

Consistent Mesh Parameterizations and Its Application in Mesh Morphing

Jin-Bey Yu*

Jung-Hong Chuang[†]

Department of Computer Science and Information Engineering
National Chiao Tung University
Hsinchu, Taiwan, Republic of China

Abstract

The correspondences establishment for a set of models is a versatile algorithm in computer graphics and geometry processing, which in general counts on a lot of specification by users to build a common dissection with a set of feature points between the input models.

We propose a novel method to compute the consistent parameterizations for multiple models with a little user-control. Our parameterization scheme performs a geometric-stretch optimization to optimize the parameterization of each patch. The alignment of feature correspondences for the result of parameterizations is also considered using a foldover-free warping algorithm. Uniform or adaptive remeshing can be performed to yield a set of semi-regular meshes. Moreover, geometry images can be generated easily with the parameterizations to store the geometric information in a simple grid structure that benefits a number of applications, including normal mapping, texture mapping, rendering, level-of-detail, compression, and morphing. We also demonstrate the mesh morphing application between two or more objects in both spatial and wavelet domain based on the correspondence established by a simple common dissection and remeshing.

Keywords: mesh parameterization, mesh simplification, geometric stretch, remeshing, geometry images, mesh morphing

1 Introduction

Many applications in visualization and computer graphics require models of 3D surface geometry. The most commonly used representation is the triangle mesh. Such meshes can be the result of careful design by using modeling software, or may come as an output of a scanning device associated with reconstruction. As 3D surface geometry becomes a popular media, more and more meshes are available, coming from a variety of sources including 3D scanners and modeling software. Although these meshes capture geometry accurately, their sampling quality is usually far from ideal for subsequent applications. For instance, these complex irregular meshes are not appropriate for rendering on low-end computers, texture mapping, morphing between different models.

Remeshing, i.e. modifying the sampling and connectivity of a geometry to generate a new mesh which approximates the original mesh, is therefore a fundamental step for efficient mesh processing. The state-of-the-art algorithms of remeshing involve initially dissecting the original irregular meshes into a set of topological disk-like patches, called *base domain*. These patches are later parameterized to compute a bijection between the 3D surface and the 2D parametric domain.

Parameterizing a mesh amounts to computing a correspondence between a 3D surface mesh and an 2D planar mesh through a piecewise linear mapping. In practice, this piecewise linear mapping is simply defined by assigning each mesh vertex a 2D coordinates (u, v) referring to its position on the planar domain. Such an one-to-one mapping provides a flat parametric domain, allowing users to perform any complex operation directly on the flat domain rather than on the curved surface. Many parameterization techniques are proposed in recent decade.

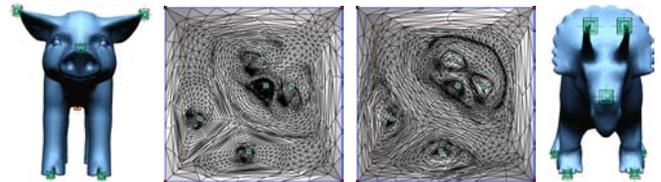


Figure 1: Example of consistent mesh parameterizations.

A single model can be remeshed to yield a (semi-)regular one. Can we extend the idea to multiple models with the same connectivity? The answer is not true unless they have a common dissection in which each patch of one model corresponds exactly to one patch in all other models. In consequence, they will possess a same base domain and their parameterizations will be consistent. The difficulty is that the common dissections are not easily found, because a good dissection of one model might be bad for another model. Previous works [14, 25, 35, 36, 42] leave this problem to users, that is, the user is required to specify a common dissection and many corresponding feature points manually. Besides, many applications such as metamorphosis (or morphing) and DGP (Digital Geometry Processing) applications [36], which require the establishment of correspondences among multiple models, benefit from consistent parameterizations.

In this paper, we propose a novel method to compute the consistent parameterizations for multiple models with a little user-control (Figure 1). The parameterization scheme combines the properties of geometry images with semi-regular remeshing. After users simply specify the initial patches, i.e. the base domain, on the model, our parameterization scheme will perform a geometric-stretch optimization to optimize the parameterization of each patch. The optimization algorithm borrows techniques from Sander et al. [38, 39]. We prevent parametric “foldover” and penalize undersampling by using a coarse-to-fine optimization strategy with geometric-stretch metric. This parameterization scheme can be extended to multiple models and guarantees to yield consistent parameterizations. Extra feature correspondences can also be specified by users. The alignment of feature correspondences during parameterization process is also considered by using a foldover-free warping [12].

Once the consistent parameterizations are obtained, a number of applications can operate directly on the base domain. We perform the remeshing process and obtain a set of semi-regular meshes. The remeshing process can be either uniform or adaptive. Based on the parameterization of the patches, *normal maps*, which capture geometry details, can also be easily resampled. This improves the real-time rendering quality of semi-regular meshes in coarse levels. A multiresolution mesh morphing both in spatial and wavelet domains is also demonstrated as an application of our proposed approach. Besides, we also exploit the idea of geometry images to store our remesh data. The geometry images introduced by Gu et al. [15] support surfaces of arbitrary genus by allowing an arbitrary surface cut. However, the cut topology constrains level-of-detail mip-mapping. In contrast, our reconstruction bases on a simple base domain, and the topology of our polyhedral cuts is fixed and simple. The cuts give rise to symmetry rules at the image boundaries, which enable morphing and better level-of-detail control.

*e-mail: jbyu@csie.nctu.edu.tw

[†]Contact author. e-mail: jhchuang@csie.nctu.edu.tw

2 Related Work

Parameterization. Several schemes have been proposed to flatten a surface region to establish a parameterization over the last ten years in computer graphics. Almost all techniques explicitly aim at producing least-distorted parameterizations, and vary only by an objective function (distortion considering) and the minimization processes used. The main distinction between the functions is how they measure the distance of the parameterization from an isometry (a mapping preserving lengths and angles).

Maillot et al. [34] base their metric on spring-like energies, that is, edge springs of nonzero rest length where rest length corresponds to edge length on the surface. Eck et al. [8] proposed the discrete harmonic map, which assigns non-uniformly spring weights to the mesh edges.

Floater [10] proposed specific weights based on the barycentric maps [40] to improve the area deformations and shape-preserving of the mapping. It guarantees embedding for convex boundaries and the parameterization can be found by solving a linear system. Floater [11] proposed another weighting algorithm. The weights are derived from the mean value theorem for harmonic functions. In addition, it is faster to compute than the shape-preserving parameterizations [10] and have the theoretical advantage.

Hormann and Greiner [20] proposed the MIPS parameterization, which roughly attempts to preserve the ratio of singular values over the parameterization with a hierarchical solver [21].

Sander et al. [39] proposed another non-linear energy for texture stretch distortion. Their stretch metric minimizes undersampling by integrating the sum of squared singular values over the map. Sander et al. [38] showed the stretch metric is related to SAE (Signal-Approximation Error) — the difference between a signal defined on the surface and its reconstruction.

Several researchers introduced methods which satisfy positional constraints approximately or “softly”. Lévy and Mallet [31] combine orthogonality and iso-parametric terms in their metric and allowed user-defined matching of iso-parametric curves. Lévy [30] suggested a method to incorporate soft positional constraints into the parameterization problem.

Eckstein et al. [9] introduced a method that enforces “hard” (exact) positional constraints by deforming an existing embedding while adding a number of Steiner vertices.

