Constructing Subdivision Connectivity Mesh via PDE Parameterization

Mingyong Pang[1,2], Alexei Sourin[1], Zhigeng Pan[2,3]

[1] School of Computer Engineering, Nanyang Technological University, Singapore
[2] Department of Educational Technology, Nanjing Normal University, China
[3] State Key Lab of CAD&CG, Zhejiang University，China

***Abstract*—In this paper we present a novel algorithm for constructing subdivision connectivity mesh from dense original mesh. Our algorithm begins from a coarse base mesh generated from the original mesh and then the original mesh is divided into a set of patches guided by the base mesh. The patches are subsequently parameterized onto a planar domain by the mean value coordinates method. For each mesh patch, four boundary condition curves are calculated via the parameterization for local PDE patch construction. Considering the boundary curves as shape control boundary curve conditions, a PDE patch can be built and its coefficients are evaluated from the boundary curves. The PDE patch gives an explicit parametric representation of the mesh patch. Finally, all the PDE patches are remeshed via the planar parametric domains and a new resampled mesh with subdivision connectivity can be obtained with an arbitrary resolution.**

***Keywords*—digital geometry processing, multiresolution mesh, remeshing method, subdivision connectivity, partial differential equations.**

I. INTRODUCTION

Succeeding to character, sound, image and video, 3D geometry model, i.e., mesh or point-set, is currently the fifth kind of media representations of real world. Dense triangular meshes routinely result from a number of 3D acquisition techniques, e.g., laser range scanning and MRI volumetric imaging followed by isosurface extraction. Due to its priority of convenient processing and versatile ability of shape expression, triangular mesh has become the dominant representation of 3D geometry objects in computer graphics society. The triangulations form a surface of arbitrary topology genus, boundaries, connected components, and have irregular connectivity. Because of their complex structure and tremendous size, these meshes are awkward to handle in such common tasks as storage, display, editing, and transmission.

Multiresolution representations are now established as a fundamental component in addressing these issues. There are two classes of multiresolution technologies. One approach extends classical multiresolution analysis and subdivision techniques to arbitrary topology surfaces, e.g., [1, 2, 3]. The alternative is more general and is based on the sequential mesh simplification [4], e.g., progressive meshes [5]. In either case, the objective is to represent triangulated 2-manifolds in an efficient and flexible way, and to use this description in fast algorithms addressing the challenges mentioned above.

In many cases, we prefer to remesh the original meshes to obtain desirable substitutes rather than using the original models or their optimized versions with the same connectivity. A lot of algorithms have been implemented to remesh original mesh for different applications such as MAPS in [6] or normal meshes [7, 8]. In a general way, remeshing should conform to the following method [6] in which the input mesh should be first parameterized, then resampled in a parameter space, and finally the resampled points are mapped onto the surface of the original 3D mesh. Ideally, mesh surface is parameterized over a base domain consisting of a small number of triangles. Once a surface is understood as a function mapping from the base domain into R3 or higher dimension when surface attributes are considered, many tools from areas such as approximation theory, signal processing, and numerical analysis are at our disposal. In particular, classical multiresolution analysis can be used in the design and analysis of algorithms. For example, error controlled, adaptive remeshing can be performed easily and efficiently.

Eck et al [1] first divided the original mesh surface into patches using the approximating Voronoi diagrams, which naturally generate a Delaunay triangulation, i.e., a dual graph of the Voronoi diagram. The algorithm can process various original meshes with arbitrary genus and obtain the high quality remeshing with subdivision connectivity structures [9]. But the algorithms mentioned here are very difficult to implement due to their complex and inefficient computations. In [6], Lee et al constructed smooth parameterizations of irregular connectivity triangulation of arbitrary genus 2-manifolds and then resampled the original mesh to construct connectivity mesh based on the parameterization. The algorithm uses hierarchical simplification to efficiently induce a parameterization of the original mesh over a base domain consisting of a small number of triangles. This initial parameterization is further improved through a hierarchical smoothing procedure based on Loop subdivision [10] applied in the parameter domain.

Just as pointed out in [8], one of the fundamental issues of surface representation is about the relationship between the approximating quality and size of the data. Even though a full theoretical characterization is not yet available now, the practical importance of efficient representation for digital

Manuscript Received on 26 April 2010
E-Mail: panion@netease.com

geometry processing is so great that a broad variety of algorithms have been put forward. Of particular interest in the context of display, editing and compression are multiresolution representations based on irregular and semi-regular meshes, i.e., subdivision connectivity meshes. The latter have many connections with classical functional representations such as wavelets and Laplacian pyramids, which can be leveraged for digital geometry processing applications [11]. Once a mesh with subdivision connectivity is constructed, some geometric data compression method can also be taken into account [7, 8, 12]. Thus, how to construct subdivision connectivity meshes is an essential technique for multiresolution applications of mesh models.

