行政院國家科學委員會專題研究計畫　期中進度報告

三維網格參數化及其應用之研究(1/2)

計畫類別：個別型計畫
計畫編號：NSC92-2213-E-009-083-
執行期間：92 年 08 月 01 日至 93 年 07 月 31 日
執行單位：國立交通大學資訊工程學系

計畫主持人：莊榮宏

報告類型：精簡報告

處理方式：本計畫可公開查詢

中　華　民　國 93 年 6 月 1 日

摘要

我們提出"遞迴五元細分割"，能夠在不需要大量人爲設定的情形之下求得一個共同的切割方式。五元細分割方法遞迴地將一個參數化之後會高度延展的曲塊細切成五個新的小曲塊。這個演算法可以延伸至多個物體，而且將使用者額外所指定的特徵對應點也列入考慮來得到一個共同的切割方式。基於五元細分割架構，我們實作出將原始物體規則性或適應性重新取樣產生"半正規網格"表示法，以及產生二個或二個以上的網格物體在三度空間中或是小波空間中的變形動畫(Morphing)等應用。

關鍵字: 參數化，五元細分割，變形

Abstract

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 easily extended to multiple objects, taking into account the alignment of extra feature points, to derive a common dissection. Based on the dissection, we implement applications such as remeshing to yield a set of semi-regular meshes, and morphing between two or more objects.

Keywords: Mesh dissection, Parameterization, 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, 17, 16].

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 one of the vital factor that dominate 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 parameterizations 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 [19, 18, 11, 7, 21] leave this problem to users, 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 [19], which require the establishment of correspondences among multiple objects, benefit from consistent parameterizations.

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 parameterization 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 compute parameterization for each patch, 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 parameterization, 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

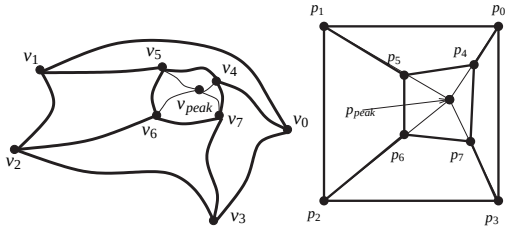
Remeshing process begins with an input irregular connectivity model, dissecting the model into a set of patches and then computing a parameterization for each patch. Eck et al. [4] proposed a method that produces a semi-regular mesh fully automatically by employing a Voronoi-like algorithm coupled with 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. [15] employed a mesh simplification algorithm to yield an initial dissection. Their algorithm is also automatic and more practical than Eck's. The *conformal mapping* is performed during mesh simplification and a patch parameterization 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. [19] 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 parameterizations. The survey of 3D morphing can be referred to [13], and for 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. [21] also require users to manually provide initial dissections. Among the methods proposed, Lee et al. utilized their MAPS to the mesh morphing application [14]. 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. [18] 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 Parameterization

3.1 Quinary Patch Subdivision

We use the parameterization scheme proposed by Floater [5] for the parameterization of patches. The L^2 stretch metric used in TMPM [20] is adopted 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. 1 illustrates the quinary subdivision. The ini-



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

Figure 1: Quinary subdivision illustration

tial patch is the region with corners v_0, v_1, v_2, and v_3 in R^3, denoted as $P_{\{0,1,2,3\}}$, where $p_0 = u(v_0)$,