Remeshing. Lounsbery [32], Lounsbery et al. [33] have developed a technique that introduces multiresolution analysis to a restricted class of meshes with *subdivision connectivity*. Initial dissection is crucial for remeshing, because it defines the base domain. Eck et al. [8] use a Voronoi-based partition to dissect the input mesh.

MAPS [29] and Normal Meshes [16] employ a mesh simplification algorithm to yield an initial dissection. The whole algorithm for finding an initial dissection is also automatic and more practical than the method in [8]. Based on the normal mesh algorithm, Praun et al. [36] proposed an algorithm for finding the common initial dissection for multiple models.

More recently, Gu et al. [15] introduced *Geometry Images*. The process involves cutting the surface into a disk using a network of cut paths, and then parameterizes the resulting disk onto a square domain [38, 39]. Using this parameterization, the surface geometry is resampled onto the pixels of an image.

Alliez et al. [3] proposed an interactive sampling technique. A mesh is decomposed into a set of maps inserted in a pipeline of signal processing algorithms. The output of this pipeline is a density map, interactively resampled using an error diffusion technique commonly used for gray level image halftoning.

Morphing. The correspondence problem in the mesh morphing is naturally related to parameterizations. The survey of 3D morphing can be referred to [27], and mesh morphing, an extensive re-

view can be found in [1]. Particularly, Alexa pointed out in [1] that the remeshing approach is appealing for morphing applications, because it allows to scale the size of the representation mesh. On the contrary, the conventional merging approach generates a more complicated intermediate representation.

Kanai et al. [24] uses a single patch and the patch will be parameterized by harmonic mapping. The corresponding patch will be aligned and merged to generate a merged mesh as intermediate representation. In their recent works [25], the user first defines a set of corresponding features vertices and applied their approximate shortest path algorithm [23] to find an initial dissection.

The work of Zöckler et al. [42] also requires users to provide the initial dissection as a high-level correspondence. The corresponding patches will be applied foldover-free warping [12] in the parametric domain for detailed level feature correspondences.

Lee et al. [28] utilized their MAPS to the mesh morphing application. Their MAPS scheme supports feature point and line tagging such that the constructed base domains will preserve these features. However, the corresponding features points manually specified by users on both models may lie in different positions in both base domain.

Michikawa et al. [35] proposed a *multiresolution interpolation meshes* representation for mesh morphing. An interface is designed for users to define a common patch layout on both source and target meshes. Each patch is then parameterized and a surface fitting is performed to produce a semi-regular mesh for the intermediate mesh. The method can be extended to multi-target morphing.

3 Consistent Mesh Parameterizations

3.1 Approach overview

Our strategy for consistent mesh parameterizations is as follows. For each of the given meshes, we first consistently partition the meshes into a set of minimum disk-like patches such that the patches from different meshes are in one-to-one correspondence. For meshes of genus 0, the consistent partition is derived by connecting four user-specified *partition points*, which are in correspondence and will serve as patch’s corners. The point connection can be done by applying the Dijkstra’s shortest path algorithm [7] based on geodesic distance. Note that, additional *feature points* in correspondence can be assigned by users for facilitating the vertex-correspondence problem commonly found in mesh morphing.

For each patch, we derive a base mesh by applying a sequence of half-edge collapsing operations. During the half-edge collapsing, all feature points are retained. On the base mesh, we compute an initial embedding of the base mesh by using the mean value coordinates and optimize the resulting embedding using a geometric-stretch metric. To achieve the consistent mesh parameterization among given meshes, all feature points in correspondence are aligned by using a foldover-free warping algorithm. Having the initial embedding of the base mesh with the aligned feature points and the refinement sequence, we perform a coarse-to-fine parameterization to embed all other vertices and then optimize its geometric-stretch metric.

Given a number of meshes with corresponding partition points and feature points specified by users, the steps of the proposed consistent mesh parameterization can be summarized as follows:

1. Consistently partition the meshes into a set of minimum disk-like patches by connecting partition points.
2. For patches in correspondence, we perform a consistent patch parameterization
 - (a) Derive a base mesh for each patch by applying a sequence of half-edge collapsing operations, compute an initial embedding for the base mesh and perform geometric stretch optimization.
 - (b) Align feature points in correspondence on the initial embeddings of all the patches.

- (c) Perform a coarse-to-fine parameterization for each patch with geometric-stretch optimization.

3.2 Consistent mesh partition

Genus-zero meshes. We partition a genus zero mesh into two quadrilateral disk-like patches by requiring users to specify four partition points on the surface and then performing Dijkstra’s shortest path algorithm [7] based on geodesic distance to trace from one partition point to another and form a closed loop. Now with this closed curves, the original mesh is dissected into two patches and the respected faces are well marshaled. Note that, to consistently parameterize the given meshes, the partition points must be in correspondence.

Extending to genus- n meshes. For a genus- n mesh, the number of patches must at least be $2(n+1)$. In the case of genus- n meshes, we specify four partition points to locate the corners of the four patches and eight auxiliary points to guide the tracing of patch’s boundaries.

3.3 Consistent patch parameterization

With consistent mesh partition, each given mesh is partitioned into a set of patches, with the property that each patch has one-to-one correspondence to a patch in each of other meshes. Consistent patch parameterization parameterizes those corresponding patches such that all corresponding feature points are aligned to identical parameter coordinates. Furthermore, stretch minimization over the parameterization is also addressed.

Although the mesh parameterization and remeshing ensure the one-to-one vertex correspondence required for mesh morphing. The methods for parameterization and remeshing, however, determine how well the original shapes are approximated. Minimizing the stretch norm over the parameterization is critical in this respect, but, on a uni-resolution parameterization, it is slow and often converges to bad local minima. Here, a strategy similar to that in Hormann et al. [21], Sander et al. [38, 39] is taken to obtain better performance in both speed and quality.

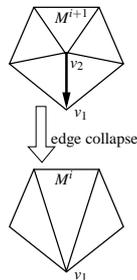
For each patch, we start the parameterization by deriving a bash mesh and its associated embedding. All feature points are retained in the base mesh, and those feature points in correspondence are aligned to have identical parameter coordinates by a fold-free mesh warping. The patch parameterization is finally derived by a coarse-to-fine parameterization with geometric-stretch optimization.

3.3.1 Base mesh derivation

The base mesh for a patch is the coarsest mesh derived after applying the progressive mesh (PM) simplification [18] to the patch.

The PM is derived by simplifying the patch’s mesh using a sequence of half-edge collapsing operations. The quadric error metric [13] that approximates both visual and geometric error is used. The half-edge collapsing operation $(v_1, v_2) \rightarrow v_1$ affects the neighborhood of v_2 as shown on the right figure and leaves the position and attributes of v_1 unchanged. Compared to the full-edge collapsing, half-edge collapsing avoids writing to the vertex buffer during run-time LOD switches, and, in consequence, parametric coordinates of every vertex are the same at all levels.

During simplification, we disallow an edge collapsing of $(v_1, v_2) \rightarrow v_1$ if v_2 is a patch corner (to preserve corners), or if v_2 is on a patch boundary and edge (v_1, v_2) is not on a patch boundary (to preserve boundary straightness), or if v_2 is a feature point (to retain feature points in the initial embedding).



