Consistent Mesh Parametrization Using Quinary Patch Subdivision

Jian-Liang Lin Jung-Hong Chuang Cheng-Chung Lin
Department of Computer Science and Information Engineering
National Chiao Tung University
1001 Ta Hsueh Road, Hsinchu, Taiwan 30050, ROC.

Submitted to Workshop on Multimedia Technologies

Contact: Jung-Hong Chuang

phone: 886-3-5731829
fax: 886-3-5724176
email: jhchuang@csie.nctu.edu.tw

Abstract

The correspondence establishment among multiple objects is a versatile operation in computer graphics and geometry processing, which in general takes as input the specification of a common dissection together with a set of feature points. We propose a systematic method called *recursive quinary subdivision* to efficiently find a dissection for an object with little user input. The quinary subdivision is a process that recursively dissects an highly stretched patch into five new patches. The process can be extended to multiple objects, taking into account the alignment of extra feature points, to derive a common dissection. Based on the dissection, uniform or adaptive remeshig can be performed to yield a set of semi-regular meshes. Moveover, geometric details can be easily resampled and stored as normal maps. We also demonstrate the mesh morphing application between two or more objects based on the correspondence established by the common dissection and parameterization.

Keywords: Mesh dissection, Parametrization, Remeshing, Morphing, Multiresolution modeling.

1 Introduction

Recently, *semi-regular* meshes are getting more and more popular as representations of complex objects in computer graphics and geometric modeling. Such meshes are multiresolution representations formed by starting from a coarse irregular base domain and applying recursive regular refinement. Due to their regular structures, parameter and connectivity information can be predicted [12], and efficient tree or array based data structure can be used in such a way that only geometry information needs to be stored. Moreover, signal processing algorithms such as wavelet analysis can be employed [4, 18, 17].

Remeshing is a process that for a given input irregular mesh, resamples its geometry information and constructs a semi-regular mesh which approximates the original irregular one. 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 parametrized to compute a bijection between the 3D domain and the parameter domain. Obviously, the patch layout generated by initial dissection is a vital factor that dominates the quality of the remeshing.

A single mesh can be remeshed to yield a semi-regular one. Can we extend the idea to multiple objects? The answer is not true unless they have a common initial dissection in which each patch of one object corresponds exactly to one patch in all other objects. In consequence, they will possess a same base domain and their parametrizations will be consistent. The diffculty is that the common initial dissections are not easily found, because a good dissection of one object might be bad for another object. Previous works [20, 19, 11, 7, 22] leave this problem to user, requiring the user manually provides a common initial dissection and many corresponding feature points. Besides, many applications such as metamorphosis (or morphing) and DGP (Digital Geometry Processing) applications [20], which require the establishment of correspondences among multiple objects, benefit from consistent parametrizations.

In this paper, we propose a systematic method called *quinary subdivision* to find a common initial dissection for multiple objects of genus-zero with little user interventions. The quinary subdivision scheme is a process that recursively subdivide an undesirable patch into five new patches, that

possess better parametrization than their parent patch. The quinary subdivision scheme can be extended to multiple objects and guarantees to yield a common initial dissection. Extra feature correspondences can also be provided by users. The alignment of feature correspondences during the quinary subdivision is also taken care of by using a foldover-free warping. After the common initial dissection is found, we begin the remeshing process and finally obtain a set of semi-regular meshes. The remeshing process can be either uniform or adaptive. Based on the parametrization, a set of *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 3D mesh morphing is demonstrated as an application of our proposed approach.

2 Related Work

Remeshing process begins with an input irregular connectivity model, dissect the model into a set of patches and then compute a parametrization for each patch. Eck et al. [4] proposed a method that produces a semi-regular mesh fully automatically by employing a Voronoi-like algorithm coupled with a Delaunay triangulation to yield the initial dissection, and parametrizing each patch using *harmonic mapping*. In contrast, the MAPS scheme proposed by Lee et al. [16] employed a mesh simplification algorithm to yield an initial dissection. Their algorithm is also automatic and more practical than Eck's. The *hierarchical conformal mapping* is performed during mesh simplification and a patch parametrization is obtained. Guskov et al. [8] proposed a new remeshing process to construct a compact representation called *Normal Mesh*. The above works focus on single objects. Praun et al. [20] illustrated that shortest paths probably lead to problems for finding an initial dissection, and proposed a modified shortest path algorithm to trace curves between given feature points and yeild a base domain. They also focus on multiple objects, but users are required to provide a patch layout for the common base domain and feature points identificaton on each objects.

The correspondence problem in the mesh morphing is naturally related to parametrizations. The survey of 3D morphing can be referred to [14], and mesh morphing, an extensive review 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. [10] used a single patch and the patch will be parametrized by harmonic mapping. In their recent works [11], the user first defines a set of corresponding features vertices and applied their approximate shortest path algorithm [9] to find an initial dissection. The works of Gregory et al. [7] and Zöckler et al. [22] also require users to manually provide initial dissections. Among the methods proposed, Lee et al. utilized their MAPS to the mesh morphing application [15]. But the base domains of the two input meshes are different—they are not consistently parametrized. They need a *"meta-mesh"* as an intermediate representation. Unfortunately this algorithm does not scale well, because the meta-mesh is more complicated than the original two models. Michikawa et al. [19] proposed a *multiresolution interpolation meshes(MIMesh)* 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 parametrized 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 Parametrization

3.1 Floater's patch parametrization

The patch parametrization builds a bijective mapping between a region $\mathcal{R}$ of the original input 3D mesh and a region $\mathcal{B}$ of the 2D parameter plane, i.e., find a bijective map $u : \mathcal{R} \rightarrow \mathcal{B}$. The map u is fixed by a boundary condition $\partial\mathcal{R} \rightarrow \partial\mathcal{B}$. For the parametrization proposed by Floater [5], we first fix the parameters of quadrilateral patch boundary vertices on a unit square, and then the function u needs to satisfy the following equation for each of the interior vertices v_i:

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

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

where $\mathcal{V}(i)$ is the 1-ring neighborhood of the vertex v_i, and n is the number of the vertices of the patch. To derive the mapping u, we first compute for each i the $\lambda_{i,k}$, $k \in \mathcal{V}(i)$, that satisfies shape-preserving criteria and $\sum_{k\in\mathcal{V}(i)} \lambda_{i,k} = 1$, then solve the linear system of Eq. 1 for $u(v_k)$. The weight finding procedure of $\lambda_{i,k}$, for each i, consists of two stages. The first one is to compute a local parametrization for v_i, followed by computing the $\lambda_{i,k}$ from the local parametrization.

The main advantage of Floater's parametrization 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 Eq. 1 can be solved robustly by the iterative biconjugate gradient method.

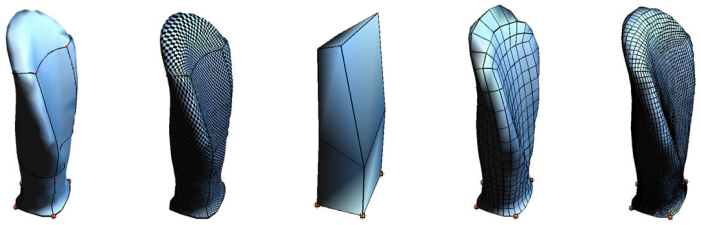
3.2 Quinary Patch Subdivision

Highly stretched parametrization is not desirable and will affect the subsequent remeshing process significantly. Fig. 1(a,b) shows a highly stretched parametrized patch with unbalanced sampling pattern. Fig. 1(c,d,e) shows the result after remeshing process is performed and the maximum remeshing levels are 0, 5, and 7. In this case, The original geometry can not be resampled well even in a finer level (M^7). Fig. 2 shows a better sampling pattern.