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

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

And a constant ω for ω_i, $i = 0,1,2,3$ is used as

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

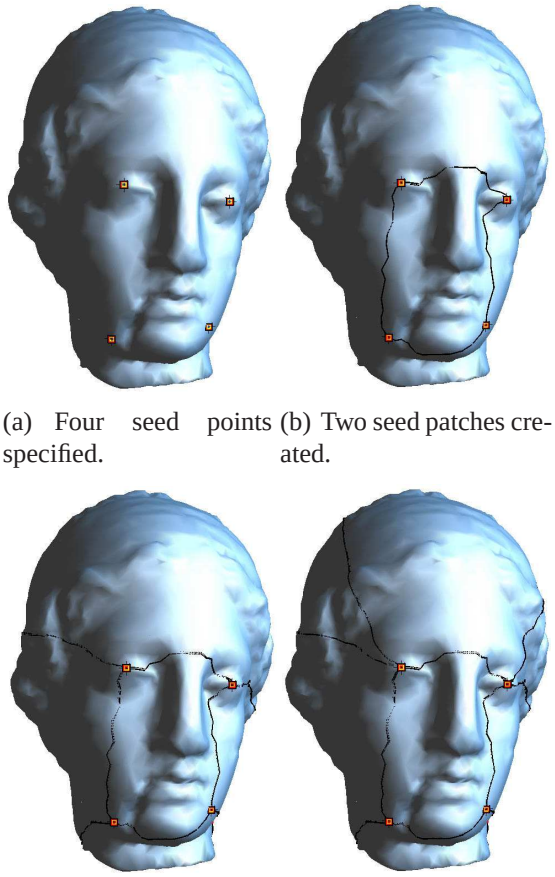
where ψ is a bias and is taken as 0.85 in our current implementation.

The four new corners are used to subdivide the patch into five patches $S_{\{0,1,5,4\}}$, $S_{\{1,2,6,5\}}$, $S_{\{2,3,7,6\}}$, $S_{\{3,0,4,7\}}$, and $S_{\{4,5,6,7\}}$. 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. Fig. 2 shows the process of quinary subdivision on a cat head model.

3.2 Dissection on a Single Object

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, and the close loop of the four points are computed by Dijkstra's shortest path algorithm [3] based on geodesic distance to dissect the input mesh. For each patch, if the stretch ratio exceeding a user-specified threshold, subdivide it into five patches using the quinary subdivision scheme and check the stretch ratios of the five new patches recursively until all patches satisfy the user-specified threshold. Then the dissection of a single object is found. Fig. 3 illustrates the dissection on a single object.

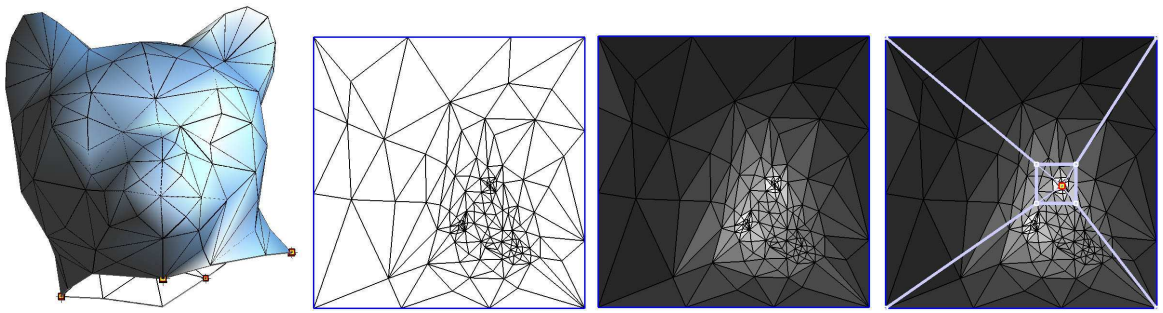
We can also perform patch boundary relaxation and global vertex relaxation (similar to [8]) to yield a better dissection. Fig. 4 shows the relaxation of a boundary curve. To improve the boundary curve with endpoints v_2 and v_5, the two incident patches of the boundary curve are used to to



(a) Four seed points specified. (b) Two seed patches created.

(c) Subdivision on the larger patch. (d) Result of recursive quinary subdivision.

Figure 3: Dissection on the venus head model.



(a) The original model. (b) The initial parameterization. (c) The face stretch ratio. (d) The quinary subdivision.

Figure 2: Quinary subdivision on cat head model.

construct the parameterization. The 3D curve on the mesh corresponding to the straight line from $u(v_2)$ to $u(v_5)$, which will be the new boundary curve. The position of a corner can be repositioned with similar process as shown in Fig. 5.

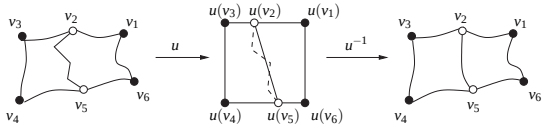


Figure 4: Relaxation of a boundary curve.