In this paper, we propose a novel algorithm for constructing semi-regular meshes from their dense irregular original meshes. Our algorithm belongs to the same group as the algorithm presented in [6] based on extended classical multiresolution analysis and subdivision techniques, but it also draws on the ideas from the other algorithm group based on sequential mesh simplification. An important component of our algorithm different from [6] and other remeshing method is that partial differential equation method is used in the parameterization and resampling process. This provides a two-level multiresolution representation of mesh model, i.e., functional approximation of local surface patch and multiresolution polygonization of model, with subdivision connectivity.

II. TALGORITHM DESCRIPTION

Our approach first takes a dense original irregular mesh as its input. A base mesh is then built from the original one by a modified mesh simplification method based on Garland's algorithm using quadric error metrics. For each facet of the base mesh, a local mesh patch on the original mesh is defined, thus an atlas of patches can be obtained for the total mesh model. The patches are further parameterized on its corresponding base facets respectively by Floater's mean value coordinates method proposed in [13]. For each mesh patch, four boundary condition curves, which are defined by four discrete point lists on the original mesh, can be calculated via the parameterization for evaluating coefficients of local PDE patches to be constructed. Considering the boundary curves as shape control conditions, a PDE patch can be evaluated, which is also a parameterization of the local mesh patch of the original mesh model. Finally, all the PDE patches are remeshed via the triangular parameter domains, i.e., the facets of the base mesh, and a new resampled mesh with subdivision connectivity can be obtained.

The steps of our algorithm are described in details below.

2.1 Building the Base Mesh Using Decimation.

First, we consider as a foundation of our method the work done by Garland et al in [4]. Their surface simplification algorithm can rapidly and efficiently produce high quality approximations of complex polygonal models by using iterative contractions of vertex pairs in models while maintaining surface error approximations using quadric matrices. By contracting arbitrary vertex pairs (not just edges), the algorithm is able to join unconnected regions of models. However, Garland's method has two weak points for our subdivision connectivity mesh construction:

First, it introduces many new vertices in the simplified mesh. Second, it can not guarantee that surface topology of the simplified mesh will correspond to the topology of its initial mesh.

To solve the first problem, our algorithm uses a special edge collapse atomic operator to simplify an edge rather than the method done in the original algorithm [4]. With reference to Fig.1, when the edge v0v1 is to be collapsed, there is no new vertex created and the edge collapses to one of its ends, v0 or v1, which leads to a smaller simplified error.

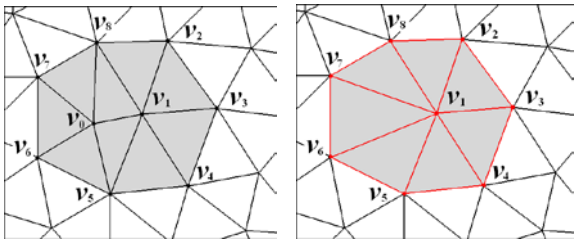


Fig. 1. Edge collapse operator used in our algorithm.

To solve the second problem, the algorithm imposes a rigorous restriction on the edge collapse to keep the simplified surface topology consistent with the initial surface when the decimation operator is performed. In Fig.1, once the edge v0v1 is collapsed to vertex v1, all the highlighted edges of the facets neighboring to v1 are "frozen", that is to say, these edges can't be collapsed anymore until all the edges in the current model are checked. Once all the edges become frozen, we say that the algorithm finishes a level of decimation. At this time, the algorithm unfreezes all the edges in the current mesh and starts its decimation work towards another level. The decimation of levels can keep the surface topology preserved and can distribute triangles of the initial complex mesh in the surface patches uniformly.

Obviously, it is very important how to select an edge to be collapsed in each decimation step. In [4], Garland et al selected the edge with the smallest decimation error to perform the atomic operator. They use the quadric error metrics to evaluate an error for each edge in the current mesh. In our work, we use a light technique presented in [14] to evaluate collapse errors for the edges. The basic idea behind the technique is to use fewer triangles in flat area on mesh model and more triangles in the areas with a high curvature. So, our algorithm evaluates the curvature of an edge as:

$$C = \max_{T_i \in T(u)} \{ \min_{T_j \in T(u,v)} \{(1 - T_i.norm \cdot T_j.norm\} / 2\} \qquad (1)$$

where, T(u) is the set of triangles neighboring the vertex u, T(u,v) is the set of triangle neighboring both the vertices u and v, and Ti.norm is the unit normal of facet Ti.

Because $T_i.norm \cdot T_j.norm$ is equal to $\| T_i.norm \| \cdot \| T_j.norm \| \cdot \cos\alpha$, where α is the angle between $T_i.norm$

and $T_j.norm$, we have

$$C = \max_{T_i \in T(u)} \{ \min_{T_j \in T(u,v)} \{(1-\cos\alpha\} / 2\} \quad (2)$$

for $\| T_i.norm \| \cdot \| T_j.norm \| = 1$, and we further have

$$C = \max_{T_i \in T(u)} \{ \min_{T_j \in T(u,v)} \sin^2 \frac{\alpha}{2}\} \quad (3)$$