(a) (b) (c) (d) (e)

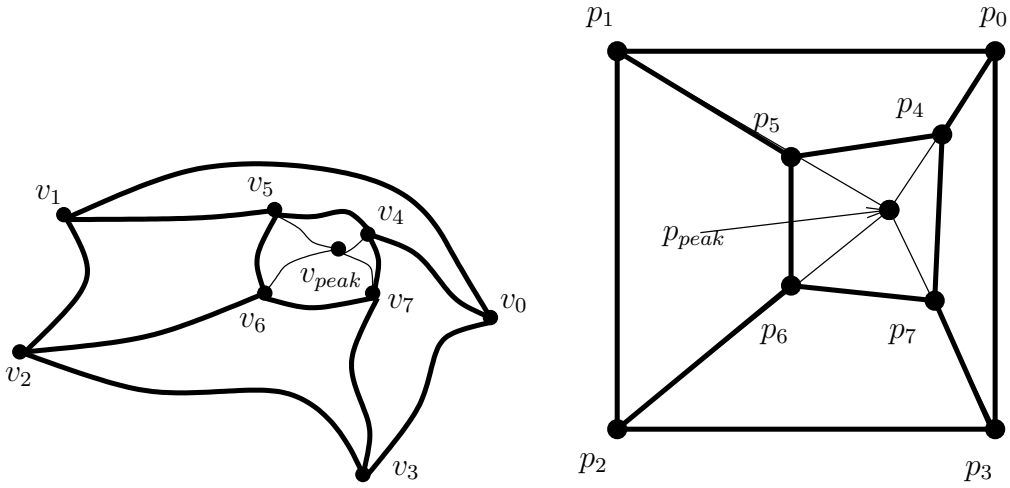
Figure 1: (a) Single patch. (b) Sampling pattern. (c) M^0. (d) M^5. (e) M^7.

We employ TMPM's L^2 stretch metric [21] to evaluate the patch's stretch ratio. If the stretch ratio of a patch P, say $L^2(P)$, exceeds a pre-defined threshold τ, it will be further subdivided into five patches by our quinary subdivision. Fig. 3 illustrates the quinary subdivision. A patch with four corners, v_0, v_1, v_2, and v_3, in R^3 is denoted as $P_{\{0,1,2,3\}}$. Likewise, a patch is parametrized on



(a) (b) (c) (d) (e)

Figure 2: (a) Subdivided to nine patches. (b) Sampling pattern. (c) M^0. (d) M^3. (e) M^5.



(a) 3D domain (b) 2D parameter domain

Figure 3: Quinary subdivision illustration

a unit square with four corners, p_0, p_1, p_2, and p_3, where $p_0 = u(v_0)$, $p_1 = u(v_1)$, $p_2 = u(v_2)$, and $p_3 = u(v_3)$, in R^2 is denoted as $S_{\{0,1,2,3\}}$. To apply our quinary subdivision to a patch $S_{\{0,1,2,3\}}$, we first find the greatest stretched face t_{peak} in R^2, which is T_{peak} in R^3, then simply take the centriod p_{peak} of t_{peak} as the peak point. For each corner p_i, $i = 0, 1, 2, 3$, of the patch, we find an intermediate point p_{i+4} on the line segment connecting p_{peak} and p_i as follows:

$$p_{i+4} = (1 - \omega_i)p_i + \omega_i p_{peak}, \quad i = \{0, 1, 2, 3\}, \quad 0 \leq \omega_i \leq 1 \tag{3}$$

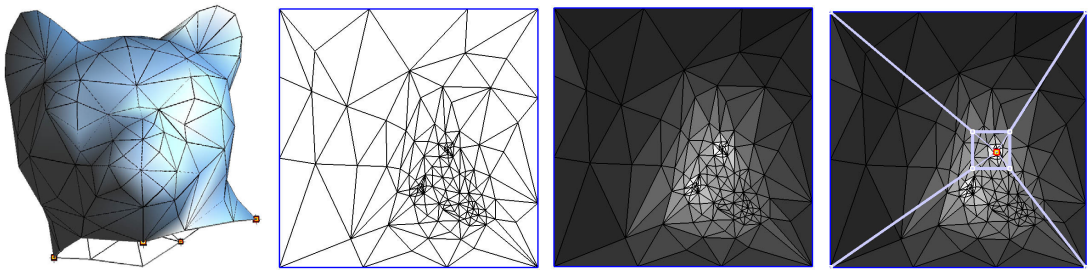
for some w_0, w_1, w_2, and w_3. Because of the exponential uptrend of the face stretch ratio, we simply choose a constant ω for ω_i, $i = 0, 1, 2, 3$, as

$$\omega = \psi + \log_{10}(L^2(t_{peak})), \tag{4}$$

where ψ is a bias and is taken as 0.85 in our current implementation. Now we have the four new corners, and the patch is subdivided into five new patches $S_{\{0,1,5,4\}}$, $S_{\{1,2,6,5\}}$, $S_{\{2,3,7,6\}}$, $S_{\{3,0,4,7\}}$, and $S_{\{4,5,6,7\}}$. The peak point finding and linear combination process ensure the subdivision be uniquely derived. As in [8], a straight line, which may go through faces, in the parameter domain will be a fair curve in 3D domain. By using the inverse mapping from 2D to 3D, the boundary curves of the newly created patches can be determined. We then employ a simple algorithm to marshal faces to their respected new patches. Pick a seed face by inverse mapping the 2D point, which is the average position of four corners of the new patches in the parameter domain, and recursively collect all its neighboring faces until the boundary curves are met. Finally the original patch can be removed. Fig. 4 shows the process of quinary subdivision on a cat head model.

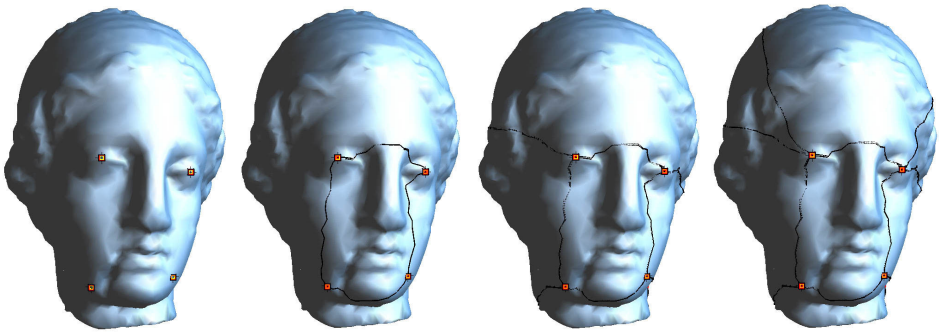
3.3 Dissection on a Single Object

It's obvious that the quinary subvision has recursive structure. If the newly created patches are still not desirable, they can be subdivided recursively. In order to perform quinary subdivision, we have to first divide a genus-zero model into two patches. This step requires users to specify four "seed points" on the surface. When four seed points are correctly given, we perform Dijkstra's shortest path algorithm [3] based on geodesic distance to trace from one seed point to another seed point



(a) (b) (c) (d)

Figure 4: (a) A cat head model. (b) The parametrization. (c) The face stretch ratio is visualized as face color. (d) The quinary subdivision.



(a) (b) (c) (d)

Figure 5: (a) Four seed points specified. (b) Two seed patches created. (c) Subdivision on the larger patch. (d) Result of recursive quinary subdivision.

to form a closed loop. Now with this closed curve, the original model is dissected into two "seed patches" and the respected faces are well marshaled; as shown in Fig. 5(a-b).

Now we have two patches in hand, and we put them into a list. For each patch in the list, we first check their stretch ratios. If there is a patch exceeding a user-specified threshold, the patch is removed from the list for further subdivision. The resulting newly created patches are then put into the list. The process is repeated until all patches in the list satisfy the stretch ratio test, and the dissection of a single object is found; as shown in Fig. 5(c-d).