3.3.2 Initial embedding of base meshes

Mean value coordinates. An initial embedding for the base mesh is derived by using mean value coordinates parameterization Floater [11]. The patch’s parameterization builds a bijective mapping u between a 3D mesh and a 2D parametric domain. With mean value coordinates approach, we first fix the parameters for boundary vertices of the quadrilateral patch on a unit square and then derive the function u that satisfies the following equations for each of the interior vertices $v_i, i = 1, 2, \dots, n$:

$$u(v_i) = \sum_{k \in \mathcal{V}(i)} \lambda_{i,k} u(v_k), \quad (1)$$

$$\sum_{k \in \mathcal{V}(i)} \lambda_{i,k} = 1, \quad (2)$$

where $\mathcal{V}(i)$ is the 1-ring neighborhood of the vertex $v_i, \lambda_{i,k} \geq 0$, for each i and k , and n is the number of the vertices of the patch. Equation (1) expresses that v_i as a convex combination of its neighboring points. To derive the mapping u , we first compute the $\lambda_{i,k}, k \in \mathcal{V}(i)$, for each i , that satisfies the properties of barycentric coordinates. Then, we solve the linear system of Equation (1) for $u(v_i)$.

In the simplest case $k = 3$, the weights $\lambda_{i,1}, \lambda_{i,2}, \lambda_{i,3}$ are uniquely determined by (1) and (2) alone; they are the barycentric coordinates of v_i with respect to the triangle $[v_1, v_2, v_3]$, and they are positive. For general k , these weights $\lambda_{i,k}$ can be derived from an application of the mean value theorem for harmonic functions, which suggests calling them mean value coordinates. As shown in Figure 2, the derivation of $\lambda_{i,k}$ for v_i is as follows:

$$\lambda_{i,k} = \frac{w_{i,k}}{\sum_{l \in \mathcal{V}(i)} w_{i,l}}, \quad w_{i,k} = \frac{\tan(\alpha_{k-1}/2) + \tan(\alpha_k/2)}{\|v_{i,k} - v_i\|}, \quad (3)$$

where α_k , the angle at v_i in the triangle $[v_i, v_k, v_{k+1}]$, is defined cyclically. The coordinates approximate harmonic maps by piecewise linear maps over triangulations, in such a way that injectivity is preserved.

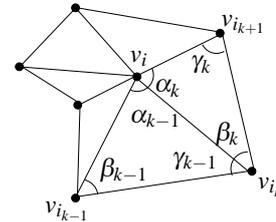


Figure 2: Star-shaped polygon.

The main advantage of mean value coordinates is that the $\lambda_{i,k}$ are always positive, because they are all derived from barycentric coordinates. This guarantees the mapping is bijective and a sparse linear system derived from Equation (1) can be solved robustly by the iterative biconjugate gradient method. In addition, the mean value coordinates are faster to compute than the shape-preserving coordinates [10], and have the theoretical advantages.

Stretch optimization. To optimize the parameterization, we apply a global optimization similar to that in Sander et al. [39]. To minimize geometric-stretch metrics $L^2(M)$ and $L^\infty(M)$, we begin with the mean value coordinates parameterization, and perform several optimization iterations. Within each iteration, the optimization procedure sweeps all vertices in the base mesh and optimize each one in turn. More precisely, they place all vertices in a priority queue ordered by the amount of geometric stretch in their neighborhood. The process stops when the largest change is below a threshold (e.g. 10^{-3}). For each vertex, we minimize the geometric-stretch metric by repeatedly updating its parametric coordinates using bracketed parabolic minimization (instead of the binary line

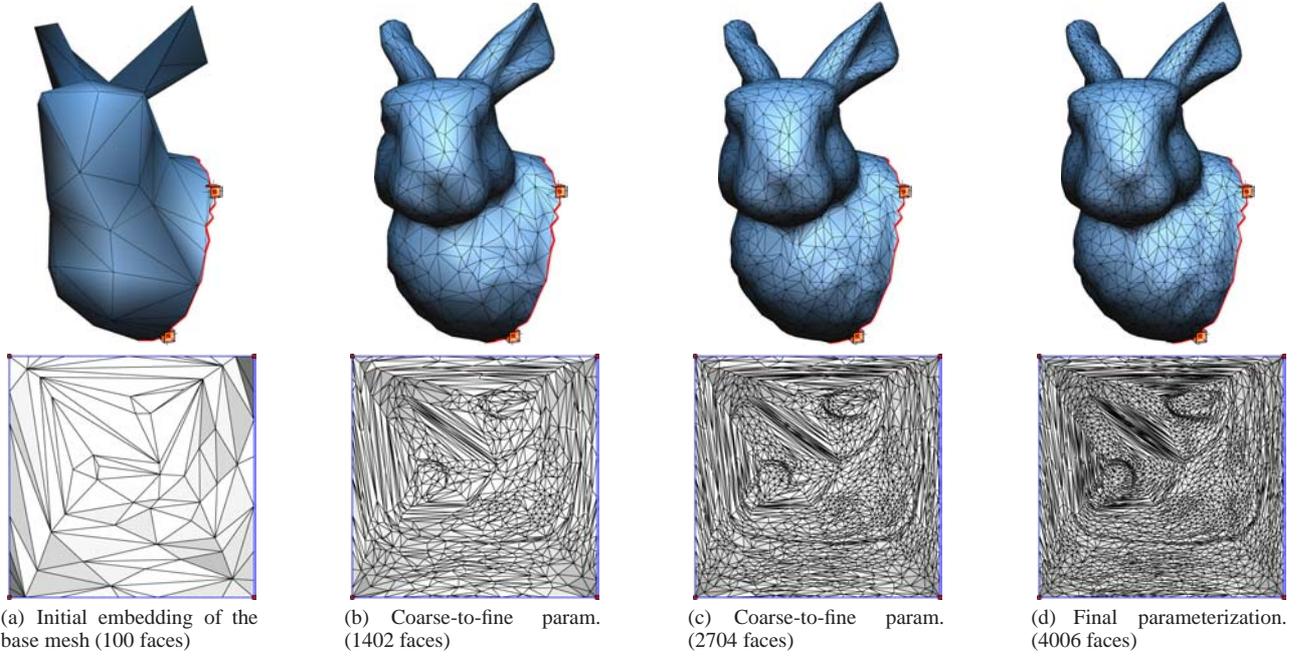


Figure 3: Several coarse hierarchy levels of a triangulated data set (top) and the corresponding optimized parameterizations (bottom).

search used in [37]) along randomly search directions in the (s,t) parametric domain.

3.3.3 Feature point alignment

To achieve the consistent mesh parameterization among multiple patches, parameter coordinates of corresponding feature points are aligned to the same position on the initial embedding by using a foldover-free warping algorithm, which will be detailed in Section 3.4.

3.3.4 Coarse-to-fine parameterization

Having derived the initial embedding with feature correspondences aligned, we now describe the coarse-to-fine parameterization algorithm which embeds all other vertices and, in the meantime, optimizes geometric-stretch metric.

Vertex insertion and optimization. For each vertex-split-refinement operation in the PM sequence, we place the new vertex at the centroid of the kernel of its neighborhood polygons to maintain an embedding (i.e. avoid flipped or degenerate triangles). Note that if the mapping is an embedding prior to the vertex insertion, the kernel can not be empty. After inserting a new vertex, we exploit the local vertex optimization to optimize the stretch of the vertex and its neighborhood one at a time. The vertex optimization always decreases the geometric-stretch metric, and since it is positive, it must converge to a local minimum. Note that feature points are retained in the optimization process to preserve the feature alignment. Figure 3 shows some hierarchy levels of the coarse-to-fine parameterization and their corresponding optimized parameterizations.

3.4 Feature point alignment

To achieve the consistent mesh parameterizations, a *foldover-free warping* in the parametric domain is applied to the initial embedding such that corresponding feature points are aligned to the same position. An approach similar to image foldover-free warping proposed [12] is used.

