Conversion of Polygonal Surfaces to Displaced Subdivision Surfaces

Muhammad Hussain[1,2], Yoshihiro Okada[1,2], and Koichi Niijima[1]
[1]Graduate School of Information Science and Electrical Engineering, Kyushu University,
6-1, Kasuga Koen, Kasuga, Fukuoka 816-8580, Japan.
[2]Intelligent Coooeration and Control, PRESTO, JST
{mhussain, okada, niijima}@i.kyushu-u.ac.jp

ABSTRACT

Displaced subdivision [9] puts forth a number of attractive features for editing, geometry compression, animation, scalability, and adaptive rendering of polygonal models. In this representation, a detailed surface model is defined as a scalar-valued displacement map over a smooth domain surface. The construction of the smooth domain surface from a polygonal model is a challenging task in the conversion process. We propose a new method for defining the smooth domain surface based on $\sqrt{3}$-subdivision scheme and a linear time optimization technique. At some fixed level of detail, the vertex and triangle complexity of the displaced surface generated by the proposed method is far less and the magnitude of the offset values defining the displacement map are smaller and so it results in higher compression ratios and better transmission speed. The proposed algorithm creates surfaces of better quality, is computationally more efficient and occupies less memory as compared to the original algorithm [9]

KEYWORDS
Subdivision, multiresolution geometry, displacement map, geometry compression, irregular connectivity

1 Introduction

Highly detailed and complex surface polygonal models are commonplace. Representing these models as triangle meshes has become a de facto standard where geometry of a model is typically encoded by three scalar values (x, y, z) representing the location of a vertex in 3D space and its connectivity often irregular. Sheer size and irregular connectivity of such meshes put a threat to manipulation, transmission, storage, animation and rendering of surface models.

To overcome these issues, Lee et al. [9] proposed the idea of displaced subdivision surface which represents a surface model as a displacement map over a simple, smooth domain surface. Although it lends a number of advantages over the mesh representation, the conversion process puts forth the challenging problem of finding the underlying smooth domain surface. Lee et al [9] define domain surface using subdivision surfaces because of their capability of defining smooth surfaces of arbitrary topology, a memory intensive decimation approach and time consuming optimization technique. An alternative technique was proposed to define smooth domain surface by Hussain et al.[6] which exploits $\sqrt{3}$ subdivision technique [8], a memory efficient decimation approach and a linear time optimization method. Although it reduces significantly memory overhead and time complexity, the magnitude of scalar offset values defining the displacement map is a little higher because of small wiggles in the domain surface and so the compression ratio is not so good. The proposed technique deals with this issue without increasing the complexity of the conversion process.

The idea of displacing a surface by a function was first introduced by Cook [1]. Guskov et al.[3] present an algorithm for representing a surface as a normal mesh by successively applying a hierarchy of displacements to a mesh as it is subdivided. Their construction encodes most part of the mesh as scalar displacements, a small fraction of vertices need vector displacements to prevent surface folding.

The subsequent paper is arranged as follows. Section 2 briefs $\sqrt{3}$ subdivision surface and the algorithm has been presented in Section 3. Results have been discussed in Section 4. Section 5 concludes the paper.

2 $\sqrt{3}$-Subdivision

Among subdivision schemes that perform on triangular meshes, 4-8 subdivision [12] and $\sqrt{3}$ subdivision [8] are based on 1-to-2 and 1-to-3 splits respectively and as such are slower than Loop scheme [11] in the sense that they increase the number of vertices (and faces) by the factor of 2 and 3 respectively instead of 4. As a consequence, using 4-8 subdivision and $\sqrt{3}$ subdivision schemes, more levels of uniform resolution can be obtained if there is an upper bound on the complexity of the generated mesh. For 4-8 subdivision, the infinite position mask of a vertex is not straight forward, so we have decided to use $\sqrt{3}$ subdivision for defining smooth domain surface.