We can also perform patch boundary relaxation and global vertex relaxation (similar to [8]) to yield a better dissection. Fig. 6 shows the relaxation of a boundary curve. Regarding this figure, we want to improve the boundary curve with endpoints v_2 and v_5. First, we find the two incident patches of the boundary curve and take all their corners except v_2 and v_5 to construct a temporary patch. Perform patch parametrization on the temporary patch and find the straight line from $u(v_2)$ to $u(v_5)$. The straight line in parameter domain will be a fair curve in 3D domain. The fair curve will be the new boundary curve. Likewise, we also can reposition the position of a corner for improve the distribution of the incident patches. Fig. 7 shows the similar process applied to a corner with five incident patches. Fig. 8(b) shows the relaxed dissection of Fig. 8(a). In practice, only initially

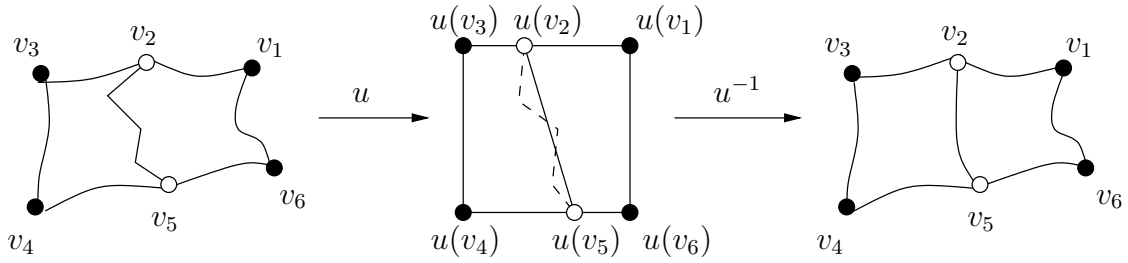


Figure 6: Relaxation of a boundary curve.

closed curves traced out by the shortest path algorithm necessarily require relaxation. All other curves resulting from the quinary subdivision are the inverse mapped of the line segments on the parameter domain, so they are often fair enough and require no further relaxation.

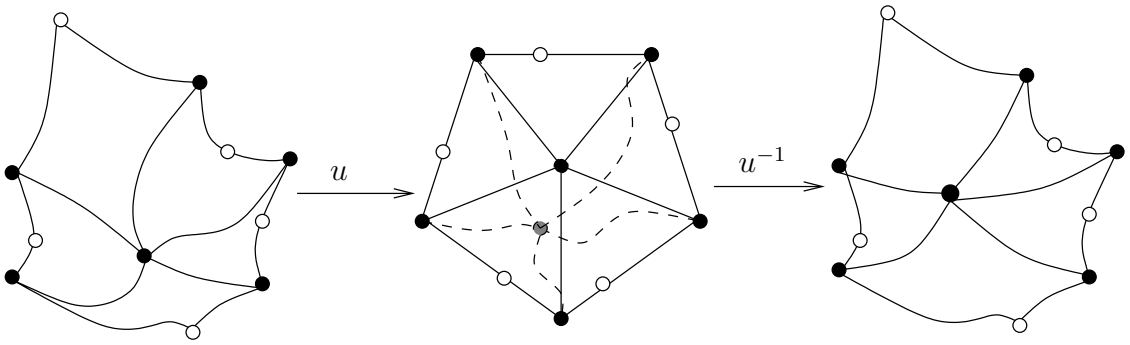
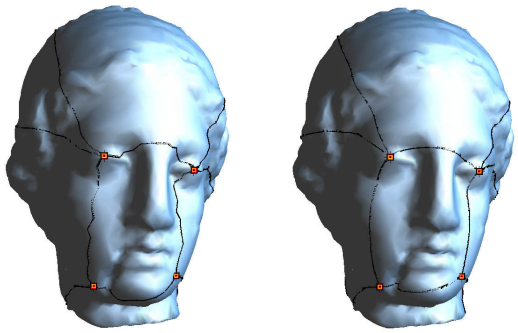


Figure 7: Relaxation of a corner vertex.



(a) (b)

Figure 8: (a) Result of recursive quinary subdivision. (b) The Relaxed version.

3.4 Common Dissection for Multiple Objects

The recursive quinary subdivision for the seed patches can be recorded as a "quinary tree". In order to yield a common dissection for multiple objects, the recursive subdivision processes must be the same. That is, their quinary trees must be the same. To this end, we can simply take the union of these quinary trees, denoted as Q_{union}. In this extent, the initial seed points act as feature corresponding points. Because of the initial seed points, there will be one unique union of the quinary trees. For an object and its respected quinary tree Q_i, we examine the difference between Q_i and Q_{union}, and further deliberately subdivide the patch if there is a newly added node or subtree. This process can be performed at end or iteratively in the course of quinary subdivision. For each iteration, quinaryly subdivide corresponding patches on other objects whenever a patch of an object needs to be subdivided. Repeat the process until all patches of all objects satisfy the stretch ratio test.

3.4.1 Extra Feature Correspondences

The derivation of base domain based on the quinary subdivision can be fully automatic with only four initial seed points specified by users. If the users require to specify extra feature corresponding points on the models for better performance on applications such as mesh morphing, the quinary subdivision rule can be enhanced to take into account the alignment of those additional feature points. Insufficient feature points may result in unexpected morphing sequence in such applications. Moreover, without proper alignment, the specified feature points in correpondence could belong to different dissected patches. Such cases often occur when feature points are specified too close or the input models are too dissimilar, Fig. 9 illustrates such situations. To prevent from this situation, a *foldover-free warping* in the parameter domain should be performed before each quinary subdivision.

We take the approach proposed in [6]. We briefly describe the main idea of the foldover-free warping algorithm. Given patches S^i, $i = 1, ..., n$, in R^2 and the associated set of feature points $\{f_1^i, ..., f_e^i\}$, where n is the number of input meshes and e is the number of feature points in S^i, we

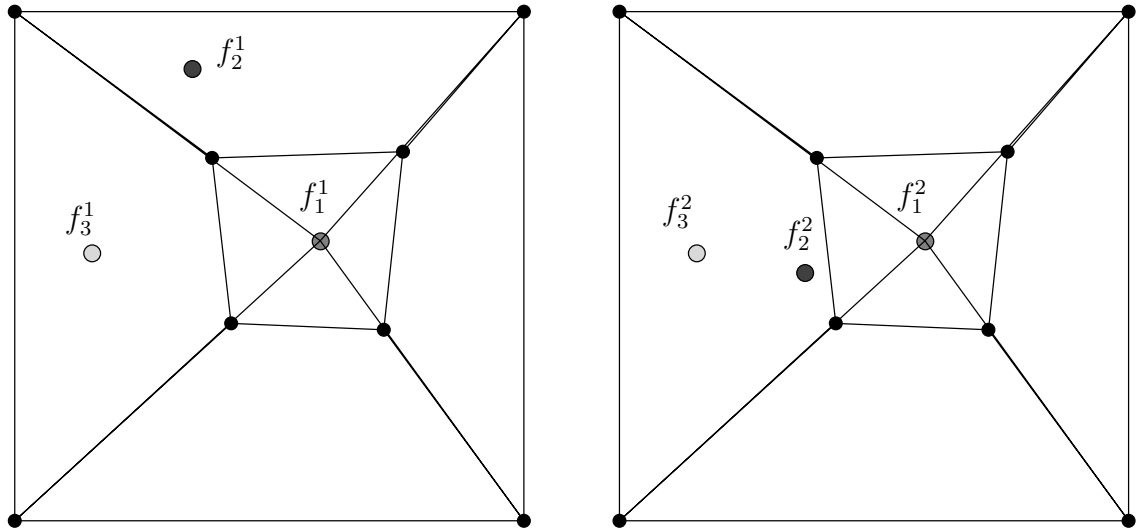


Figure 9: Feature points f_2^1 and f_2^2 are marshaled into different patches during quinary subdivision.

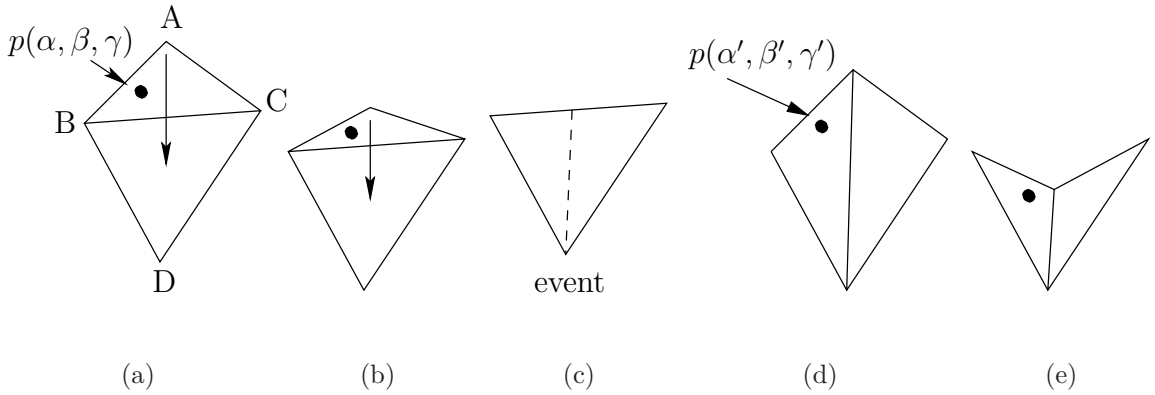
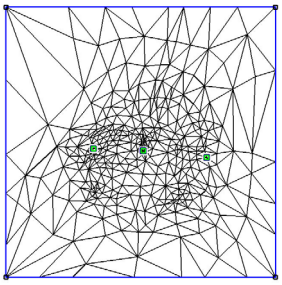


Figure 10: Triangulation over time. (a) Vertex A tends to move down. (b) Check if there is an event occuring during the movement. (c) An event, identifying the degeneration of triangle ABD, is detected. (d) The local triangulation is altered to prevent from the degeneration, and the parameters are recomputed by new barycentric coordinates. (e) Warped result.