3.4.1 Feature adjustment

Given patches S^i , $i = 1, \dots, m$, and each associated set of feature points $\{f_1^i, \dots, f_e^i\}$, where e is the number of feature points on S^i . We compute the averaged feature position \bar{f}_k for each set of corresponding feature points $f_k^1, f_k^2, \dots, f_k^m$, $k = 1, \dots, e$, as the destination of the alignment for each f_k^i , $i = 1, \dots, m$; i.e.,

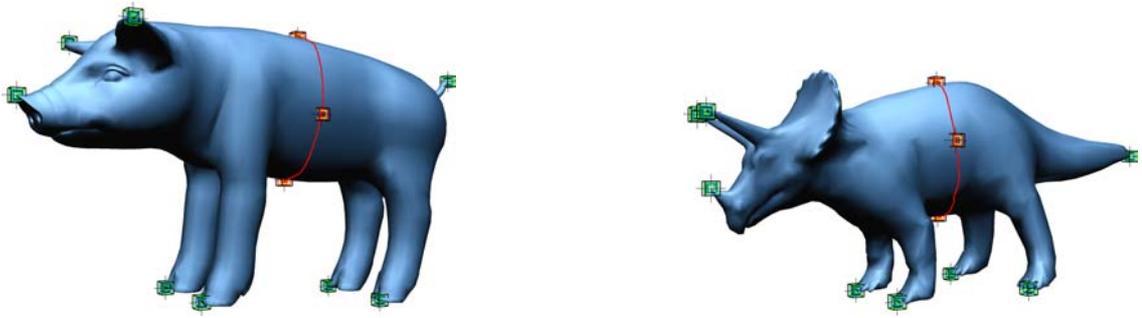
$$\bar{f}_k = \frac{1}{m} \sum_{i=1}^m f_k^i, \quad k = 1, \dots, e.$$

3.4.2 Parameterization warping

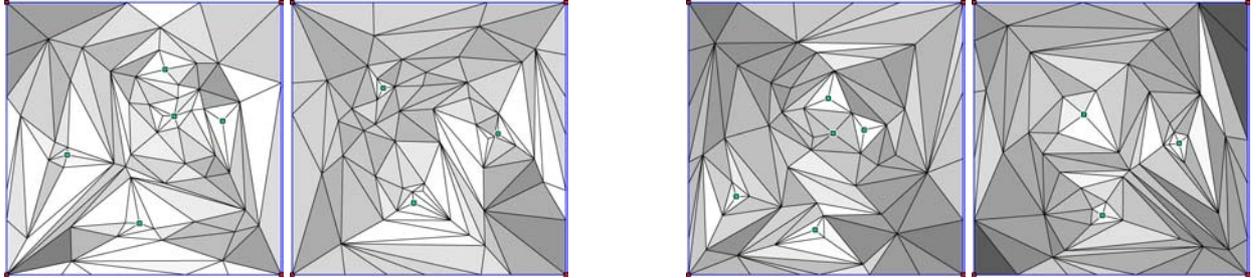
Patch triangulation For each S^i , we first construct a *warp mesh* by a 2D Delaunay triangulation which takes four corners (four seed points), and f_1^i, \dots, f_e^i as input. All other points will be marshaled into their respected enclosing triangles and their Barycentric coordinates are also computed. The objective of the parameterization warping for S^i is to move feature point f_k^i to \bar{f}_k for all i and k and recompute parametric coordinates for all other points such that the mapping is still bijective.

Triangulation over time During warping the warp mesh, the movement of the feature points may results in triangle degeneration and, in the consequence, folding over. We call such case an *event*. Before moving the feature point f_k^i , we detect if there will be an event by a binary search between its current position and the destination. If an intermediate position of the event is found, we first alter the local triangulation and recompute all Barycentric coordinates affected by this alteration, and then move f_k^i to the position of the event. Then all parameters are recomputed by using the new barycentric coordinates. If no event is found, simply move the feature point to the destination and recompute all parameters. Note that the influence range of the warping is not global and only parameters marshaled inside this range are affected.

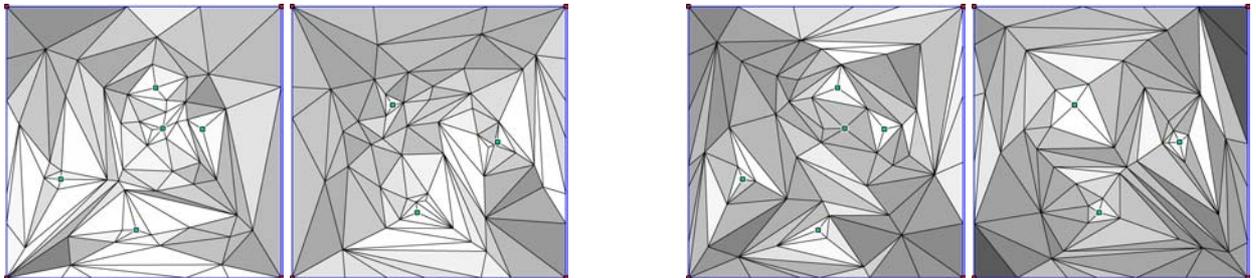
The triangulation over time process is applied to all feature point, one at a time. The processing order shall affect the final distribution of the non-feature points, the mapping is, however, bijective.



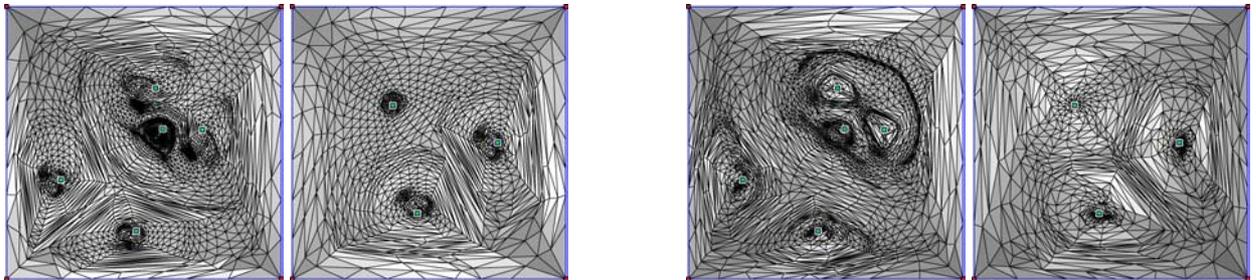
(a) Base domain and feature points



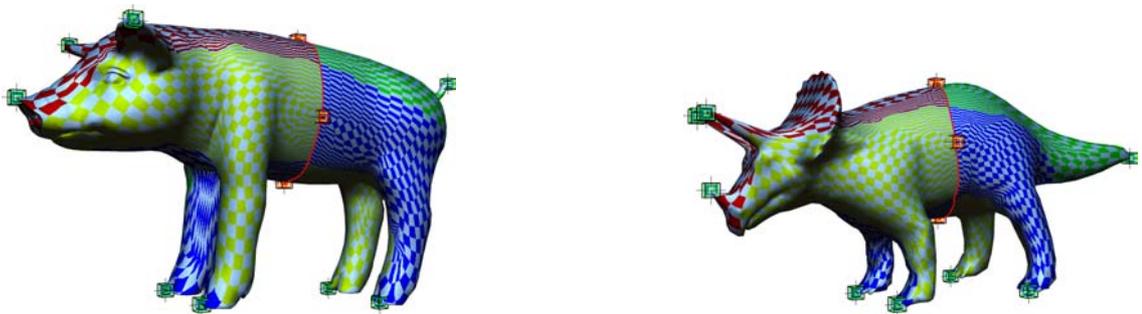
(b) Initial embedding



(c) After feature alignment



(d) After coarse-to-fine parameterization



(e) With checked texture mapping

Figure 4: The steps of consistent mesh parameterizations. (The green marks are “feature points”)

3.5 Examples

Figure 4 shows the steps of our consistent mesh parameterization. The checkerboard texture is used to indicate the quality of the parameterization. Note that four different colors represent the parts in correspondence between two models.