The length of each edge should also be considered as an important factor of the edge in mesh, so the error or weight to collapse the edge can be defined as

$$weight(\boldsymbol{u},\boldsymbol{v}) = \| \boldsymbol{u} - \boldsymbol{v} \| \times \max_{T_i \in T(u)} \{ \min_{T_j \in T(u,v)} \sin^2 \frac{\alpha}{2}\} \quad (4)$$

On the other hand, the higher degree a vertex has in a mesh, the more power it has to control the local shape of the model, i.e., removing the high degree vertex with edge collapse can cause big change to the model appearance. So our algorithm should prevent the vertices with higher degrees to be deleted during simplification. We can therefore rewrite the weight of edge collapse as

$$weight(\boldsymbol{u},\boldsymbol{v}) = \| \boldsymbol{u} - \boldsymbol{v} \| \times \max_{T_i \in T(u)} \{ \min_{T_j \in T(u,v)} \sin^2 \frac{\alpha}{2}\} \times D^2 \quad (5)$$

where D is the degree of vertex u or v to be deleted.

A heap data structure is used in our method to manage the errors of edges to rapidly access the edge with the smallest error. Once an edge is collapsed, the errors of edges related to the collapsed edge are updated immediately and the heap is also adjusted at the same time.

Since the edge collapse operator always contracts an edge to one of its ends, it can easily create triangles with a bad aspect ratio. These thin triangles can result in a low quality mesh. Thus, we should find a method to restrain the thin triangles. For this purpose, we introduce a shape measure of the triangle:

$$Q = \frac{4 \cdot a \cdot \sqrt{3}}{l_0^2 + l_1^2 + l_2^2} \quad (6)$$

where, a and $l_i\,(i = 0,1,2)$ are the area and edge lengths of the triangle. Hence, Q is 1 for an equilateral triangle and Q equals to 0 when the triangle is degenerated. So we can define a threshold degree for the triangles to be thin at which the edge collapse has to be canceled.

As mentioned above, most of the existing mesh decimation algorithms are not able to preserve the topology of a simple mesh consistent with the initial complex mesh. For example, Garland's mesh decimation algorithm can simplify an initial manifold surface to a non-manifold one [10]. This is bad for post-processing when constructing inner PDE boundary condition curves in a patch. In our method, we use a strict checking condition to prevent the topological change of the surface mesh. Before an edge is collapsed, we check if it satisfies the condition, otherwise the edge collapse will be refused.

The checking condition is as follows: without loss of generality, let us suppose the edge to be collapsed is $\boldsymbol{v}_0\boldsymbol{v}_1$, and $N(\boldsymbol{v}_0)$ and $N(\boldsymbol{v}_1)$ are the sets of neighboring vertices respectively.

If $\| N(\boldsymbol{v}_0) \cup N(\boldsymbol{v}_1) \| < (deg(\mathbf{v}_0) + deg(\mathbf{v}_1) - 2)$, where $\deg(\boldsymbol{v}_i)\ (i = 0,1)$ are the degrees of $\boldsymbol{v}_0$ and $\boldsymbol{v}_1$, and $\| S \|$ is the number of the components in set S, we consider the edge collapse may change the topology of the mesh surface. In Fig.5, a few simplified polygon meshes built with our algorithm are presented.

2.2 Parameterizing Patches

Given any two surfaces with similar topology it is possible to find a one-to-one mapping between them [15]. If one of these surfaces is represented by a triangular mesh, the problem of computing such a mapping is referred to finding a mesh parameterization [16]. The surface that the mesh is mapped to is typically referred to as the parameter domain. The input surfaces with disk-like topology can be directly parameterized on a planar domain. Many planar paramerterization methods have been proposed ever since Tutte's theorem [17]. Though any planar parameterization would be suitable for our considerations, we recommend Floater's mean-value theorem [16] because of its angle preserving and robustness.

Let $P = (V, E, F)$ be a local mesh patch with the vertex set $V = \{\boldsymbol{v}_i = (x_i, y_i, z_i), 1 \le i \le N\}$, where $\boldsymbol{v}_i (i = 1, \mathrm{L}, n)$ are the internal vertices and $\boldsymbol{v}_{n+1}, \boldsymbol{v}_{n+2}, \mathrm{L}, \boldsymbol{v}_N$ are the boundary vertices in any anticlockwise sequence, E and F are the edge and facet set of the patch, respectively. We now choose some points, $\boldsymbol{p}_{n+1}, \boldsymbol{p}_{n+2}, \mathrm{L}, \boldsymbol{p}_N$, to be the vertices of any planar triangle $D \subset \boldsymbol{R}^2$ in an counter-clockwise sequence. For each $\boldsymbol{v}_i\ (i = 1, \mathrm{L}, n)$, we choose a set of real numbers $\lambda_{i,j}$ for $\boldsymbol{v}_i (i = 1, \mathrm{L}, N)$, such that

$$\lambda_{i,j} = \begin{cases} 0 & edge(\boldsymbol{v}_i \boldsymbol{v}_j) \in E \\ > 0 & edge(\boldsymbol{v}_i \boldsymbol{v}_j) \notin E \end{cases} \quad (7)$$

and

$$\sum_{j=1}^{N} \lambda_{i,j} = 1 \quad (8)$$

Further, let's define $\boldsymbol{p}_1, \boldsymbol{p}_2, \mathrm{L}, \boldsymbol{p}_n$ to be the solutions of the linear system of equations,

$$\boldsymbol{p}_i = \sum_{j=1}^{N} \lambda_{i,j} \boldsymbol{p}_j,\ i = 1, \cdots, n \quad (9)$$