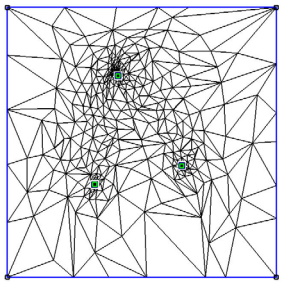
first compute the averaged feature point position as

$$\overline{f_k} = \frac{1}{n}\sum_{i=1}^{n} f_k^i, \qquad k = 1, ..., e.$$

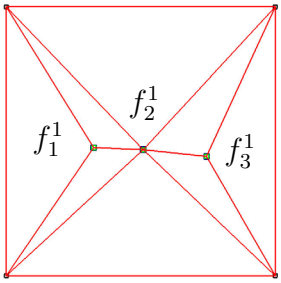
For each S^i, we first construct a *warp mesh* for it by a 2D Delaunay triangulation which takes four corners, and f_1^i,..., and f_e^i as input. All other points will be marshaled into their respected enclosing triangles and their barycentric coordinates are also computed. The objective of a warping in parameter domain of S^i is to move feature point f_k^i to $\overline{f_k}$ for all i and k and recompute all other parameters which will still keep it as a bijective mapping. Therefore, the algorithm is just performed to each individual patch, not to all patches simultaneously. During the deformation of the warp mesh by the movement of the feature points, some triangles of the warp mesh may degenerate and start folding over. We call such situation an *event*. In order to prevent triangles from folding over, an algorithm called *triangulation over time* will be performed. Before starting the deformation, we detect if there will be an event by employing a binary search between the source position and the destination position of an arbitrary feature point f_t. If an intermediate position of the event is found, we alter the local triangulation and recompute all barycentric coordinates affected by this alteration. Then we move f_t to the position of the event, which was detected but will not occur this time. Then all parameters are recomputed by using the new barycentric coordinates. The influence range of the warping is not global and only parameters marshaled in this range will be affected. Furthermore, if there are more than one feature point, the triangulation over time algorithm will process them one by one in an arbitrary order. Although the processing order will affect the final distribution of the non-feature points, what we wish is to keep the mapping bijective. After processing one feature point, we set the position as the source position and repeat the event detection for other feature points. The process terminates when all feature points reach their destination positions. If no event is found, simply deform the warp mesh to destination and recompute all parameters. Because the barycentric coordinates make the parameters inside a triangle bijective and the triangulation of the warp mesh is also bijective, we can assure that the warped parameters are still bijective. Fig. 10 illustrates the triangulation over time. Fig. 11 shows an example of foldover-free warping in the parameter domain. An example is shown in Fig. 12, in which, besides four limbs, more feature



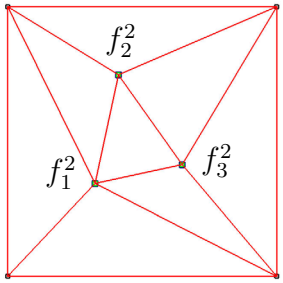
(a) Patch A.



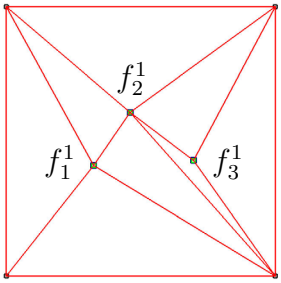
(b) Patch B.



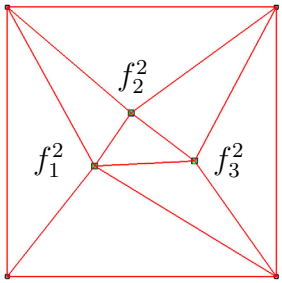
(c) Warp mesh of patch A.



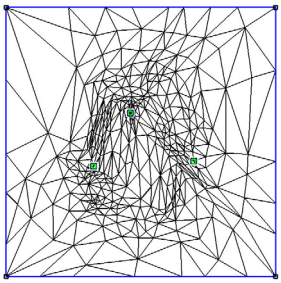
(d) Warp mesh of patch B.



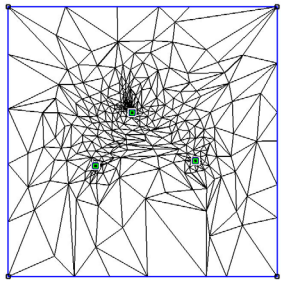
(e) Triangulation is altered during deformation.



(f) After deformation.



(g) Warped parameters.



(h) Warped parameters.

Figure 11: Foldover-free warping in the parameter domain.

points are specified for mouths, tail of the triceratops and hip of the horse, horn of the triceratops and ear of the horse. Fig. 13 shows the result of QS applied to the models of venus, venus head, and isis.

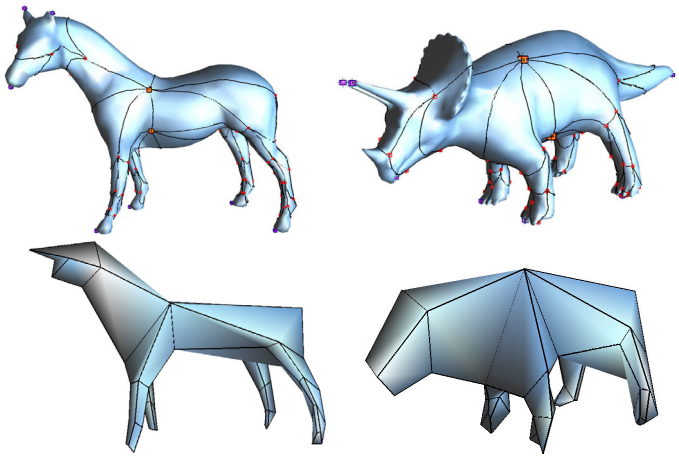


Figure 12: Common dissection and base domains of a horse model and a triceratops model.

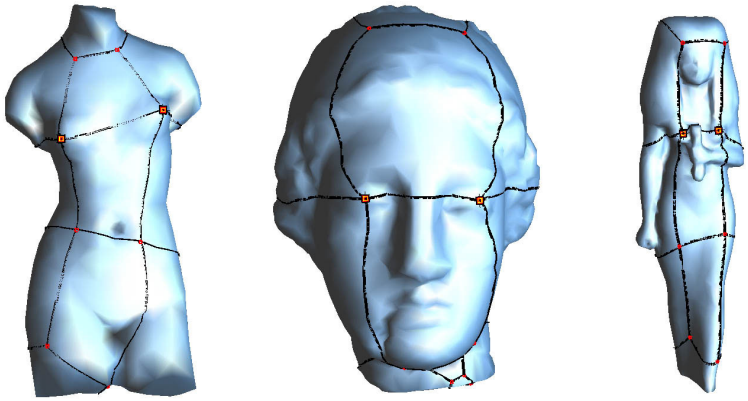


Figure 13: Common dissection of venus, venus head, and isis models.