4 Remeshing

With derived consistent parameterization for each corresponding patches, the original models can be approximated by either uniform remeshing or adaptive remeshing. Remeshing is a process that samples in the parameterized domain and derives inverse mapped 3D vertices. The face containing the inverse mapped vertex is first derived by a point location algorithm. The inverse mapped vertex is then computed using the barycentric coordinates.

Uniform remeshing does the sampling based on regular subdivision on the square domain, and in consequence, the total number of samples is $(2^n + 1)^2$ for level n . For multiple models, the remeshed models of the same level will automatically have the identical connectivity. Note that, with the parameterization, all required geometry information can be stored in a 2D array or a quad-tree data structure depending on the applications. Figure 5 shows the result of uniform remeshing derived from the parametrizations in Figure 4.

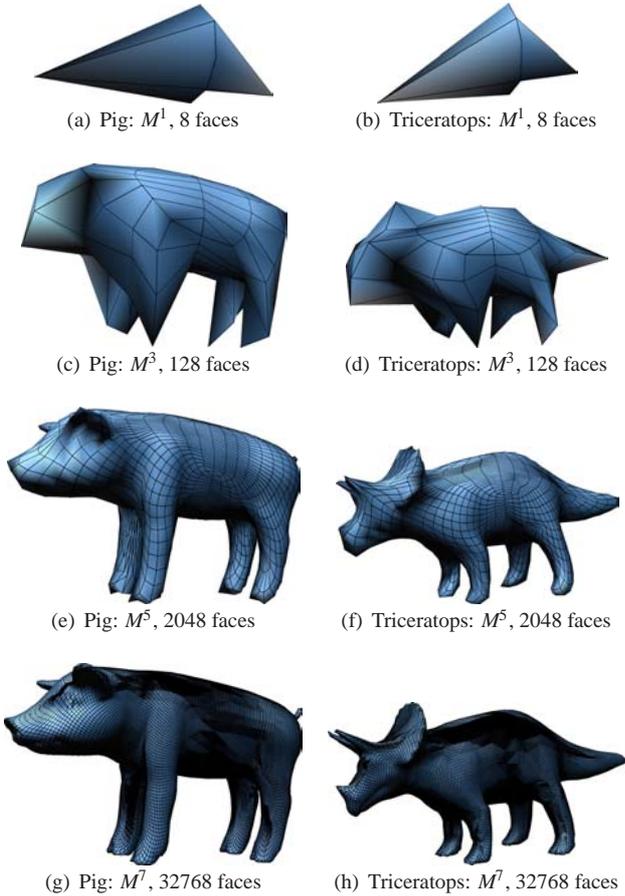


Figure 5: The uniform remeshing derived from the parameterizations in Figure 4.

Uniform remeshing has the drawback that in order to resolve a small local feature on the original mesh, one may need to subdivide to a very fine level. This total number of faces will be quadrupled.

Adaptive remeshing will have more samples in the area of high curvature.

For a given input mesh \mathcal{M} , a base domain consisting of some quad-faces corresponding to patches are constructed. For a given quad-face q , we find a best-fitting plane g , and measure the minimum Euclid distance between the plane g and each vertex in the patch P_q associated with quad-face q . Let $|v - p|$ be the minimum Euclid distance for each $v \in q$ and each $p \in g$. We define a error function $E(q)$ for each quad-face q of the remesh as the maximum of such distances, i.e.,

$$E(q) = \max_{v \in P_q} d(v) \quad (4)$$

The error function can be normalized by the diagonal length of the bounding box of the input mesh, denoted as $B(\mathcal{M})$; i.e.,

$$E_N(q) = \frac{E(q)}{B(\mathcal{M})} \quad (5)$$

We begin the remeshing process with base domain mesh and construct a quadtree root for each quad-face of base domain mesh. Then we evaluate the error of quad-faces in each quadtree based on the error function $E_N(q)$. If the error of a quad-face q , $E_N(q)$, is exceeding a pre-defined error bound ϵ , the quad-face is further refined and its geometry information are resampled. In other words, we take the quad-face to next finer level and four new children will be attached into the quadtree. The process is performed recursively and the adaptive remesh is constructed. However, there will be lots of *T-vertices*, which appear along the boundaries between quad-faces of different levels. We first force the level difference between neighboring quad-faces to be at most one by deliberately refining the quad-face of coarser level. Then perform adaptive subdivision to quad-face of coarser level. To perform adaptive remeshing consistently on multiple models, we simply take the maximum of the error of each corresponding quad-faces in the multiple models as follows:

$$E_C = \max_{1 \leq i \leq n} (E_N(q_i)) \quad (6)$$

where n is the number of models. This scheme of adaptive remeshes will result in meshes with the same connectivity. Figure 6 shows the results of the adaptive remeshing.

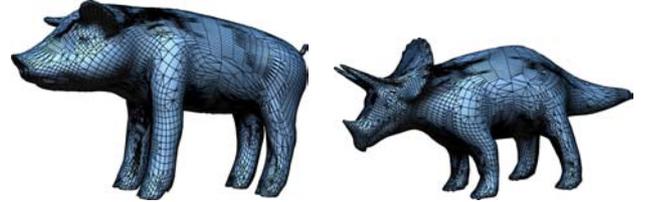


Figure 6: Adaptive remeshing of pig and triceratops models, 17998 faces ($\epsilon = 0.002$).

Although adaptive remeshing has done a great job for reducing the total number of faces under the geometry criterion, it still is an approximation of the original mesh. Much small detailed geometric features might still not well reconstructed. We can employ *normal mapping* technique for further compensation on this issue. With the modern graphics hardware, which support “*multi-texturing*” and “*pixel shader*” (or “*register combiners*” in OpenGL), the per-pixel lighting can be performed in real-time [26]. Figure 7 shows the result with and without normal mapping.

5 Results and Analysis

We have implemented consistent mesh parametrization and remeshing as described above. The applications was written in C++ using

