac{1}{|X^{\dagger}|}\sum_{p\in X^{\dagger}}|\boldsymbol{W}_p\boldsymbol{x}_p(\boldsymbol{w}_p)-\boldsymbol{z}_p|^2 + \frac{\alpha}{|Z^{\dagger}|}\sum_{q\in Z^{\dagger}}|\boldsymbol{W}_q\boldsymbol{x}_q(\boldsymbol{w}_q)-\boldsymbol{z}_q|^2, \tag{7}$$

where $\boldsymbol{W}$ is per-vertex sparse interpolation matrix of geometric texture vertices $\boldsymbol{x}$ for initial texture region X, $\boldsymbol{z}$ represents the sample values for output texture region Z, $\boldsymbol{w}$ is the orientation field for X, q/p runs through a subset $X^{\dagger}/Z^{\dagger}$, $\boldsymbol{x}_p/\boldsymbol{z}_q$ indicates the spatial neighborhood around p/q, $\boldsymbol{z}_p/\boldsymbol{x}_q$ is the most similar neighborhood with respect to $\boldsymbol{x}_p/\boldsymbol{z}_q$, and α is a user tunable weighting which $\alpha = 0.01$. This function can be solved algorithmically like Expectation-Maximization.

Our texture synthesis process is similar to Han's texture synthesis method. Different from their work, our input texture sample is the result calculated by eq. (10). In our implementation, we first assign each mesh vertex a random displacement, then flatten and resample each output neighborhood into a regular grid and determine each output vertex displacement by minimizing an energy function $E(\boldsymbol{x})$:

$$E(\boldsymbol{x}) = E_t(\boldsymbol{x};\boldsymbol{z}_p) + \lambda E_c(\boldsymbol{x};\boldsymbol{u}), \tag{8}$$

where E_t measures local neighborhood similarity across the current active subset $X^{\dagger}$ of the output, E_c imposes displacement constrains, and λ weighs these two different energy terms according to user

preference. E_t can be calculated by

$$E_t(\boldsymbol{x}; \boldsymbol{z}_p) = \sum_{p \in X^\dagger} |\boldsymbol{W}_p \boldsymbol{x} - \boldsymbol{z}_p|^2, \tag{9}$$

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

Our solving process is also algorithmically similar to Expectation-Maximization. In M-step we use approximate nearest neighbor (ANN) matching [19] to accelerate the matching process, and in E-step we use least square to optimize texture appearance.

3.5 Texture reconstruction

After the texture synthesis process, vertex j on the target mesh is assigned a corresponding difference vector ξ_j. However, this difference cannot be simply added to the vertex. Because this difference is a vector, and the orientation of a point on the sample patch is not the same as that on the target mesh, we should rotate the difference vector with respect to the target to compensate for the different local surface orientations. Then the new Laplacian coordinate $\tilde{\delta}_j^t$ of vertex j can be obtained by

$$\tilde{\delta}_j^t = \delta_j^t + R \cdot \xi_j, \tag{10}$$

where R is the rotation of ξ_j to match the orientation of the corresponding vertex in the target mesh. The rotation R can be calculated by the directions of the normal of the sample patch and the target compensation.

Then, the texture of the synthesized mesh can be obtained by solving the following equation system:

$$L^t \tilde{V}^t = \tilde{\delta}^t, \tag{11}$$

where V^t denotes the synthesized texture vertices of the result mesh, and $\tilde{\delta}^t$ denotes the adjusted Laplacian coordinates of the target mesh obtained from eq. (13).

In order to uniquely reconstruct the global mesh, we need to solve a full-rank linear system. Assuming the target mesh is connected, we need to specify the Cartesian coordinates of at least one vertex or add some other constraints to solve the equation system. In this paper, to minimize effects on final results and preserve silhouette of target model, we choose vertices adjacent to the textured region as constraints of Possion Function.

4 Experimental results

We demonstrate the anisotropic resizing and deformation application of our method to several models as follows. We compare simply-scaling, Kraevoy's method and our method.

The first example involves twisting geometric texture (Figure 6). Figure 6(b) shows that simple scale greatly changes the size of each screw thread. Figure 6(c) shows that Kraevoy's method preserves the size of each screw thread, but the relative size of screw thread part in the model is greatly changed. Figure 6(d) shows that our method not only preserves the size of each screw thread, but also guarantees the relative size of screw thread part.