4 Remeshing

4.1 Uniform Remeshing

With the parametrization ready for each patch of the base domain, we can start remeshing process. Each patch already has been parametrized onto a unit square. For each patch, take regular grid point

parameters and compute their inverse mapped 3D position. For an arbitrary parameter (vertices in parameter domain), we have to determine which face contains it. As previously mentioned, this is a point location problem. Standard computational geometry algorithms can be employed. In current implementation, we first perform spatial partitioning technique on 2D parameter domain and a 2D quadtree which all faces in the parameter domain will be marshaled into quadtree leaf node will be constructed. For a point location query, traverse the quadtree to find its respected leaf node. And each faces belonging to this leaf node will be tested. We restrict the face capacity of a leaf node, and the time complexity of a point-location query will be bounded to the depth of the quadtree.

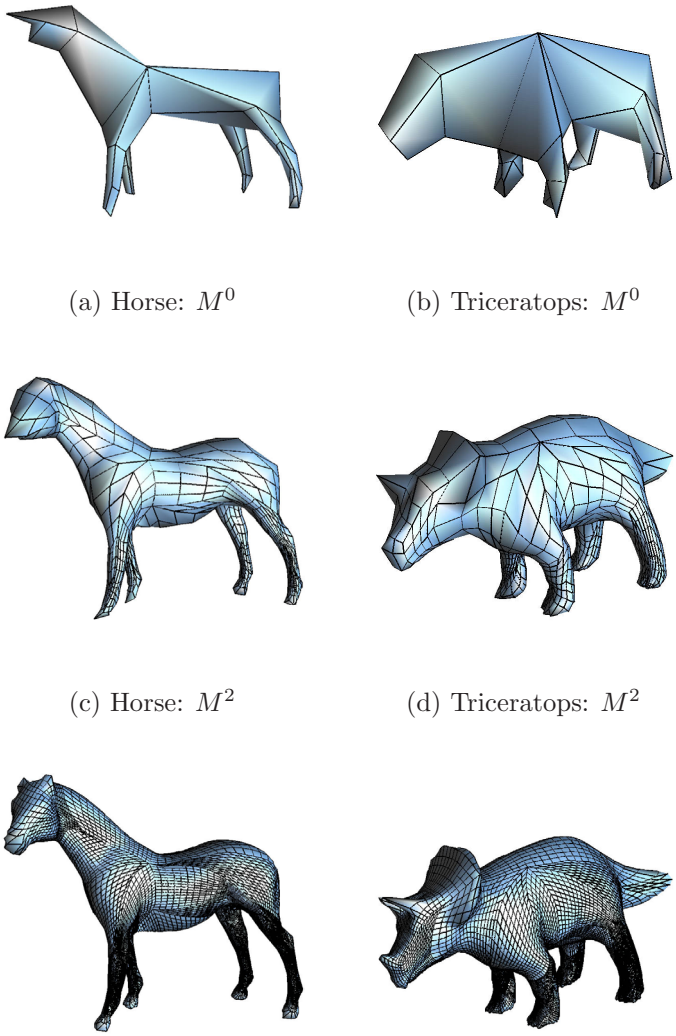
After the face is found, simply use the barycentric coordinates to compute its 3D position. If you want to resample to level n, the total amount of the samples is $(2^n + 1)^2 m$, where m is the number of patches. All required geometry informations can be resampled, and stored into a 2D array or a quadtree data structure depending on the applications. No matter 2D array or tree structure is used, only geometry informations need to be stored by arraging the data in a special ordering. The resulting file size will be small. As for multiple objects, the final remeshes will have the same connectivity structures. Fig. 14 shows the result of uniform remeshing derived from the dissection in Fig. 12.

4.2 Adaptive Remeshing

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. We now describe a simple method to build the adaptive remesh within a conservative error bound.

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 $d(v)$ be the minimum Eucilid distance for each $v \in P_q$ and each $p \in g$.

$$d(v) = \min_{p \in g} |v - p|$$



(a) Horse: M^0 (b) Triceratops: M^0

(c) Horse: M^2 (d) Triceratops: M^2

(e) Horse: M^4 (f) Triceratops: M^4

Figure 14: The uniform remeshing derived from the dissection in Fig. 12.

We define the error function $E(q)$ for each quad-face q as the maximum of $d(v)$ for all $v \in P_q$; i.e.,

$$E(q) = \max_{v \in P_q} d(v) \tag{5}$$

Fig. 15 illustrates the error function.

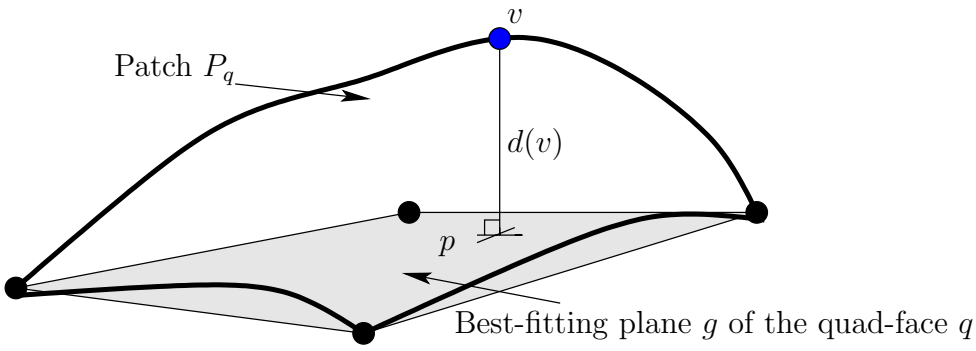


Figure 15: Error function definition.

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.,

$$E_N(q) = \frac{E(q)}{B(\mathcal{M})} \tag{6}$$

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 informations 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. Fig. 16 enumerates all cases for resolving T-vertex.

Now consider to perform adaptive remeshing consistently on multiple objects, we can simply take the maximum of the error of each corresponding quad-faces in the multiple objects. The result will

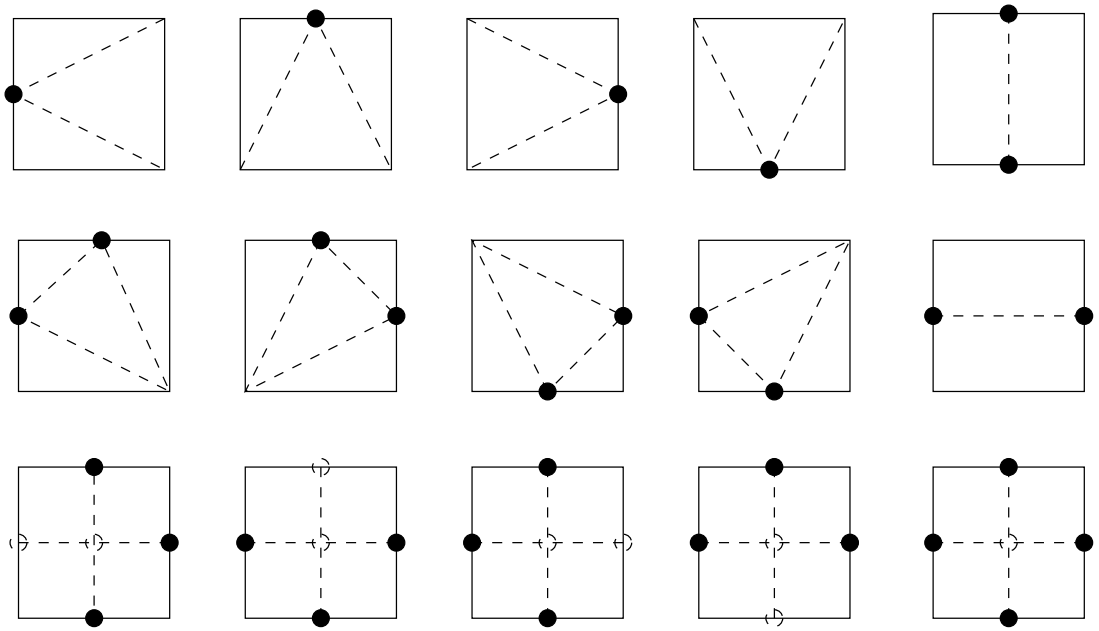


Figure 16: Fifteen cases to resolve T-vertex (Black points are T-vertices).