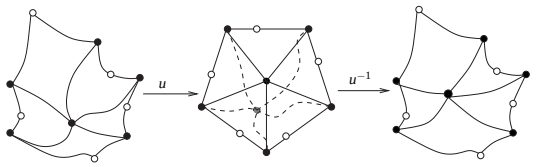


Figure 5: Relaxation of a corner vertex.

3.3 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. We simply take the union of these quinary trees, denoted as Q_{union}, and the initial seed points act as feature corresponding points. 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. The quinary subsivision should be performed on all corresponding patches of all objects if there is one patch with stretch ratio exceeds the threshold.

3.3.1 Extra Feature Correspondences

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,

To prevent from this situation, a *foldover-free warping* proposed by [6] is used in the parameter domain before each quinary subdivision. 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 first compute the averaged feature point position as

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

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 simul-

taneously. 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. An example is shown in Fig. 6, in which, besides four limbs, more feature points are specified for mouths, tail of the triceratops and the pig, horn of the triceratops and ear of the pig.

4 Applications

With the result of the parameterization, lots of applications can be performed. Two applications are discussed here, which are the remeshing and mesh morphing.

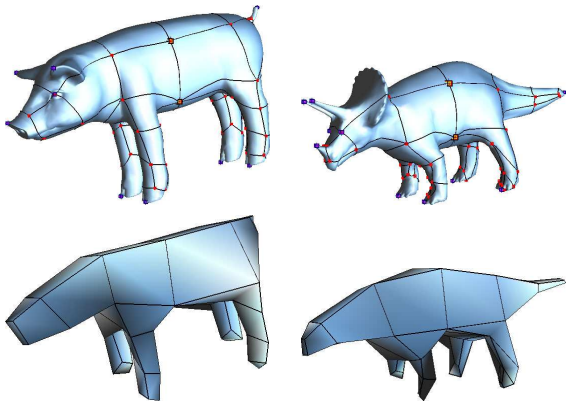
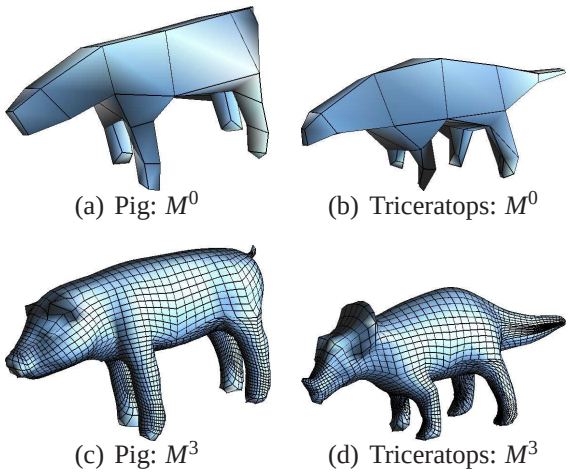


Figure 6: Common dissection and base domains of a pig model and a triceratops model.



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

(c) Pig: M^3 (d) Triceratops: M^3

Figure 7: The uniform remeshing derived from the dissection in Fig. 6.

4.1 Remeshing

4.1.1 Uniform Remeshing

With the parameterization ready for each patch of the base domain, we can start remeshing process. For each patch, we simply take regular grid point parameters and compute their inverse mapped 3D position. Fig. 7 shows the result of uniform remeshing derived from the dissection in Fig. 6.

4.1.2 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 $\mathscr{M}$, 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 $p \in g$.

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

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.,

$$E(q) = \max_{v \in P_q} d(v) \tag{3}$$

Fig. 8 illustrates the error function. The error

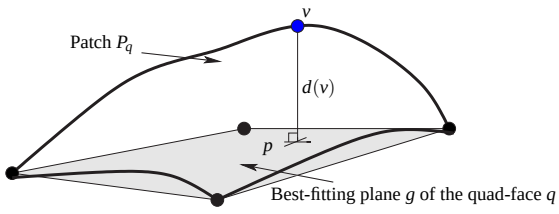
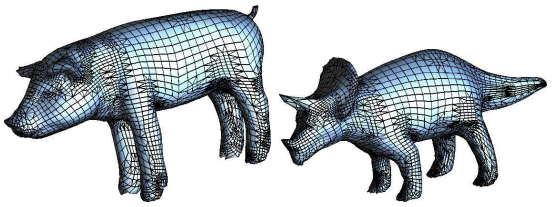


Figure 8: Error function definition.