In order to establish a one-to-one mapping from a local mesh patch on the original mesh to planar parameter domain, i.e., theplanar triangular, we need to solve a large sparse linear system with coefficients, i.e. an equation of the form, derived from Eqn.(9):

$$\boldsymbol{A}\boldsymbol{X} = \boldsymbol{b}, \quad (10)$$

where, $\boldsymbol{A}$ is a $n \times n$ non-singular square matrix, $\boldsymbol{b}$ is a vector of dimension n, and $\boldsymbol{X}$ is the unknown vector of dimension n we need. Many solver tools can be used such as Successive Over-Relaxation, Bi-Conjugate Gradient [18], and Matlab API functions. After we fix the boundary vertices of the original local mesh patch to the boundary of the planar triangle, then all "inner vertices" of the patch and all "inner points" of the triangle

become a one-to-one mapping through solving a sparse liner system, Eqn.(10).

2.3 Constructing Boundary Curves of PDE Patches

Once a simplified version of the initial complex mesh is derived, we can define a patch on the initial model for each facet of the simplified mesh and parameterize all the patches onto a planar triangle in a plane respectively. Thus, constructing the PDE boundary condition curves on the patch becomes a natural thing based on the patch parameterizations.

Because there are no new vertices introduced in the simple mesh during the decimation, it is very easy to find a corresponding vertex in the initial mesh for each vertex in the simplified mesh. This gives us an easy way to define a patch for each facet of the simplified mesh. There are many methods which can do this work. For example, we can take the initial mesh as a graph embedded in a 3D space. Thus, for each edge of a triangle facet in the simplified mesh, we can use the Dijkstra algorithm to find the consecutive edges of the initial complex mesh to connect the ends of the edge [19]. However, the consecutive edges do not form the shortest path between the two vertices of the initial mesh model. The shortest path should be a geodesic path, which typically cuts across facets in the mesh and therefore cannot be found by the traditional graph-based Dijkstra algorithm for the shortest paths. In order to achieve an accurate boundary of each patch of the initial mesh, we employ a method of geodesics calculation to find an optimized curve for the patch boundary.

The exact geodesic algorithm was described by Chen et al. in [20] and partially implemented by Kaneva et al. [21]. The algorithm has a complexity of $O(n^2)$. Another algorithm with $O(n \log^2 n)$ time for the "single source, single destination" geometric path between two given mesh vertices was described by Kapoor [22], which is a very complicated method with many other computational geometry algorithms used in it. There are also some approximating algorithms [23] with pre-given error bounds in which some extra edges are added and the traditional Dijkstra algorithm is employed to obtain the approximating shortest paths. Martinez et al. [24] presented an iterative optimization algorithm, but its results significantly depend on the initial approximation path. A detailed survey of approximation algorithms for graph and geodesic search can be found in [25].

To calculate geodesics on the mesh model, we use Surazhsky's implementation of the continuous Dijkstra-like algorithm presented in [26]. In the algorithm, the shortest paths can be visualized as rays cast from the source vertex in all tangent directions. Within a triangle, the shortest path is a straight line. When crossing an edge, the shortest path must correspond to a straight line if the two adjacent facets are unfolded into a common plane. The basic idea of the algorithm is to track together groups of the shortest paths that can be parameterized atomically and this is achieved by partitioning each mesh edge into a set of intervals called windows. The windows are then propagated across the facets of the mesh in Dijkstra-like sweep. In order to optimize the performance of the algorithm, all propagating windows are organized as a wave-front by ordering them in the queue according to minimal distances from the source vertex.

Once all the edges are covered by the windows representing geodesic distance, it is very easy to trace the shortest path from any surface point to the source by back-tracing. For each triangular facet of the simplified mesh, it is easy to compute the curve boundary on the initial mesh by calculating the geodesics between vertices of the facet. The examples of the curve patches created by calculating geodesics are given in Fig.6.

2.4 Sampling PDE Boundary Curves

Now, that we have derived the curved patches on the initial mesh, we need to define the PDE boundary curves for each patch in order to prepare the boundary condition data for the PDE patch construction. Our method first defines a set of isometric polylines in the triangular base facets of the simplified mesh as parameter curves, just like it is illustrated in Fig.2.