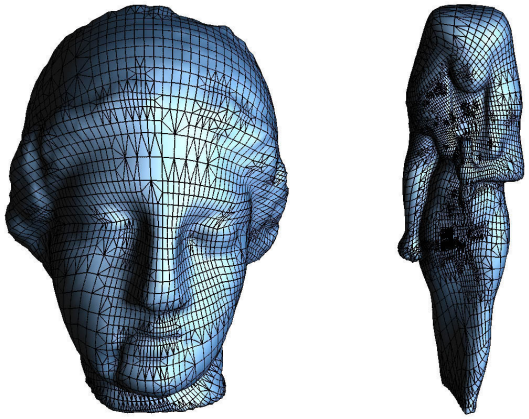


Figure 17: Adaptive remeshing of venus and isis models ($\epsilon = 0.0025$).

still be consistent.

$$E_C = \max_{1 \le i \le n} (E_N(q_i)) \tag{7}$$

where n is the number of objects. This will also result in adaptive remeshes with the same connectivity structure. If there are some local geometric features in one object, which correspond to a flat region in another object, the flat region will be forced to refined and still result in lots of faces. Fig. 17 shows the result of the consistently adaptive remeshing.

4.3 Normal Mapping

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.

Normal map is an image storing quantized normal vectors of surfaces. By mapping (n_x, n_y, n_z) in the range $[-1, 1]$ to (r, g, b) in the range $[0, 255]$, it's possible to recover the detailed geometry feature from this map and improve the rendering visual effects. This is appealing while a remesh of coarse level is rendered in real-time applications. Normal vectors are also geometry information and can be resampled as 3D position resampling. With barycentric coordinates, the intermediate interpolated normal vectors within a face can also be resampled. As shown in Fig. 18, because the patches are all square in parameter domain, the resampled normal maps do not require further packing algorithm such as pull-push algorithm in [21]. At run-time, we have to use normal map to calculate lighting equation in per-pixel fashion. 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 [13]. Fig. 19 shows the result with and without normal mapping.

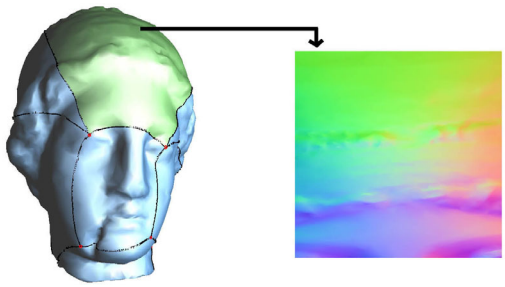
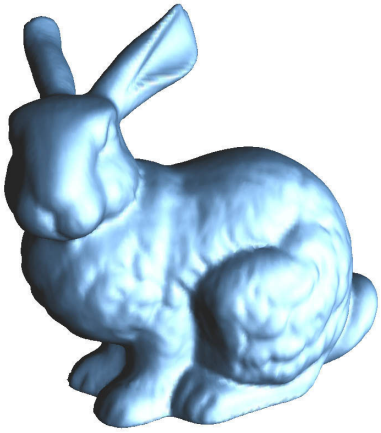


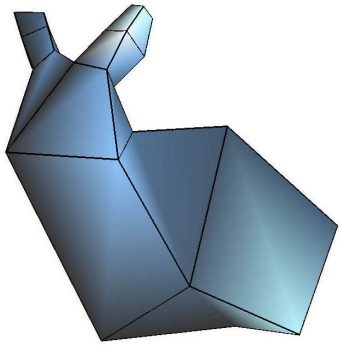
Figure 18: Normal map

4.4 Experimental Results

We have implemented quinary subdivision and remeshing as described above. The applications was written in C++ using standard computational geometry data structures, for example, half-edge

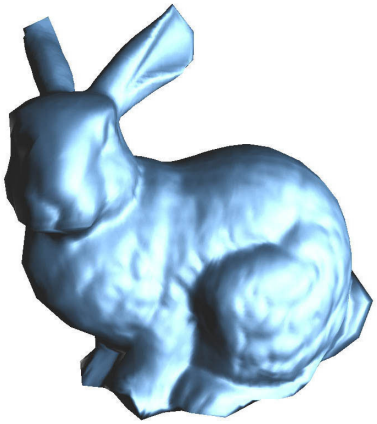


(a) Original model: 69664 faces

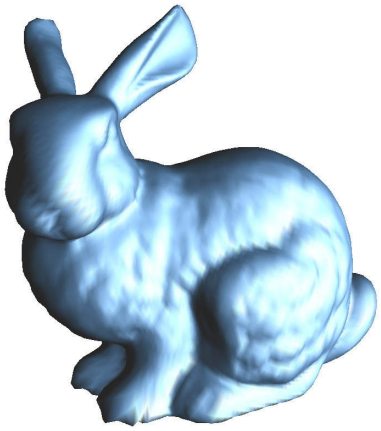


(b) Base domain: 34 faces

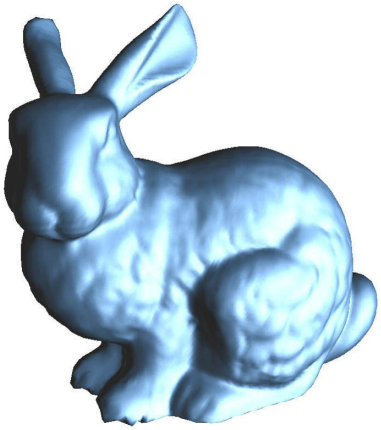
(c) M^3: 2176 faces



(d) M^3 with normal mapping



(e) M^5: 34816 faces



(f) M^5 with normal mapping

Figure 19: Normal mapping example.

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 [2]. We show the percentage of mean square error (L^2) normalized by the diagonal length of the bounding box of the models. Roughly speaking, the L^2 error below 0.05% is considered to be a nice approximation.

Table 1 shows the result of uniform and adaptive remeshing. Both single object adaptive remeshing and multiple object adaptive remeshing are shown. Table 2 shows the approximate errors. Table 3 shows calculation time, which is performed on a 750MHz Pentium III machine.

Model	size								
	original	uniform remeshing							adaptive remeshing ($\epsilon = 0.0025$)
		M^0	M^1	M^2	M^3	M^4	M^5	M^6	
venus	10000	14	56	224	896	3854	14336	57344	8656
isis									

Table 1: Statistics of polygon numbers.

Model	L^2 error(%)							
	uniform remeshing							adaptive remeshing ($\epsilon = 0.0025$)
	M^0	M^1	M^2	M^3	M^4	M^5	M^6	
venus	7.97	2.46	0.83	0.26	0.086	0.033	0.012	0.047
isis	2.64	1.32	0.66	0.19	0.073	0.026	0.009	0.037

Table 2: Statistics of errors.

5 Mesh Morphing

5.1 Morphing Among Two Models

The semi-regular meshes constructed from the remeshing process for multiple objects have the same connectivity. We can simply linearly interpolate their positions. A sequence of intermediate

Model	size	M^0	time(sec)	
			dissection	remeshing
venus+isis	10000+10000	14	11.967	1.522
horse+human	5000+5000	106	6.169	4.156
horse+triceratops	2000+5660	70	3.836	2.904
bunny	69664	34	143.967	10.145

Table 3: Statistics of calculation time. The remeshing is uniform and up to level 5.

morphed objects will be generated, and they still have multiresolution structures. Fig. 20 shows a multiresolution morphing using uniform remeshing. A multiresolution morphing using adaptive remeshing is depicted in Fig. 21. Fig. 22 shows a morphing sequence with remeshing of coarse level; but visually enhanced using normal maps. Fig. 23 shows the morphing sequence from a pig model to a triceratops model. Fig. 24 shows the morphing sequence from a triceratops model to a horse model.

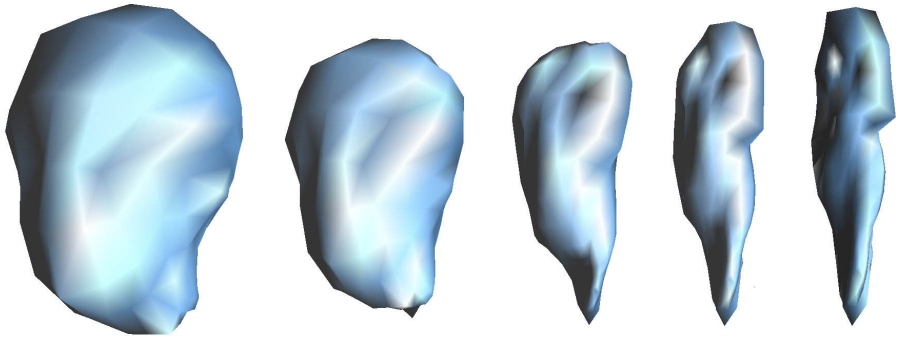
5.2 Multi-Target Morphing

As shown in Fig. 13, the common dissection for multiple objects can be established. Based on the constructed semi-regular meshes, we can produce any morphing sequence among these objects. Fig. 25 shows the result.