The second example in Figure 7 shows a CAD model. Similar to the first example, simply scaling in Figure 7(b) remodels the textures, Kraevoy's method in Figure 7(c) shifts the relative size of texture part, and our method in Figure 7(d) preserves both the shape and the relative size of texture part in the model.

The third example in Figure 8 shows a mushroom model that contains two parts of textures. Figure 8(b) shows that simply scaling makes the top part of the model stretched excessively; Kraevoy's method in Figure 8(c) makes the texture of the model distorted greatly and our method in Figure 8(d) keeps both the shape of the model and the textures look fine.

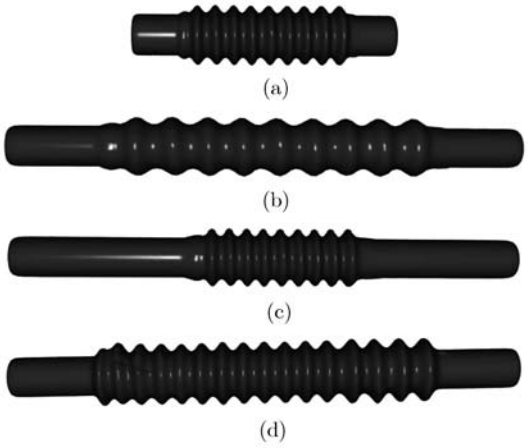

Figure 6 Canal (7778 vertices). (a) Original; (b) simple scale; (c) Kraevoys' method; (d) our method.

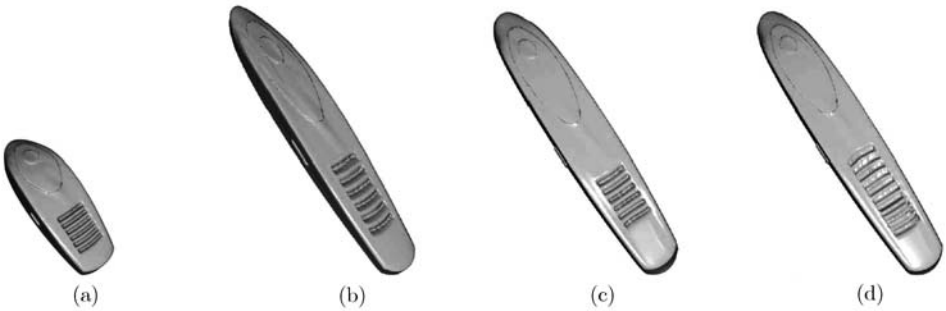

Figure 7 Electronic equipment (31564 vertices). (a) Original; (b) simple scale; (c) Kraeovys' method; (d) our method.

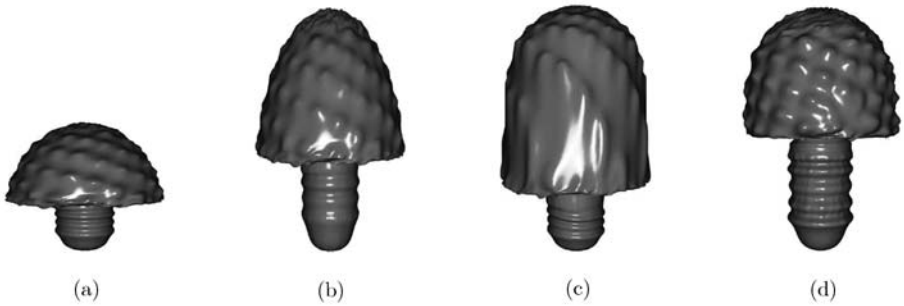

Figure 8 Mushroom (28849 vertices). (a) Original; (b) simple scale; (c) Kraeovys' method; (d) our method.

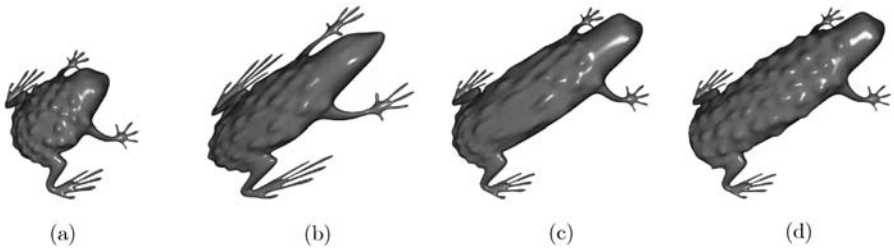

Figure 9 Frog (23248 vertices). (a) Original; (b) simple scale; (c) Kraeovys' method; (d) our method.