Topological operation of $\sqrt{3}$-subdivision performs a 1-to-3 split for every triangle by inserting a new vertex at its center and introducing three new edges by connecting it to each of the three vertices of the triangle; the old edges are flipped to re-balance the valence of the mesh vertices. Smoothing operator consists of two rules: one for odd vertices and one for even vertices. For thorough treatment of

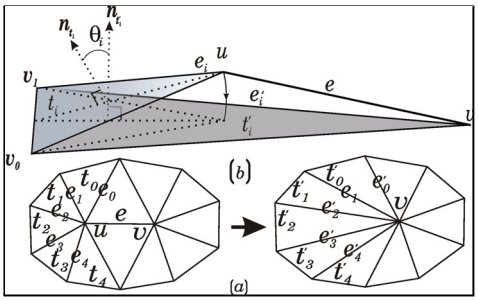

Figure 1: (a) Edge collapse operation (b) Collapse of edge $e = \{u, v\}$ maps the edge e_i onto e_i' and the adjacent triangle t_i onto t_i'.

the scheme consult [8].

3 Description of the Algorithm

The algorithm takes a triangle mesh as an input and outputs the control mesh of the subdivision surface along with displacement field values. The conversion process follows the steps give below:

1. Simplification: The original mesh is simplified to obtain the raw control mesh.
2. Optimization: The vertices of the raw control mesh are optimized so that the domain surface closely fits the original surface.
3. Subdivision and Sampling: The difference between the original surface and the domain surface is sampled as displacement field values.

Figure 3 illustrates the conversion process. In the sequel, the main procedures involved in the conversion process have been elaborated.

3.1 Simplification

A fast, simple and memory efficient error metric proposed in [7] and the half-edge collapse transformations have been exploited for simplifying the triangle mesh for obtaining the raw control mesh. A typical half-edge collapse transformation $u \rightarrow v$, has been shown in Figure 1(a). The vertex u and the two triangles incident on the edge $e = \{u, v\}$ are eliminated and in the remaining triangles incident on u, u is substituted with v. The cost of this transformation is

$$\text{Cost}(u \rightarrow v) = \sum_{e \in E} \theta_e A_e$$

where $E = \{e_0, e_1, e_2, e_3, e_4\}$ and A_{e_i} is the area of the triangle $t_i'' = \{u, v, v_1\}$ described by the edge e_i as it moves when the edge e is collapsed and θ_{e_i} is the angle between the normals of the adjacent triangles t_i and t_i' before and after the transformation, see Figure 1(b). For the sake of efficiency, θ_{e_i} is approximated by $1 - \mathbf{n}_{t_i} \cdot \mathbf{n}_{t_i'}$.

To ensure that the normal space of the original surface is locally preserved, the edge collapse transformation $u \rightarrow v$, which results in significant normal deviation at the vertex v, is not allowed.

3.2 Optimization of the Control Mesh

The raw control mesh obtained in the preceding section is optimized so as the vertex positions of the resulting smooth subdivision domain surface closely fits the original surface. For achieving this objective, a very simple and rigorous technique has been proposed that is based on the $\sqrt{3}$ subdivision rule used for positioning a vertex at subdivision level m, see [8]. In particular for $m = 1$, this rule can be expressed as

$$p^1 = \gamma_n p + (1 - \gamma_n) p^\infty$$

where $\gamma_n = (\frac{2}{3} - \alpha_n)$. But the smoothing rule for even vertices, see [8], yields

$$p^1 = (1 - \alpha_n) p + \tfrac{\alpha_n}{n} \sum_{i=0}^{n-1} p_i.$$

Considering all the r vertices of the raw control mesh and these two equations, we end up with a linear system of equations whose one equation is

$$\frac{1}{3} p + \frac{\alpha_n}{n} \sum_{i=0}^{n-1} p_i = (1 - \gamma_n) p^\infty. \tag{1}$$

where p^∞ is the limit position of p, and p_i $(i = 1, 2, \cdots, n)$ are 1-ring neighbors of p.