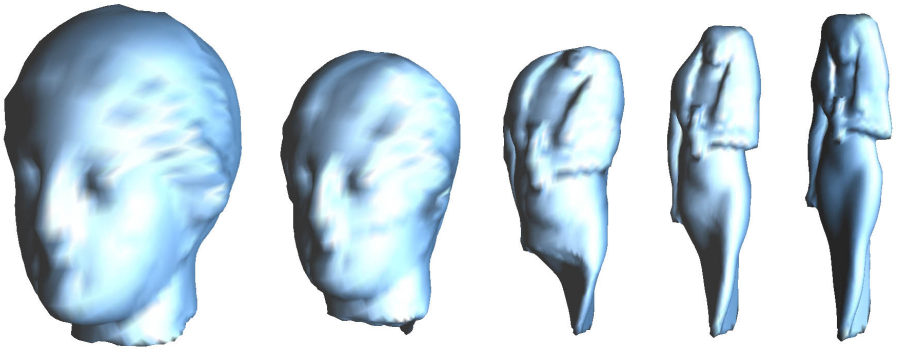
6 Concluding Remarks

The correspondence establishment among multiple objects is a versatile algorithm in computer graphics and geometry computing, especially in the morphing applications. Other applications such as geometry processing also benefit from the correspondences establishment. However, large efforts required by users suppress the establishment. A simpler and intuitive framework is necessary for alleviating the effors.

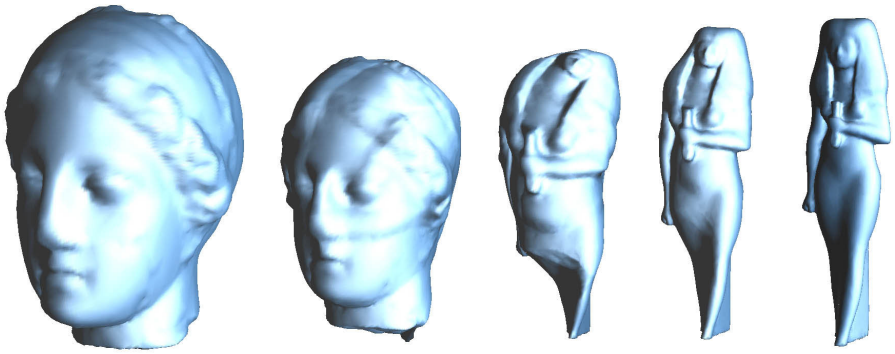
We have proposed a systematic method called recursive quinary subdivision to find a common



(a) M^2: 224 faces



(b) M^4: 3584 faces



(c) M^6: 57344 faces

Figure 20: Multiresolution morphing using uniform remeshing.

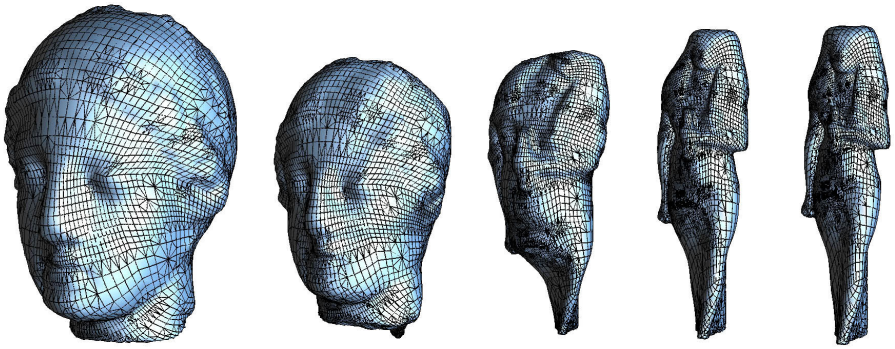


Figure 21: Morphing by adaptive remeshing ($\epsilon = 0.0025$: 8656 faces).

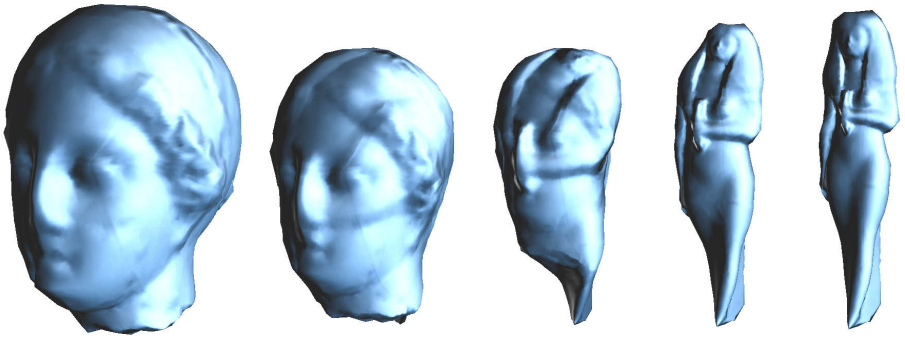
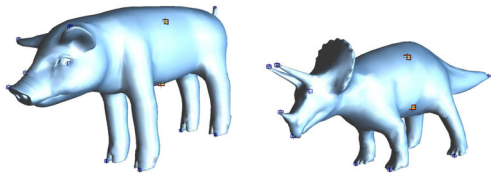


Figure 22: Morphing with normal mapping (M^3: 896 faces).

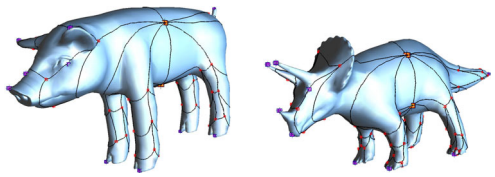
dissection for multiple objects with only four feature points specified by the user. Extra feature points in correspondences can also be specified for semantics and aligned during the subdivision. Based on this dissection, uniform and adaptive remeshing can be performed to yeild a set of semi-regular meshes. Moveover, geometric details can easily be resampled and stored as normal maps to improve the visual effects using the modern graphics hardware. We have demonstrated the 3D mesh morphing application between two or more objects using the correspondence established by the common dissection and remeshing. In addition to morphing in spatial domain, scheduled morphing between objects in wavelet domain is also demonstrated.

References

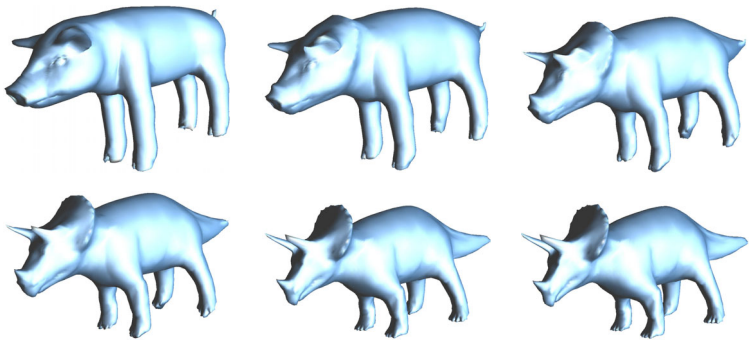
[1] M. Alexa. Mesh morphing. In *EUROGRAPHICS'01 State of The Art Report*, 2001.



(a) User-specified feature points



(b) Common dissection



(c) Morphing sequence

Figure 23: Morphing from a pig model to a triceratops model.