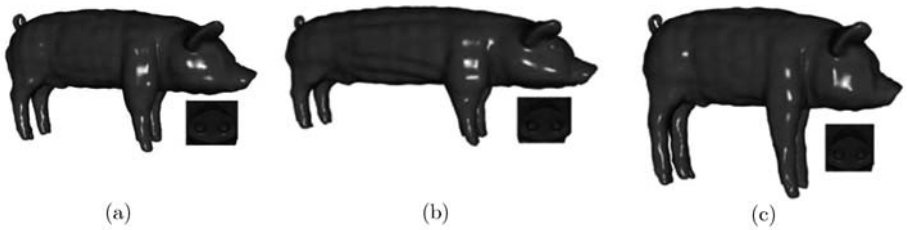

Figure 10 Pig (24376 vertices). (a) Original; (b) resizing along x direction; (c) resizing along y direction.

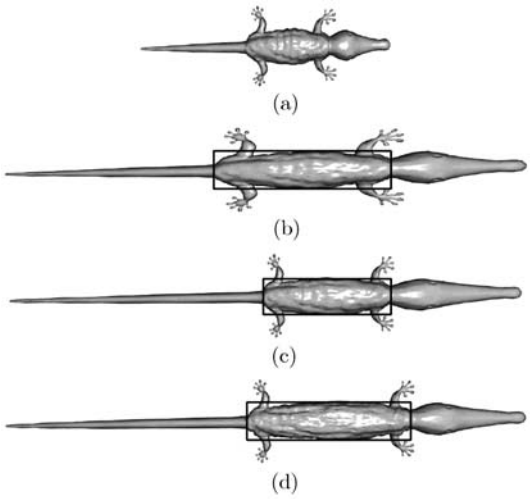

Figure 11 Crocodile (8176 vertices). (a) Original; (b) simple scale; (c) Kraeovys' method; (d) our method.

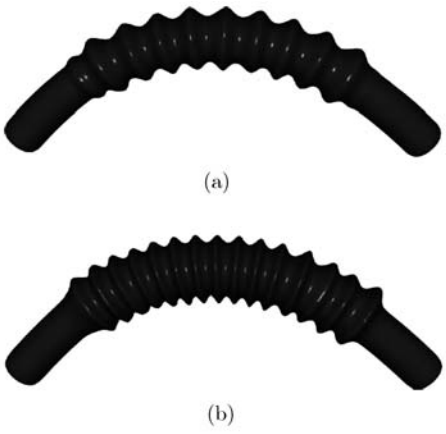

Figure 12 Deformation with scale and rotation. (a) Simply deformation; (b) our method.

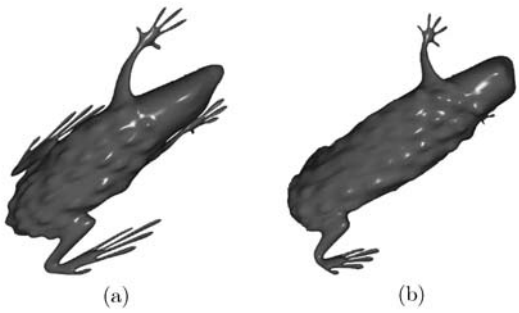

Figure 13 Deformation with scale and twist. (a) Simply deformation; (b) our method.

The fourth example in Figure 9 shows a frog model which has a relatively complicated base mesh. Simple scale in Figure 9(b) cannot preserve features of toes and head parts of the frog; Kraevoys' method in Figure 9(c) preserves features better, but because the vulnerability of the texture on back is relatively small, the shape of the texture is greatly distorted. Our method in Figure 9(d) preserves features and relatively retains the appearance of the texture.

The fifth example in Figure 10 shows a pig model which has more complex features on its legs and head. Figures 10(b) and 10(c) show resizing results of the model along X axis and Y axis respectively. We can see that not only the geometric texture on the body but geometric features on legs and nose are well preserved.

Figure 11 shows a crocodile model. Figure 11(b) shows simple scale distorts features of legs and head parts. Kraevoys' method in Figure 11(c) shows that because the vulnerability of texture part is relatively large, the head and the tail parts of the crocodile are scaled much longer, and even the displacement of texture is slight. Figure 11(d) shows that our method preserves features and textures much better.

Figures 12 and 13 show the results with model deformation, scale, rotation and twist involved. Figure 12(a) and Figure 13(a) show the results of simple deformation, Figure 12(b) and Figure 13(b) show the results of our method. Both figures show that our method preserves geometric textures much better.