Solving this linear system, we can get the optimal position of the vertices, but the problem with this approach is that it results in excessive undulations in the smooth domain surface. To discourage such undulations and improve the quality of the resulting smooth domain surface, we introduce additional degrees of freedom and then set these degrees of freedom by optimizing some energy functional subject to the linear constraints (1). We exploit the energy functional proposed in [4], which is defined as

$$E = \sum_{e \in \mathcal{E}} (De)^2 \tag{2}$$

where $\mathcal{E}$ is the set of all edges of the control mesh, and D_e is the difference between the normals to the triangles incident on the edge e and is expressed as

$$D_e = \omega_e^i p_i + \omega_e^j p_j + \omega_e^k p_k + \omega_e^l p_l$$

where $\{p_i, p_j, p_k, p_l\}$ are vertices of the triangles incident on the edge e, $\omega_e^i = \frac{l_e}{\Delta_{kji}}$, $\omega_e^j = \frac{l_e}{\Delta_{lij}}$, $\omega_e^k = -\frac{l_e \Delta_{jlk}}{\Delta_{kji} \Delta_{lij}}$, $\omega_e^l = -\frac{l_e \Delta_{ikl}}{\Delta_{kji} \Delta_{lij}}$, Δ_{ijk} is the signed area of the triangle (i, j, k) and l_e is the length of the edge e. These areas and

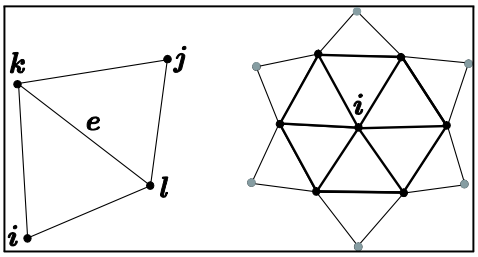

Figure 2: (Left) Stencil of edge e, (e) is the set of vertices in the stencil. (Right) The sets of vertices $V(i)$ and $VF(i)$ in 1-ring neighborhood (dark shaded) and in flapped 1-ring neighborhood (dark and light shaded without p_i), respectively, of the vertex p_i, and $\mathcal{E}(i)$ is the set of bold edges.

length are computed after applying *hinge map* on the stencil of the edge e (see Figure 2) taking e as hinge and rotating one of the triangles so that it lies in the plane of the other triangle.

We choose this energy functional because it not only discourages excessive undulations in the smooth domain surface, but also smooths and preserves locally the normal space of the control mesh, and so results in offset values of smaller sizes.

Now for optimal position of the vertices, the problem is to optimize the energy functional (2) subject to the linear constraints (1). Employing the method of Lagrange multipliers, this problem is equivalent to the solution of the following system of linear equations

$$\eta_i p_i + \sum_{j \in VF(i)} \delta_{ij} p_j + \sum_{k \in V(i)} \frac{\alpha_{n_k}}{n_k} \lambda_k + \frac{1}{3}\lambda_i = 0 \quad (3)$$

$$\frac{1}{3} p_i + \frac{\alpha_{n_i}}{n_i} \sum_{l \in V(i)} p_l - (1 - \gamma_{n_i}) p_i^{\infty} = 0 \quad (4)$$

where $\eta_i = \sum_{e \in \mathcal{E}(i)} (\omega_e^i)^2$, $\delta_{ij} = \sum_{\{e \in \mathcal{E}(i) | j \in V(e)\}} \omega_e^i \omega_e^j$, for $\mathcal{E}(i)$ and $V(e)$ see Figure 2, $VF(i)$ and $V(i)$ are the sets of vertices in the flapped 1-ring neighborhood and 1-ring neighborhood of the vertex p_i (see Figure 2) and (λ_1, λ_2, $\lambda_3, \cdots, \lambda r$) is the vector of Lagrange multipliers, r being the number of vertices in the control mesh.

Equations (3) and (4) are rows of a large linear system of order $2r$. Since the number of non-zero entries in each row is much smaller than $2r$, so the system is a sparse linear system. So the problem of optimizing the vertex positions decomposes into the problem of solving three independent sparse linear systems, one for each of the three coordinates of the vertex positions. We solve each of the system using biconjugate gradient method [13] which yields an acceptable solution in O(r) time.

The goal is to optimize the position of each vertex on $\tilde{M}^0$ such that its limit position is on the original surface M when $\tilde{M}^0$ is subdivided up to level k, and the vertices of $\tilde{M}^0$ form a subset of the vertices of M because $\tilde{M}^0$ is generated by applying half-edge collapse transformations, so each of p_i^{∞} in the above system is treated as a vertex on $\tilde{M}^0$ and the system is solved for the optimal positions of p_i's on the optimized control mesh M^0.