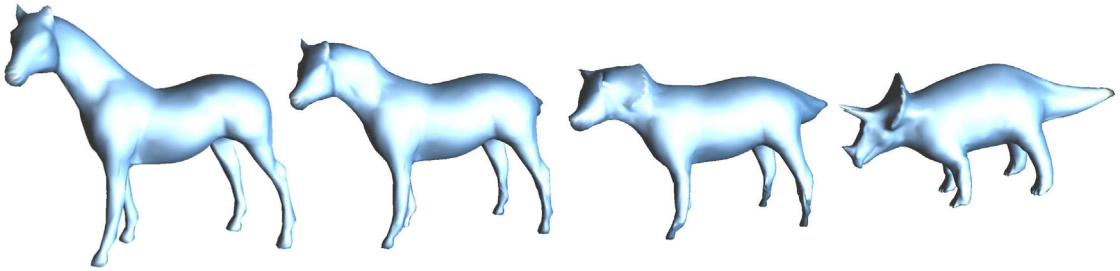
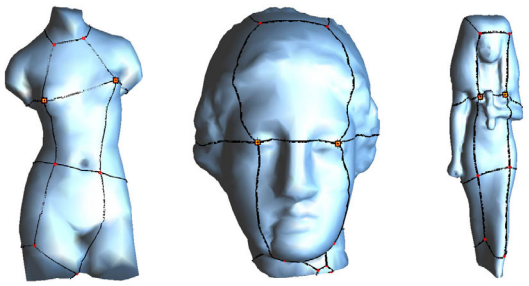
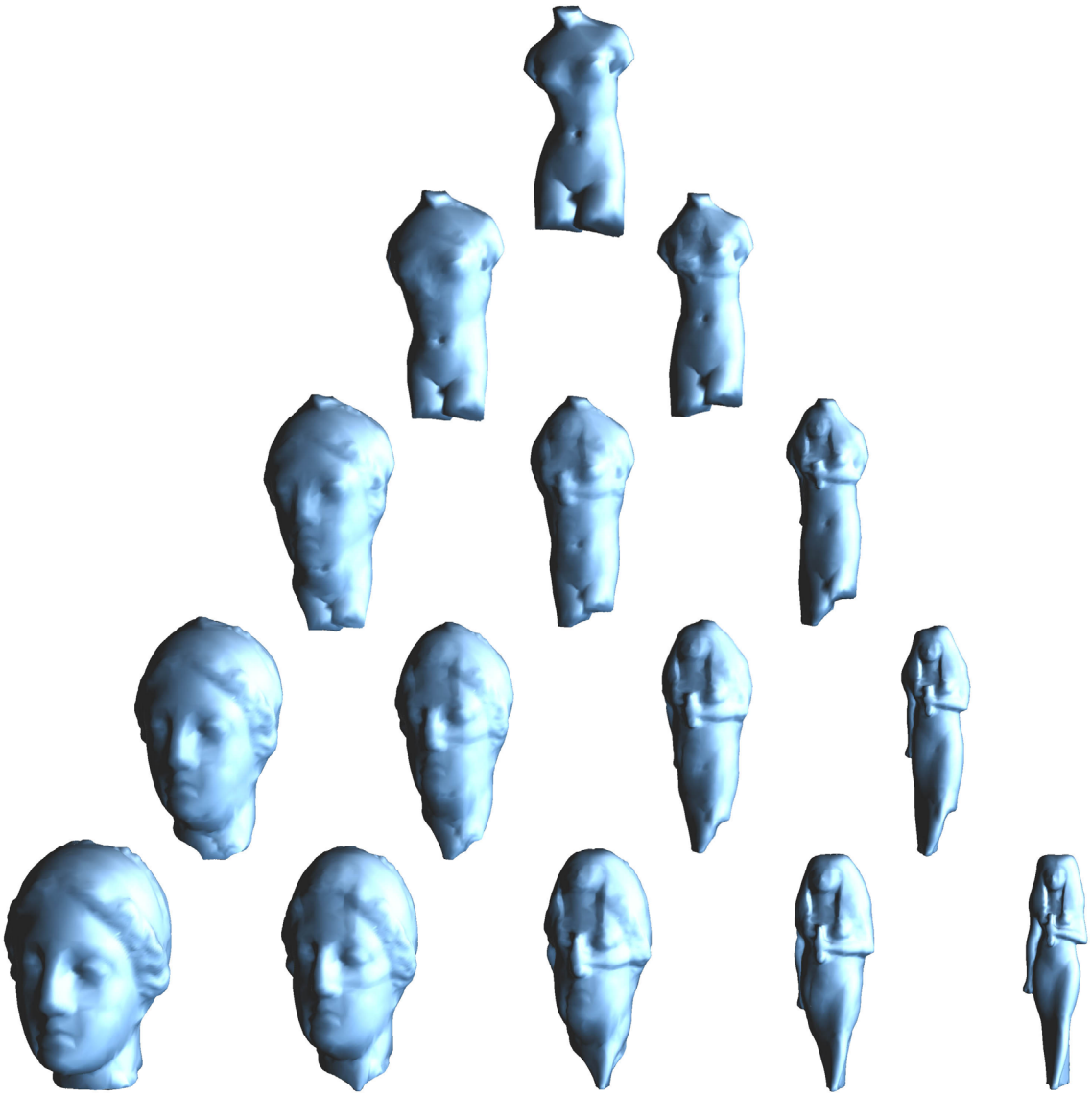


Figure 24: Morphing example by the dissection shown in Fig. 12 (M^5: 71680 faces).



(a) The common dissection.



(b) All morphing sequences.

Figure 25: Multi-target morphing (M^5: 14336 faces).

[2] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: Measuring error on simplified surfaces. In *Computer Graphics Forum*, 1998.

[3] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, 1990.

[4] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *Proceedings of SIGGRAPH'95*, 1995.

[5] M. S. Floater. Parametrization and smooth approximation of surface triangulation. In *Computer Aided Geometric Design*, 1997.

[6] K. Fujimura and M. Makarov. Foldover-free image warping. *Graphical models and image processing*, 60(2):100–111, March 1998.

[7] A. Gregory, A. State, M. Lin, D. Manocha, and M. Livingston. Feature-based surface decomposition for correspondence and morphing between polyhedra. In *Computer Animation'98*, June 1998.

[8] I. Guskov, K. Vidimče, W. Sweldens, and P. Schröder. Normal meshes. In *Proceedings of SIGGRAPH'00*, 2000.

[9] T. Kanai and H. Suzuki. Approximate shortest path on polyhedral surface and its applications. *Computer-Aided Design*, 33(11):801–811, September 2001.

[10] T. Kanai, H. Suzuki, and F. Kimura. Three-dimensional geometric metamorphosis based on harmonic maps. *The Visual Computer*, 14(4):166–176, 1998.

[11] T. Kanai, H. Suzuki, and F. Kimura. Metamorphosis of arbitrary triangular meshes. *IEEE Computer Graphics & Applications*, 20(2):62–75, March/April 2000.

[12] A. Khodakovsky, P. Schröder, and W. Sweldens. Progressive geomtry compression. In *Proceedings of SIGGRAPH'00*, 2000.

[13] M. J. Kilgard. A practical and robust bump-mapping technique for today's gpus. In *Game Developer's Conference 2000 Course "Adavanced OpenGL Game Development"*, 2000.

[14] F. Lazarus and A. Verroust. Three-dimensional metamorphosis: a survey. *The Visual Computer*, 14(4):373–389, 1998.

[15] A. W.F. Lee, D. Dobkin, W. Sweldens, and P. Schröder. Multiresolution mesh morphing. In *Proceedings of SIGGRAPH'99*, 1999.

[16] A. W.F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. MAPS: Multiresolution adaptive parameterization of surfaces. In *Proceedings of SIGGRAPH'98*, 1998.

[17] M. Lounsbery. *Multiresolution Analysis for Surfaces of Arbitrary Topological Type*. PhD thesis, Department of Computer Science and Engineering, University of Washington, 1994.

[18] M. Lounsbery, T. DeRose, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Transactions on Graphics*, 16(1):34–73, 1997.

[19] T. Michikawa, T. Kanai, M. Fujita, and H. Chiyokura. Multiresolution interpolation meshes. In *Proceedings of Pacific Graphics'01*, 2001.

[20] E. Praun, W. Sweldens, and P. Schröder. Consistent mesh parameterizations. In *Proceedings of SIGGRAPH'01*, 2001.

[21] P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe. Texture mapping progressive meshes. In *Proceedings of SIGGRAPH'01*, 2001.

[22] M. Zöckler, D. Stalling, and H. C. Hege. Fast and intuitive generation of geometric shape transitions. *The Visual Computer*, 16(5):241–253, 2000.