Then, the algorithm samples the polylines and maps the sampled points onto the surface of the initial mesh via the patch parameterization. For example, let $\boldsymbol{p}$ is a sampled point on the parameter curves and $\boldsymbol{p}_i, \boldsymbol{p}_j, \boldsymbol{p}_k$ are the three nearest points of the planar parameterized mesh. It is not difficult to evaluate the barycentric coordinates, c_1, c_2, c_3, of $\boldsymbol{p}$ with respect to $\boldsymbol{p}_i, \boldsymbol{p}_j, \boldsymbol{p}_k$, i.e.,

$$\boldsymbol{p} = c_i \boldsymbol{p}_i + c_j \boldsymbol{p}_j + c_k \boldsymbol{p}_k \tag{11}$$

Obviously, $\boldsymbol{p}_i, \boldsymbol{p}_j, \boldsymbol{p}_k$ are the parameter points of mesh vertices of $\boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{v}_k$. Thus it is easy to calculate the 3D position $\boldsymbol{v}$ on the original mesh for parameter point $\boldsymbol{p}$, i.e.,

$$\boldsymbol{v} = c_i \boldsymbol{v}_i + c_j \boldsymbol{v}_j + c_k \boldsymbol{v}_k \tag{12}$$

The boundary curves of the patches are also to be resampled to create the most outer boundary data. Eventually, all the data set are stored in a file which is then processed by the PDE engine for PDE coefficient calculation and rendering. In Fig.7, we illustrate two examples of the boundary condition curves created by our algorithm.

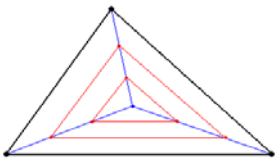


Fig. 2. Defining boundary curves in triangular parameter domain.

2.5 PDE Patchwise Approximation

Bloor et al [27] has done the pioneer research on using PDEs in computer graphics by producing a parametric surface S(u, v), defined as a solution to an elliptic fourth order PDE:

$$\left(\frac{\partial^2}{\partial u^2} + a^2 \frac{\partial^2}{\partial v^2} \right)^2 S(u,v) = 0 \tag{13}$$

where a ≥ 1 is a parameter that controls the relative rates of smoothing between the u and v parameter directions. The partial differential operator in Eqn.(13) represents a smoothing process in which the value of the function at any point on the surface can be understood as a weighted average of the surrounding values. Thus, a single PDE patch of a surface is obtained as a smooth transition between the boundary conditions. Commonly, the PDE boundary curves are extracted in a circular though not necessarily co-planar way around the object so that the whole reconstructed object is a single entity in its parametric space, and that each PDE patch borders with maximum two other patches (Fig.3, left). In the patchwise PDE method [28], each shape is represented by patches with their own uv coordinate system (Fig.3, right). This configuration allows for representing shapes with branches and for preservation of irregular and sharp details on the object surface by matching the respective patches with various sizes and orientations to the surface.

With reference to [28], each PDE patch P(u, v) can be formulated by Eqn.(13). Assuming that the effective region in the uv space is restricted to $0 \le u \le 1$ and $0 \le v \le 2\pi$, and using the method of separation of variables, the analytic solution to Eqn.(13) is generally given by

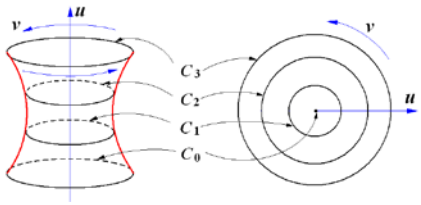


Fig.3. Illustration of PDE patch configurations: Conventional (left) and the patchwise (right) configurations.

$$P(u,v) = A_0(u) + \sum_{n=1}^{\infty}\left[A_n(u)\cos(nv) + B_n(u)\sin(nv)\right]$$

where

$$A_0(u) = \alpha_{00} + \alpha_{01}u + \alpha_{02}u^2 + \alpha_{03}u^3 \quad (14)$$

$$A_n(u) = \alpha_{n1}e^{anu} + \alpha_{n2}ue^{anu} + \alpha_{n3}e^{-anu} + \alpha_{n4}ue^{-anu} \quad (15)$$

$$B_n(u) = \beta_{n1}e^{anu} + \beta_{n2}ue^{anu} + \beta_{n3}e^{-anu} + \beta_{n4}ue^{-anu} \quad (16)$$

and PDE coefficients α00, α01, …, αn3, αn4 and β11, β12,…, βn3, βn4 are vector-valued and determined by the boundary conditions. The solution is the sum of signals with different frequencies. It can be observed that the low frequency signals contain the most essential geometric information while the higher frequency ones can be neglected. Therefore, Eqn.(13) can be rewritten as:

$$P(u,v) = A_0(u) + \sum_{n=1}^{N}\left[A_n(u)\cos(nv) + B_n(u)\sin(nv)\right] \quad (17)$$

where only N lower frequencies are selected. Strictly speaking, this equation needs a remainder term to guarantee that the boundary conditions are satisfied. The boundary conditions imposed on Eqn.(17) take the following forms:

$$P(0, v) = C_0(v) \qquad P(u_1, v) = C_1(v) \quad (18)$$
$$P(u_2, v) = C_2(v) \qquad P(1, v) = C_3(v)$$

where C0, C1, C2 and C3 are isoparm boundary curves on the surface patch at u = 0, u1, u2, and u = 1, respectively, and 0<u1, u2<1. Fig.3 (right) illustrates the layout of the PDE boundary curves for an individual PDE patch, where C0 degenerates into one point.