We implemented our algorithm on a PC with a Core2 3.00 GHz 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 s, and the texture extraction process is about 1–2 s, 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 s, and base models about 10K vertices need 10–20 s differently. Each *E-M* synthesis step takes 2–5 s due to the resolution of texture sample, and generally 25–30 steps can generate an acceptable result.

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 via 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 recognize different parts of geometric textures. Secondly, in our implementation we use pixel-based approach for surface marching, and the searching speed is relatively limited. Maybe patch-based texture synthesis approach can be used here for acceleration. Finally, our method is more suitable for dealing with elaborated models. This is a limitation. In case the triangle of a geometric feature is coarse and the scale of resizing is large, some unnatural results may appear. If topology is allowed to be changed, a subdivision pretreatment can be applied firstly. We will investigate these problems in our future work.

Acknowledgements

This work was supported by the National Key Basic Research and Development Plan (Grant No. 2006CB303102), and the National Natural Science Foundation of China (Grant No. 60703028).

References

1 Chen L, Meng X X. Anisotropic Resizing of Models with geometric textures. In: 2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling. New York, NY: ACM, 2009. 289–294
2 Avidan S, Shamir A. Seam carving for content-aware image resizing. In: SIGGRAPH'07: ACM SIGGRAPH 2007 papers. New York, NY: ACM, 2007. 10
3 Huang H, Fu T N, Rosin P L, et al. Real-time content-aware image resizing. Sci China Ser F-Inf Sci, 2009, 52: 172–182
4 Zhang Y F, Hu S M, Ralph M R. Shrinkability maps for content-aware video resizing. Comput Graph Forum, 2008, 27: 1797–1804
5 Kraevoy V, Sheffer A, Shamir A, et al. Non-homogeneous resizing of complex models. ACM Trans Graph, 2008, 27: 1–9
6 Wang K P, Zhang C M. Content-aware model resizing based on surface deformation. Comput Graph, 2009, 33: 433–438
7 Lai Y K, Hu S M, Martin R R, et al. Rapid and effective segmentation of 3D models using random walks. Comput Aided Geom Des, 2009, 26: 665–679

8 Lai Y K, Zhou Q Y, Hu S M, et al. Feature sensitive mesh segmentation. In: SPM '06: Proceedings of the 2006 ACM Symposium on Solid and Physical Modeling. New York, NY: ACM, 2006. 17–25
9 Yamauchi H, Lee S, Lee Y J, et al. Feature sensitive mesh segmentation with mean shift. In: SMI '05: Proceedings of the International Conference on Shape Modeling and Applications 2005. Washington, DC: IEEE Computer Society, 2005. 238–245
10 Wei L Y, Levoy M. Texture synthesis over arbitrary manifold surfaces. In: SIGGRAPH '01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques. New York, NY: ACM, 2001. 355–360
11 Lai Y K, Hu S M, Gu D X, et al. Geometric texture synthesis and transfer via geometry images. In: SPM '05: Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling. New York, NY: ACM, 2005. 15–26
12 Han J W, Zhou K, Wei L Y, et al. Fast example-based surface texture synthesis via discrete optimization. Vis Comput, 2006, 22: 918–925
13 McLachlan G J, Krishnan T. The EM algorithm and extensions. In: Wiley Series in Probability and Statistics, Hoboken, NJ: John Wiley & Sons Inc., 1997
14 Yang Y L, Lai Y K, Hu S M, et al. Robust principal curvatures on multiple scales. In: Konrad P, Alla S, eds. SGP '06: 4th Eurographics Symposium on Geometry Processing, Eurographics Association, Cagliari, Sardina, Italy, 2006. 223–226
15 Desbrun M, Meyer M, Schröder P, et al. 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. New York, NY: ACM Press/Addison-Wesley Publishing Co., 1999. 317–324
16 Sorkine O, Cohen-Or D, Lipman Y, et al. Laplacian surface editing. In: SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing. New York, NY: ACM, 2004. 175–184
17 Yan H B, Hu S M, Martin R R, et al. Shape deformation using a skeleton to drive simplex transformations. IEEE Trans Vis Comput Graph, 2008, 14: 693–706
19 Wei L Y, Han J W, Zhou K, et al. Inverse texture synthesis. In: SIGGRAPH '08: ACM SIGGRAPH 2008 papers. New York, NY: ACM, 2008. 1–9
19 Mount D, Arya S. ANN: a library for approximate nearest neighbor searching. 2006