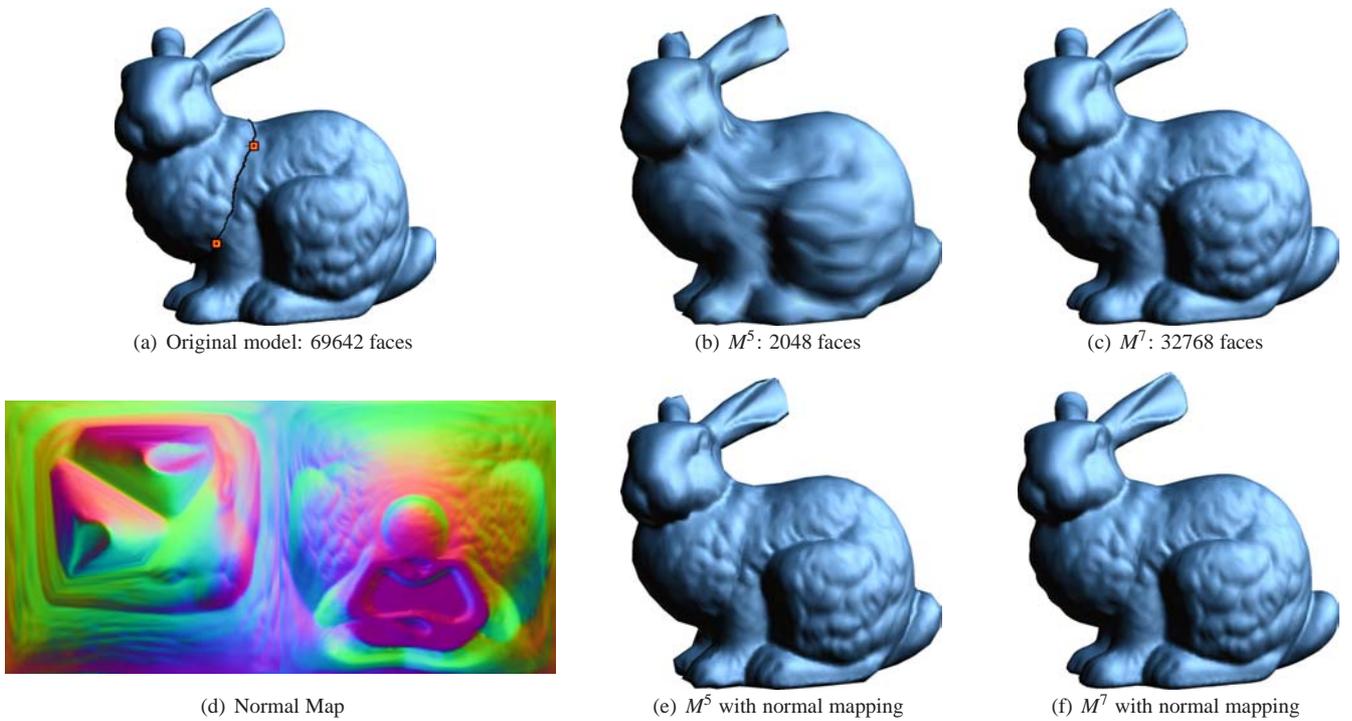


Figure 7: Normal mapping example.

standard computational geometry data structures, for example, half-edge data structures. The operating system is Microsoft Windows 2000. The graphics rendering is based on OpenGL with nVidia extensions.

The remeshing results are evaluated by IRI-CNR Metro tool [6]. The Metro tool compares two triangular meshes which describe the same surface. The tool accepts input meshes in the Open Inventor, PLY and SMF format. Error can be measured in terms of the symmetric mean distance, the symmetric RMS distance and the volume difference for closed surfaces. After got the symmetric RMS distance, we follow the evaluate function in [15]. Accuracy is measured as Peak Signal to Noise Ratio $PSNR = 20\log_{10}(peak/d)$, where *peak* is the diagonal length of the bounding box and *d* is the symmetric RMS Hausdorff error (geometric distance) between the original mesh and the remesh. Roughly speaking, the PSNR about 70dB is considered to be a nice approximation.

Table 1 shows calculation time, which is performed on a 1.5GHz AMD Athlon XP PC. Table 2 shows the result of uniform and adaptive remeshing. Both single object adaptive remeshing and multiple object adaptive remeshing are shown. Table 3 shows the PSNR and Table 4 shows the symmetric Hausdorff distance, i.e. maximum error.

6 Mesh Morphing

6.1 Morphing in spatial domain

The meshes constructed from the remeshing process for multiple models have the same connectivity. We can simply linearly interpolate their positions. A sequence of intermediate morphed models will be generated, and they still have multiresolution structures. Note that our method can handle multi-target morphing as well. Extra feature correspondences are also crucial to the quality of the morphing.

The morphing speed is fast at coarser level, but the visual quality is poor. As previously mentioned, by using consistently adaptive remeshing, we can construct adaptive remeshes for multiple models and morph them. By using normal mapping technique, a coarse

level remesh can still be morphed with high visual quality. It's a pity that normal map interpolation can not be easily formulated for hardware acceleration such as register combiners or pixel shaders. Because linear interpolated normal vector will not have the unit length, the further normalization is needed. Figure 8 shows the spatial morphing sequence from a pig model to a triceratops model.

6.2 Scheduled interpolation in wavelet domain

The interpolation path is also a major issue in the context of morphing. Besides linear interpolation, spline interpolation can also be employed easily. We also can perform the interpolation in another domain instead of spatial domain. Hughes [22] apply Fourier transform to volume data and perform interpolation for volume morphing in frequency domain while He et al. [17] performed similar process in wavelet domain. This idea is impossible for irregular meshes, but is now possible for regular and semi-regular meshes. We use the bi-orthogonal lifting wavelet proposed by Bertram et al. [4]. Their wavelet is much similar to Catmull-Clark [5] subdivision wavelet, and has small support. Therefore, both forward and inverse wavelet transform can be computed quickly. The generated wavelet coefficients can be manipulated in such a way that the inverse wavelet transform will reconstruct a morphed model. We apply different schedules to different resolutions in wavelet domain, i.e. assign different starting interpolation time and speed for different resolution in wavelet domain. We design the schedule as shown in Figure 9. The high frequency part of source object is disappearing quickly, and the low frequency part is interpolated gradually. The high frequency part of target object is then add back quickly to the end of morphing.

6.3 Multi-target morphing

As shown in Figure 10, with the consistent parameterizations and remeshing, we can produce any morphing sequence among these models.



Figure 8: Spatial morphing examples.

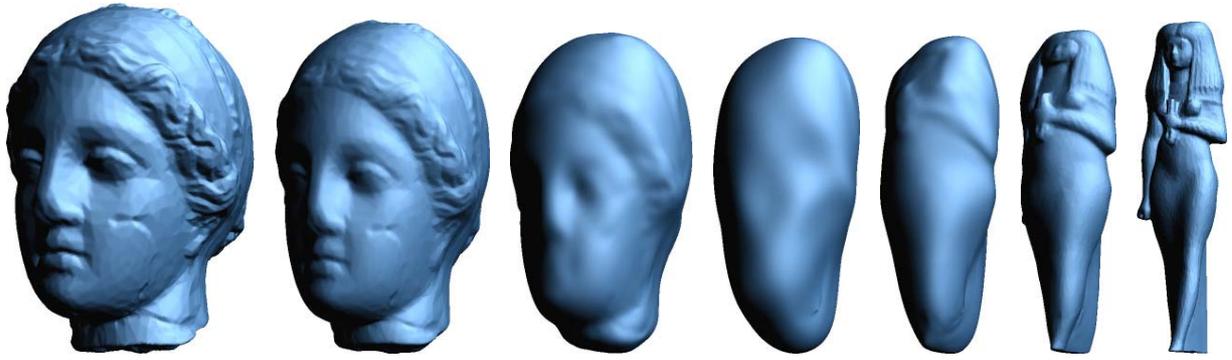


Figure 9: Morphing of scheduled interpolation in wavelet domain.

7 Conclusion and Future Work

We have proposed a novel method to compute consistent parameterizations for multiple models with a few seed points specified by users. Extra feature points in correspondences can also be specified for semantics and aligned during the parameterization procedure. Based on the result of parameterizations, uniform and adaptive remeshing can be performed to yield a set of semi-regular meshes. Moreover, geometric details can easily be resampled and stored as normal maps to improve the visual effects using the modern graphics hardware. In addition, the geometry images generated using the same parameterizations have a simple grid structure that benefits a number of applications, including normal mapping, texture mapping, rendering, level-of-detail, compression, and morphing. We have demonstrated the 3D mesh morphing application by the resulting remeshing. Besides interpolation in spatial domain, scheduled interpolation in wavelet domain is also demonstrated.