From the boundary condition samples, our algorithm automatically solves Fourier coefficients of PDE patches according to Eqn.(14), (15) and (16). Once the coefficients are derived, the PDE solution in Eqn.(17) naturally gives a parameterization of each patch. It also provides a convenient way for polygonization to visualize the patches.

2.6 Subdivision Connectivity Polygonization

For all patches, our algorithm defines a uniform mask of subdivision connectivity polygonization with user-defined level of multi-resolution. The mask allows for keeping the same mesh topology for patches. In Fig.4, we illustrate how the mask is defined. In the mask, some edges consist of loop curves around the center of the base triangle of the patch. The loop curves are defined as a set of concentric circles on the polar parameter plane. For each edge of the base triangle, the samples of it exactly distribute to one of the 120o angle sectors on polar circles. Once the (u, v) parameter coordinates are derived, their 3D Cartesian geometric coordinates can easily be computed according to Eqn.(17). Examples of such isoparametric curves on PDE patches are given in Fig.8.

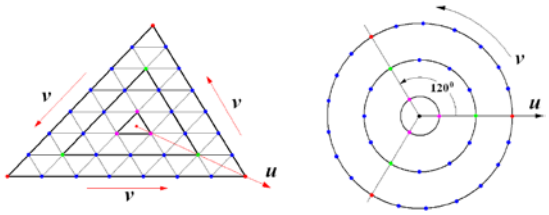
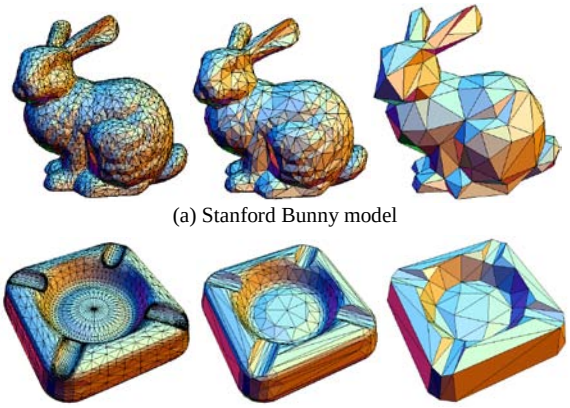


Fig.4. Defining parameter (u,v)-mask for vertices of polygonization mesh of a single patch: (a) polygonization mask (left) and (b) vertex parameter points on polar parameter plane (right)



(a) Stanford Bunny model

(b) ashtray model (http://www.oyonale.com).

Fig. 5. Examples of simplified models created by our modified decimation algorithm

III. EXPERIMENTS AND RESULTS

We have implemented the proposed algorithm using C++ and OpenGL library on PC. In Fig.5, Stanford Bunny and the ashtray models are simplified into simple versions for constructing base meshes. Fig.6 illustrates the boundary curves of patches created from the simplified Bunny base mesh by the computing geodesics method. An example of the boundary condition curves for PDE patch construction is show in Fig.7. Once the PDE patches on the original models are constructed, we can calculate arbitrary iso-parameter curves on the patches to polygonize the surface with subdivision connectivity. In Fig.8, a base mesh and the iso-parameter curves are illustrated. The iso-parameter curves are computed from the PDE patches by Eqn.(17). In Fig.9, three subdivision connectivity models with different resolutions are finally constructed. The experiments illustrate that our method can create connectivity meshes not only from dense mesh models, but also from very simple models, e.g., a simple cube mesh in Fig.9.

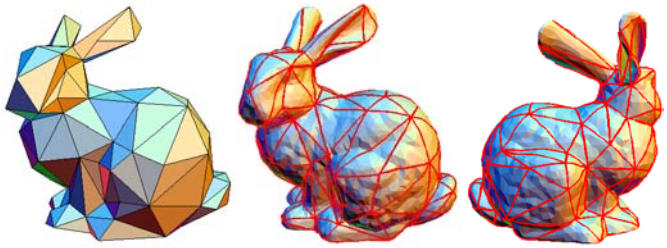


Fig. 6. PDE patches created by calculating geodesics from the simplified Stanford Bunny model with 200 triangles

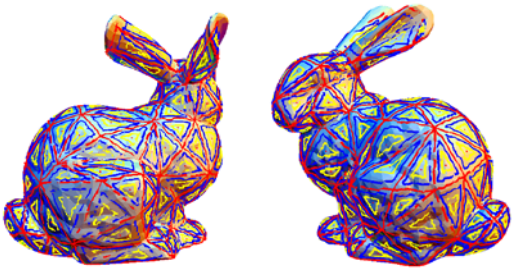


Fig. 7. PDE boundary condition curves of Stanford Bunny model

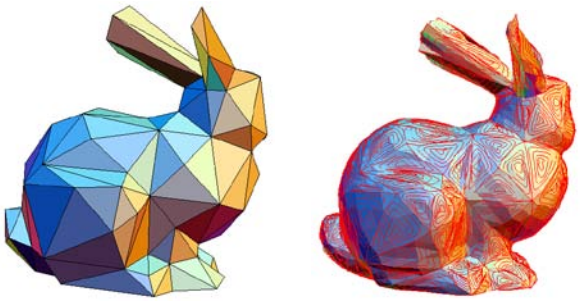


Fig. 8. Iso-parametric curves on PDE.