3.3 Subdivision and Sampling

Optimized control mesh M^0, obtained exploiting the techniques presented in the previous subsections, is refined up to k subdivision levels using refinement operators of $\sqrt{3}$-subdivision scheme. Then, each vertex of the resulting surface $\tilde{M}^k$is pushed to its limit position using infinite position operator of the scheme for getting the smooth surface M^k, which serves as the domain for representing the given surface polygonal model as displacement map. The normal at each vertex is computed and to capture the details of the original surface model as a scalar displacement map defined over the smooth domain surface M^k, for each vertex of M^k, its signed distance along its normal from the original polygonal model M is computed. The position of a vertex on M^k and its normal define a straight line; the length of the segment of this line between the vertex and the original surface is the measure of the offset value, which is positive if intersection occurs in the direction of the outward normal otherwise it will be negative. This line may have multiple intersections or the original surface may be oriented in the wrong direction with respect to this line. If the directed line is intersected at more than one points, then we pick the one closest to the domain surface. In the second case we reject the intersection. To compute the intersections efficiently, we store the original triangle mesh in an OBB tree data structure [2].

4 Results and Discussion

Using our implementation of the proposed algorithm described in the preceding sections, we converted many closed triangle meshes, available in the public domain, into displaced $\sqrt{3}$-subdivision model representation and found encouraging results. Due to space constraints, we present here the conversion results of only one model in Figures 3, after defining optimized control mesh, it is refined up to subdivision level 4 and then the vertices of the resulted mesh are pushed to their infinite positions by applying infinite position mask. The total time taken by this conversion process on 550 MHz Pentium III PC is 78.25 sec. which is far less than that taken by the method of Lee et al. on PC of similar configuration.

5 Conclusion

A new algorithm has been presented for defining smooth domain surface for displaced $\sqrt{3}$-subdivision surfaces, which is not only fast and memory efficient but generates