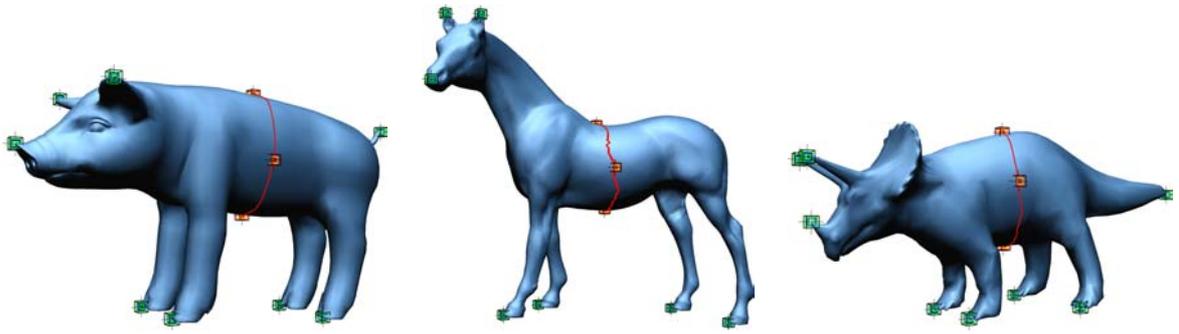
There are a number of areas for future work:

- **Higher genus models:** We seek more convenient methods to handle the layout of the domain for higher genus models.
- **Improvement on parameterizations:** Models with some protrusions are difficult to get good parameterizations. Besides, anisotropy parameterization along these protrusions also becomes unavoidable.
- **Feature-sensitive remeshing:** Remeshing schemes used here are not effective in capturing sharp features. Other resampling schemes may improve the geometric quality to represent sharp features accurately [2, 3, 19, 41].
- **Robust to multiple models:** An improvement or a controllable scheme on feature alignment is significant for the correspondence establishment among models.

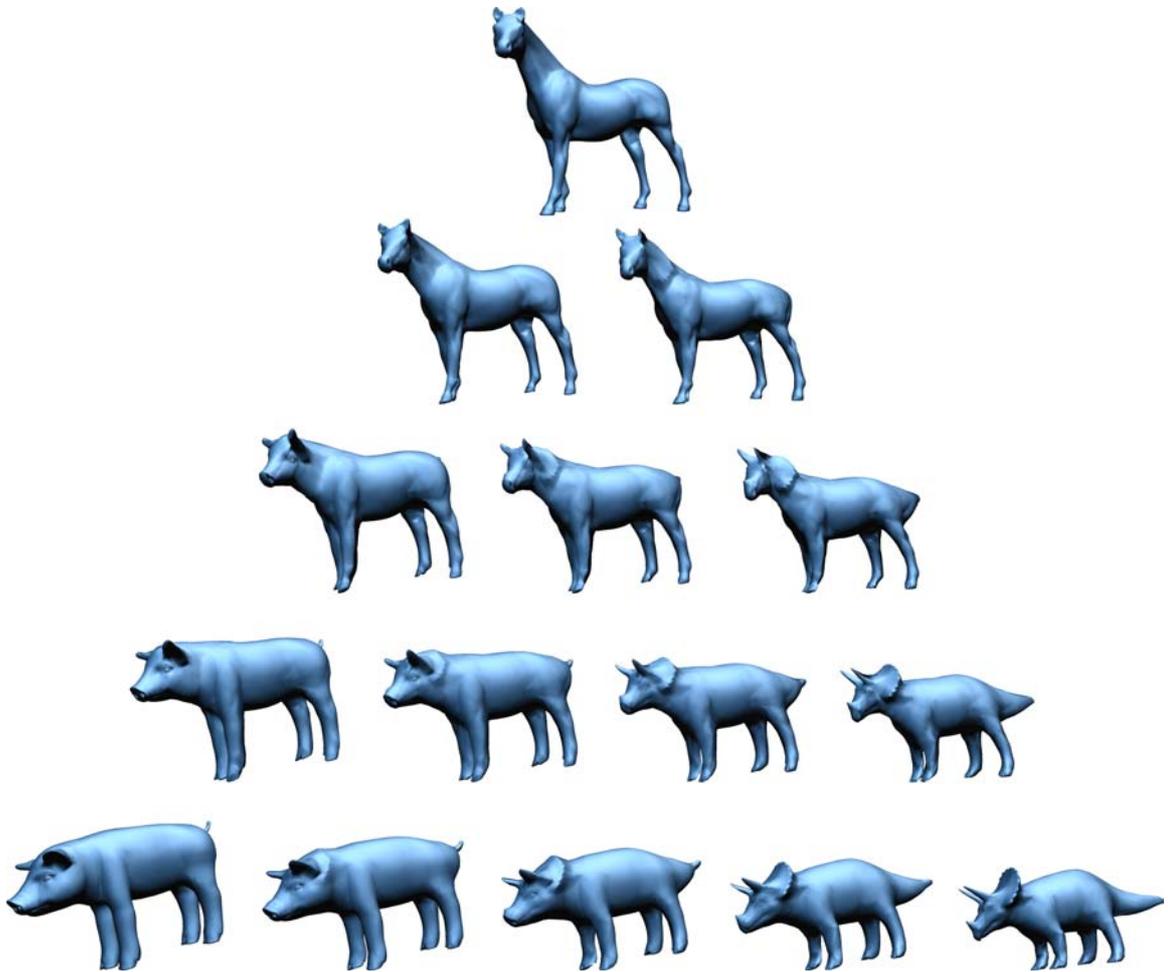
References

- [1] M. Alexa. Mesh morphing. In *EUROGRAPHICS'01 State of The Art Report*, 2001.
- [2] P. Alliez, É. C. de Verdière, O. Devillers, and M. Isenburg. Isotropic surface remeshing. In *Proceedings of Shape Modeling International*, 2003.
- [3] P. Alliez, M. Meyer, and M. Desbrun. Interactive geometry remeshing. *ACM Transactions on Graphics. Special issue for SIGGRAPH conference*, 21, 3:347–354, 2002.
- [4] M. Bertram, M. A. Duchaineau, B. Hamann, and K. I. Joy. Bicubic subdivision-surface wavelets for large-scale isosurface representation and visualization. In *IEEE Visualization '00 (VIS '00)*, pages 389–396, Oct. 2000.
- [5] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10:350–355, Sept. 1978.
- [6] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum*, 17(2):167–174, 1998. URL <http://vcg.iei.pi.cnr.it/metro.html>.
- [7] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, Mass., 1990.
- [8] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *SIGGRAPH 95 Conference Proceedings*, pages 173–182, Aug. 1995.
- [9] I. Eckstein, V. Surazhsky, and C. Gotsman. Texture mapping with hard constraints. *Computer Graphics Forum*, 20(3), 2001. ISSN 1067-7055.
- [10] M. S. Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231–250, 1997.
- [11] M. S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20:19–27, 2003.
- [12] K. Fujimura and M. Makarov. Foldover-free image warping. *Graphical models and image processing: GMIP*, 60(2):100–111, Mar. 1998.
- [13] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. pages 209–216, Aug. 1997.
- [14] A. D. Gregory, A. State, M. C. Lin, D. Manocha, and M. A. Livingston. Feature-based surface decomposition for correspondence and morphing between polyhedra. In *Computer Animation'98*, June 1998.
- [15] X. Gu, S. J. Gortler, and H. Hoppe. Geometry images. In *SIGGRAPH 2002 Conference Proceedings*, pages 335–361, 2002.
- [16] I. Guskov, K. Vidimče, W. Sweldens, and P. Schröder. Normal meshes. In *Proceedings of the Computer Graphics Conference 2000 (SIGGRAPH-00)*, pages 95–102, July 23–28 2000.
- [17] T. He, S. Wang, and A. Kaufman. Wavelet-based volume morphing. In *Proceedings of the Conference on Visualization*, pages 85–92, Oct. 1994.
- [18] H. Hoppe. Progressive meshes. In *SIGGRAPH 96 Conference Proceedings*, pages 99–108, Aug. 1996.