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

$$E_N(q) = \frac{E(q)}{B(\mathscr{M})} \tag{4}$$

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.



(a) pig (b) triceratops

Figure 9: Adaptive remeshing ($\varepsilon = 0.0025$).

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 still be consistent.

$$E_C = \max_{1 \le i \le n} (E_N(q_i)) \tag{5}$$

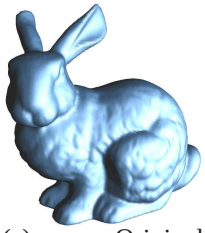
where n is the number of objects. This will also result in adaptive remeshes with the same connectivity structure. If there are some local geometric features in one object, which correspond to a flat region in another object, the flat region will be forced to refined and still result in lots of faces. Fig. 9 shows the result of the consistently adaptive remeshing.

4.1.3 Normal Mapping

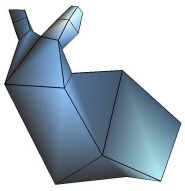
Normal map is an image storing quantized normal vectors of surfaces. By mapping (n_x, n_y, n_z) in the range $[-1,1]$ to (r,g,b) in the range $[0,255]$, it's possible to recover the detailed geometry feature from this map and improve the rendering visual effects. This is appealing while a remesh of coarse 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. No further packing algorithm such as pull-push algorithm in [20] is needed since the patches are all square in parameter domain. Fig. 10 shows the result with and without normal mapping.

4.1.4 Results of Remeshing

We have implemented quinary subdivision and remeshing as described above on a PC with Athlon 900Mhz CPU and NVidia geforce 2 graphics card. The remeshing results are evaluated by

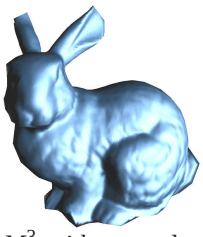


(a) Original model: 69664 faces



(b) Base domain: 34 faces

(c) M^3: 2176 faces



(d) M^3 with normal mapping

Figure 10: Normal mapping example.

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.

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.

Model	size							
	original	uniform remeshing						adaptive remeshing
		M^0	M^1	M^2	M^3	M^4	M^5	M^6	($\varepsilon = 0.0025$)
venus	10000	14	56	224	896	3854	14336	57344	8656
isis									
pig	7164	66	264	1056	4224	16896	67584	270336	10730
isis	5660								
body	1418	14	56	224	896	3854	14336	57344	14624
venus	5000								
isis	5000								

Table 1: Statistics of polygon numbers.

Model	L^2 error(%)							
	uniform remeshing							adaptive remeshing
	M^0	M^1	M^2	M^3	M^4	M^5	M^6	($\varepsilon = 0.0025$)
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
pig	3.09	1.73	1.16	0.55	0.287	0.124	0.058	0.085
triceratops	2.80	1.68	0.85	0.73	0.369	0.114	0.072	0.105
body	4.59	2.12	0.96	0.39	0.210	0.085	0.022	0.042
venus	5.73	2.87	0.99	0.38	0.143	0.057	0.021	0.054
isis	3.97	1.47	0.65	0.33	0.136	0.046	0.015	0.040

Table 2: Statistics of errors.

4.2 Mesh Morphing

4.2.1 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 morphed objects will be generated, and they still have multiresolution structures. Fig. 11 shows the morphing sequence from a pig model to a triceratops model.

Model	size	M^0	time(sec)	
			dissection	remeshing
venus+isis	10000+10000	14	4.715	1.688
horse+human	5000+5000	106	5.276	3.751
horse+triceratops	2000+5660	70	4.760	2.532
bunny	69664	34	108.729	8.714
body+venus+isis	1418+5000+5000	14	1.621	1.145
pig+triceratops	7164+5660	66	5.034	2.783