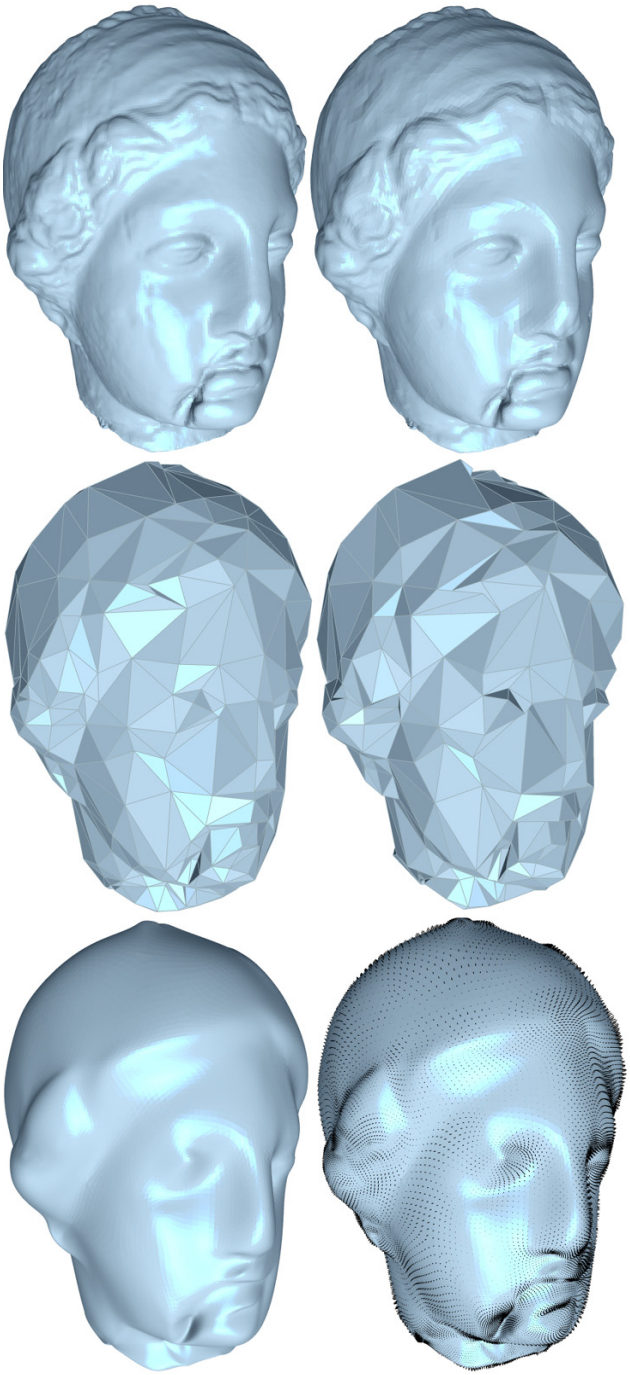

Figure 3: (Top row: left) Original Venus model M (# faces 268,686; size on disk 9.67 MB) is read in and the algorithm creates (middle row: left) raw control mesh $\tilde{M}^0$ (#faces 796), then it is optimized to get (middle row: right) control mesh M^0. Finally, M^0 is subdivided to get (bottom row: left) smooth domain surface M^k, $k = 4$. The difference of M and M^k is encoded as (bottom row: right) displacement field. (Top row: right) Displaced $\sqrt{3}$ subdivision surface (#faces 64476, size on disk 460KB (24KB for control mesh and 336KB for displacement map)).

displaced surfaces of good quality having higher compression ratios. As our main contribution is the efficient method for the construction of smooth domain surface, so the displaced subdivision surfaces generated by our algorithm offer all those benefits as have been demonstrated by Lee et al., i.e. compression, editing, animation, and scalability. The only limitation of this algorithm is that it converts only closed polygonal surface models.

References

[1] R. Cook. Shade trees. In *Computer Graphics* (*Proc. SIGGRAPH'84*), pages 223-231, 1984.

[2] S. Gottschalk, M. Lin and D. Manocha. OBB-tree: A hierarchical structure for rapid interference detection. In *Computer Graphics* (*Proc. SIGGRAPH'96*), pages 171-180, 1996.

[3] I. Guskov, K. Vidimce, W. Sweldens and P. Schroder. Normal meshes. In *Computer Graphics* (*Proc. SIGGRAPH'00*), pages 95-102, 2000.

[4] I. Guskov, W. Sweldens, and P. Schroeder. Multiresolution signal processing for meshes. In *Computer Graphics (Proc. SIGGRAPH 99*), pp. 325-334.

[5] Hussain, M., Okada, Y. and Niijima, K. : Fast, simple and memory efficient mesh simplification. In *Proc. CGIM'01*, pages 72-77, USA, August 2001.

[6] Hussain, M., Okada, Y. and Niijima, K. Displaced Subdivision Meshes. In *Proc. MSO*03, pp. 214-219, Banff, CANADA, June 2003

[7] Hussain, M., Okada, Y. and Niijima, K. Fast, simple, feature-preserving and memory efficient simplification of triangle meshes. International Journal of Image and Graphics, 3(4):1-18, 2003.

[8] L. Kobbelt. $\sqrt{3}$-Subdivision. In *Computer Graphics* (*Proc. SIGGRAPH '00*), pages 103-112, August 2000.

[9] A. Lee, H. Moreton, and H. Hoppe. Displaced subdivision surfaces. In *Computer Graphics* (*Proc. SIGGRAPH '00*), pages 85-94, August 2000.

[10] A. Lee, W. Sweldens, P. Schroder, L. Cowsar and D. Dobkin. MAPS: Multiresolution adaptive parameterization of surfaces. In *Computer Graphics* (*Proc. SIGGRAPH '98*), pages 95-104, 1998.

[11] C. Loop. *Smooth subdivision surfaces based on triangles*. Master's thesis, University of Utah, Department of Mathematics, 1987.

[12] Luiz Velho,and Denis Zorin. 4-8 Subdivision. *CAGD*, volume 18, Issue 5, Pages 397-427.

[13] William H., Saul A., William T., and Brian P. *Numerical recipes in* C++, Cambridge university press.