- [19] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. 27:19–26, Aug. 1993.
- [20] K. Hormann and G. Greiner. MIPS: An efficient global parametrization method. In *Curve and Surface Design: Saint-Malo 1999*, pages 153–162. Vanderbilt University Press, 2000.
- [21] K. Hormann, G. Greiner, and S. Campagna. Hierarchical parametrization of triangulated surfaces. In *Proceedings of Vision, Modeling, and Visualization 1999*, pages 219–226, Nov. 1999.
- [22] J. F. Hughes. Scheduled Fourier volume morphing. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 43–46, July 1992.
- [23] T. Kanai and H. Suzuki. Approximate shortest path on polyhedral surface and its applications. *Computer-Aided Design*, 33(11):801–811, September 2001.
- [24] T. Kanai, H. Suzuki, and F. Kimura. Three-dimensional geometric metamorphosis based on harmonic maps. *The Visual Computer*, 14(4):166–176, 1998.
- [25] T. Kanai, H. Suzuki, and F. Kimura. Metamorphosis of arbitrary triangular meshes. *IEEE Computer Graphics and Applications*, 20(2):62–75, Mar./Apr. 2000.
- [26] M. J. Kilgard. A practical and robust bump-mapping technique for today's gpus. In *Game Developer's Conference 2000 Course "Advanced OpenGL Game Development"*, 2000.
- [27] F. Lazarus and A. Verroust. Three-dimensional metamorphosis: a survey. *The Visual Computer*, 14:373–389, 1998.
- [28] A. Lee, D. Dobkin, W. Sweldens, and P. Schröder. Multiresolution mesh morphing. In *Proceedings of the Conference on Computer Graphics (Siggraph99)*, pages 343–350, Aug. 8–13 1999.
- [29] A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. MAPS: Multiresolution adaptive parameterization of surfaces. In *SIGGRAPH 98 Conference Proceedings*, pages 95–104, July 1998.
- [30] B. Lévy. Constrained texture mapping for polygonal meshes. In *SIGGRAPH 2001 Conference Proceedings*, pages 417–424, 2001.
- [31] B. Lévy and J. Mallet. Non-distorted texture mapping for sheared triangulated meshes. In *SIGGRAPH 98 Conference Proceedings*, pages 343–352, July 1998.
- [32] M. Lounsbery. *Multiresolution Analysis for Surfaces of Arbitrary Topological Type*. PhD thesis, Department of Computer Science and Engineering, University of Washington, 1994.
- [33] M. Lounsbery, T. D. DeRose, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Transactions on Graphics*, 16(1):34–73, Jan. 1997.
- [34] J. Maillot, H. Yahia, and A. Verroust. Interactive texture mapping. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 27–34, Aug. 1993.
- [35] T. Michikawa, T. Kanai, M. Fujita, and H. Chiyokura. Multiresolution interpolation meshes. In *Proceedings of the ninth Pacific Conference on Computer Graphics and Applications (PACIFIC GRAPHICS-01)*, pages 60–69, Oct. 16–18 2001.
- [36] E. Praun, W. Sweldens, and P. Schröder. Consistent mesh parameterizations. In *SIGGRAPH 2001 Conference Proceedings*, pages 179–184, 2001.
- [37] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing (2nd ed.)*. Cambridge University Press, Cambridge, 1992. ISBN 0-521-43108-5.
- [38] P. V. Sander, S. J. Gortler, J. Snyder, and H. Hoppe. Signal-specialized parametrization. In *Proceedings of the 13th Eurographics Workshop on Rendering (RENDERING TECHNIQUES-02)*, pages 87–98, June 26–28 2002.
- [39] P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe. Texture mapping progressive meshes. In *SIGGRAPH 2001 Conference Proceedings*, pages 409–416, 2001.
- [40] W. T. Tutte. How to draw a graph. *Proc. London Mathematical Society*, 13:743–768, 1963.
- [41] J. Vorsatz, C. Rössl, L. P. Kobbelt, and H.-P. Seidel. Feature sensitive remeshing. *Computer Graphics Forum*, 20(3), 2001.
- [42] M. Zöckler, D. Stalling, and H.-C. Hege. Fast and intuitive generation of geometric shape transitions. *The Visual Computer*, 16(5):241–253, 2000.



(a) The target models with extra feature correspondences.



(b) All morphing sequence.

Figure 10: Another example of multi-target morphing.

Model	face	time(sec)	
		parametrization	remeshing
venus+isis	10000+10000	35.69	2.38
horse+female	15000+10000	51.21	2.63
horse+triceratops	15000+5660	43.61	2.47
horse+pig	15000+7164	45.63	2.51
pig+triceratops	7164+5660	30.03	2.41
bunny	69642	100.14	2.15

Table 1: Statistics for computational time.
The parametrization applies coase-to-fine parametrization.
The remeshing scheme is uniform and the maximum remeshing level is 8.

Model	face										
	original	uniform remeshing								adaptive	
		M^1	M^2	M^3	M^4	M^5	M^6	M^7	M^8	SA	CA
venus	10000	8	32	128	512	2048	8192	32768	131072	9647	13570
isis										7470	
pig	7164									14693	17998
triceratops	5660									10475	
bunny	69642									14006	

Table 2: Statistics for various faces.
SA means "separately adaptive remshing with $\varepsilon = 0.002$;
CA means "consistently adaptive remeshing with $\varepsilon = 0.002$.

Model	PSNR (dB)									
	uniform remeshing								adaptive	
	M^1	M^2	M^3	M^4	M^5	M^6	M^7	M^8	SA	CA
venus	23.71	31.33	39.91	48.69	56.40	64.41	73.53	82.77	65.76	67.03
isis	27.12	30.15	37.48	47.18	55.83	65.06	74.46	84.00	66.23	68.00
pig	24.45	29.74	33.63	40.78	47.14	53.21	59.52	64.27	53.36	53.39
triceratops	24.72	29.86	34.35	42.93	48.89	54.20	61.07	68.69	55.56	55.62
bunny	23.34	29.24	36.87	44.24	52.31	60.79	70.96	80.33	65.82	

Table 3: Statistics for PSNR.

Model	symmetric Hausdorff distance(%)									
	uniform remeshing								adaptive	
	M^1	M^2	M^3	M^4	M^5	M^6	M^7	M^8	SA	CA
venus	9.99	6.58	4.46	2.50	1.05	0.63	0.42	0.16	0.27	0.27
isis	9.99	9.99	7.71	2.35	1.37	0.62	0.42	0.20	0.24	0.25
pig	9.99	9.99	9.34	4.05	2.89	1.51	1.09	0.80	0.93	0.91
triceratops	9.99	9.99	9.21	3.77	2.53	1.37	0.82	0.61	0.72	0.72
bunny	10.00	9.99	6.68	3.65	2.35	1.28	0.49	0.21	0.35	

Table 4: Statistics for symmetric Hausdorff distance with respect to the bounding box diagonal.