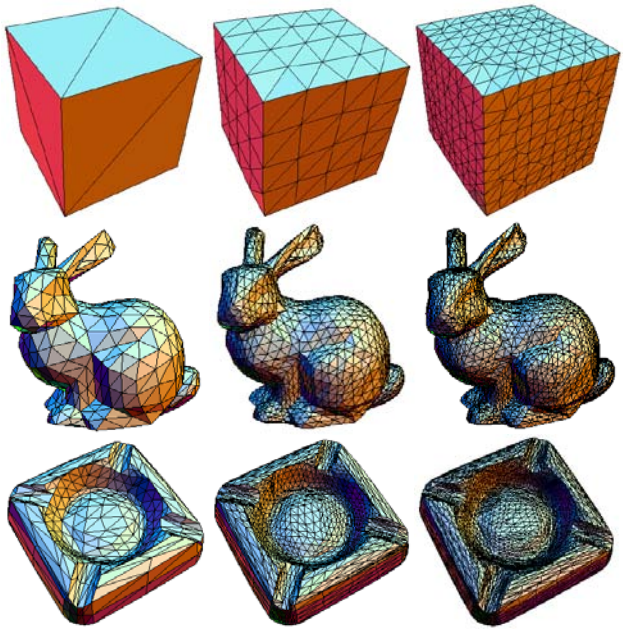


Fig. 9. Multiresolution models created by our algorithm with subdivision connectivity (Color Plate 9)

IV. CONCLUSION

In this paper, we have proposed an algorithm to build a multiresolution mesh model with subdivision connectivity. The new meshes are constructed from their original meshes, by taking piecewise PDE patch approximations as their parameterization. Our method is the combination of several digital geometry processing techniques, such as mesh decimation, surface parameterization, piecewise PDE patch construction, and surface resampling, etc. According to the subdivision level configuration, the method can create different mesh models with various subdivision structure. The experimental results show that that the constructed piecewise PDE patches can provide an efficient parameterization of the local mesh patches on original mesh for subdivision resampling.

ACKNOWLEDGMENT

This project is supported by the Singapore National Research Foundation Interactive Digital Media R&D Program, under research Grant NRF2008IDM-IDM004-002 “Visual and Haptic Rendering in Co-Space”.

REFERENCES

[1] M. Eck, T. Derose, T. Duchamp, et al. Multiresolution analysis of arbitrary meshes. In *Computer Graphics (SIGGRAPH 95 Proceedings)*, 1995, 173-182.

[2] A. Certain, C. J. Popovi, T. Derose, et al. Interactive multiresolution surface viewing. In *Computer Graphics (SIGGRAPH 96 Proceedings)*, 1996, 91-98.

[3] M. Lounsbery, T. Derose, J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. *Transactions on Graphics*, 1997, 16(1): 34-73.

[4] M. Garland, S. P. Heckbert. Surface simplification using quadric error metrics. In *ACM SIGGRAPH Conference Proceeding*, 1997, 209-216.

[5] H. Hoppe. Progressive meshes. In *ACM SIGGRAPH Conference Proceedings*, 1996, 99-108.