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

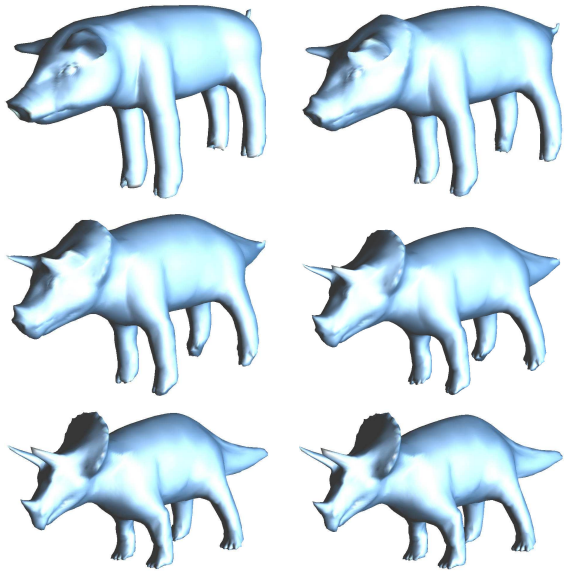


Figure 11: Morphing from a pig model to a triceratops model.

4.2.2 Multi-Target Morphing

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. 12 shows the result.

5 Conclusion

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

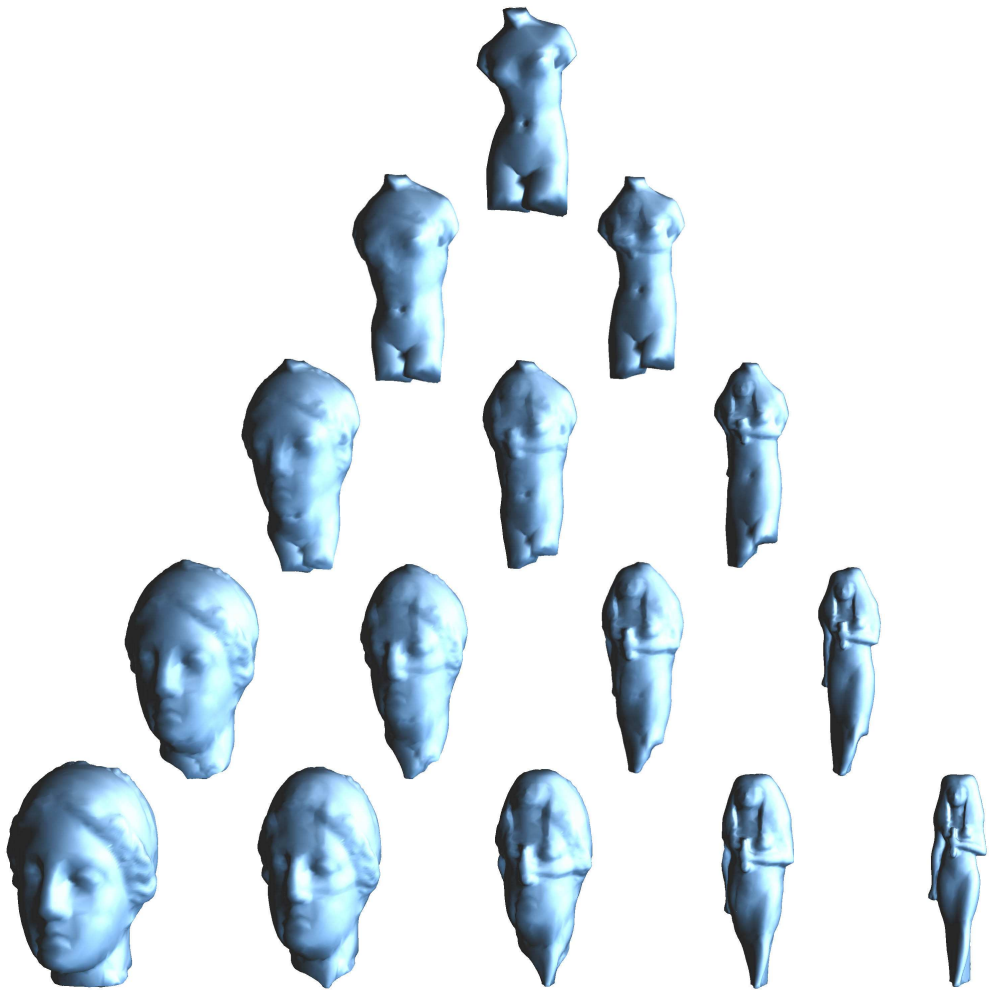


Figure 12: Multi-target morphing (M^5: 14336 faces).

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 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.

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

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

[15] 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.

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

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

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

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

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

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