[6] W. F. Aaron. A. W. Lee, W. Sweldens, et al. MAPS: multiresolution adaptive parameterization of surfaces, In *ACM SIGGRAPH Conference Proceedings*, 1998, 95-104.
[7] I. Guskov, K. Vidimce, W. Sweldens. Normal meshes. In *ACM SIGGRAPH Conference Proceedings*, 2000, 95-102.
[8] I. Friedel, A. Khodakovsky, P. Schörder. Variational normal meshes. *ACM Transactions on Graphics 2004*, 23(4): 1061-1073.
[9] G. Taubin. Detecting and reconstructing subdivision connectivity. *Visual Computer*, 2001,18(5-6): 357-367.
[10] C. T. Loop. Smooth subdivision surfaces based on triangles. Master Thesis, Dep. of Mathematics, University of Utah, 1987.
[11] P. Schörder, W. Sweldens. Digital geometry processing. Course Notes, *ACM SIGGRAPH*, 2001.
[12] M. Y. Pang, Z. G. Pan, J. Tang, et al. Polygonizing implicit surface with normal subdivision triangulation. *Journal of Computational Information Systems*, 2007, 3(1): 189-195.
[13] M. Floater. Parameterization and smooth approximation of surface triangulations, *Computer Aided Geometric Design*, 1997, 14(3): 231-250.
[14] S. Melia. A simple, fast, effective polygon reduction algorithm, *Game Developer*, 1998, 10: 44-49.
[15] B. Levy. Parameterization and deformation analysis on a manifold. Technical report, Alice, 2007. *http://alice.loria. fr/publications.*
[16] M. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 2003, 20(1): 19-27.
[17] W. Tutte. How to draw a graph. In: *Proc. of the London Mathematical Society*, 1963, 743-768.
[18] K. Hormann, A. Sheffer, B. Levy, et al. Mesh parameterization: theory and practice. *ACM SIGGRAPH 2007 Course Notes*, 2007, 1-122.
[19] G. L. Chen, M. Y. Pang, J. Wang. Calculating shortest path on edge-based data structure of graph, In Proc. *The 2nd International Workshop on Digital Media and its Application in Museum and Heritage*, 2007, 416-421.
[20] J. Chen, Y. Han. Shortest paths on a polyhedron, Part I: Computing shortest paths. *International Journal of Computer Geometry Application*, 1996, 6: 127-144.
[21] B. Kaneva, J. O'Rourke. An implementation of chen & han's shortest paths algorithm, In *Proc. the 21st Canadian Conference on Computer Geometry*, 2000, 139-146.
[22] S. Kapoor. Efficient computation of geodesic shortest paths, In Proc. 31st ACM Symposium on Theory of Computing, 1999, 770-779.
[23] M. Lanthier, A. Maheshwari, J. R. Sack. Approximating weighted shortest paths on polyhedral surfaces, In *Proc. the 13th annual symposium on Computational geometry*, 1997, 274-283.
[24] D. Martínez, L. Velho, P. Carvalho. Geodesic paths on triangular meshes, In *Proc SIBGRAPI/SIACG*, 2004, 210-217.
[25] J. Mitchell. Geometric shortest paths and network optimization, In: J. Sack and J. Urrutia, eds., *Handbook of Computational Geometry*, Elsevier Science, 2000, 633-702.
[26] V. Surazhsky, T. Surazhsky, D. Kirsanov, et al. Fast exact and approximate geodesics on meshes, *ACM Transactions on Graphics*, 2005, 24(3): 553-560.
[27] M. Bloor, M. Wilson. Generating blend surface using partial differential equations, *Computer Aided Design*, 1989, 21(3): 165-171.
[28] Y. Sheng, A. Sourin, G. G. Castro, et al. A PDE method for patchwise approximation of large polygon meshes, *The Visual Computer*, 2010, 26(6-8): 975-984.

Mingyong Pang is an associate professor at the Department of Educational Technology, Nanjing Normal University, Chian. He received his Ph.D. degrees in mechanical design and theory from Jiangsu University in 2004. He ever worked as a postdoc in the State Key Lab of Novel Software Technology, Nanjing University, China. He has published more than 80 peered papers in journal and internationaal conferences. His research interest covers digital geometry processing, visualization of dynamic systems, and computer-aided geometric design.

Alexei Sourin is an associate professor at the School of Computer Engineering, Nanyang Technological University, Singapore. He received his M.Eng. and Ph.D. degrees in computer graphics from the Moscow Engineering Physics Institute, Russia in 1983 and 1988, respectively. His research interests are in function-based shape modeling, shared virtual environments, haptic interaction and rendering, web visualization and visualization on the grid, virtual surgery, scientific visualization, and cyberlearning. He is a co-founder of the function representation (FRep) concept. He has also proposed a function-based web-visualization technique where analytical formulas are used for defining the geometry, appearance and physical properties of the objects in shared virtual scenes.

Dr. Sourin published over 150 referred research papers and was invited to give talks at many scientific events. Dr. Sourin is a senior member of IEEE, a member of ACM SIGGRAPH, and a recipient of many research awards. He is on the editorial boards and serves as a guest editor of several international journals. He was on the program committees of over 70 international conferences. He is a coordinator of the International Conferences on Cyberworlds, for which he was a program chair in 2003–2005 and a general co-chair in 2006, 2007 and 2010. He was also a general co-chair of the 2009 ACM Symposium on Web3D and Computer Graphics International 2010 (CGI 2010). More details are available at *http://www.ntu.edu.sg/home/assourin*.

Zhigeng Pan received his Bachelor's degree and Master's degree from the Computer Science Department, Nanjing University in 1987 and 1990 respectively and Ph.D. degree in 1993 from Zhejiang University. Since 1993, he has been working at the State Key Lab of CAD&CG, Zhejiang University. He has published more than 100 papers on international journals, national journals and international conferences. His research interests include distributed graphics, virtual reality, multimedia, digital entertainment. Prof. Pan is a member of SIGGRAPH, Eurographics, IEEE, a senior member of the China Image and Graphics Association.

He is on the director board of the International Society of VSMM (Virtual System and Multimedia), and a member of IFIP Technical Committee on Entertainment Computing (acting as representative from China). Currently, he is the Editor-in-Chief of The International Journal of Virtual Reality, Co-EiC of LNCS Transactions on Edutainment. He is on the editorial board of International Journal of Image and Graphics, International Journal of CAD/CAM, Journal of Image and Graphics, Journal of CAD/CG etc. He is the director of the Virtual Reality Committee, China Society of Image and Graphics; he is on the steering committee of VRCAI series conference. He is the organizing committee chair of VRAI'2002, ICIG'2004, etc.