Journal of Computer Science 5 (4): 242-249, 2009
ISSN 1549-3636

Representation of Polygonal Surfaces as Displaced Subdivision Surfaces

Muhammad Hussain
Department of Computer Science, King Saud University, Saudi Arabia

Abstract: Problem statement: Displaced subdivision representation possesses a number of attractive features for efficient and convenient processing tasks like editing, geometry compression, animation, scalability and adaptive rendering of polygonal models. In this representation, a detailed surface model was built as a scalar-valued displacement map over a smooth domain surface. The construction of the smooth domain surface from a polygonal model was a challenging task in the conversion process. **Approach:** For building the smooth domain surface, we proposed an efficient algorithm that was based on √3-subdivision scheme, memory efficient simplification and a linear time optimization technique. **Results:** At some fixed level of detail, the vertex and triangle complexity of the displaced surface generated by the proposed algorithm was far less and so it resulted in better compression ratios and transmission speed. **Conclusion:** The proposed algorithm created surfaces of better quality, computationally more efficient and occupied less memory as compared to the original algorithm by Lee.

Key words: Polygonal models, subdivision surfaces, displacement map, geometry compression

INTRODUCTION

Recent advances in scanning technologies, CAD/CAD systems and computer vision techniques have made it possible to acquire 3D (three dimensional) information that is widely represented in the form of polygonal surfaces, which are, now-a-days, widespread in various application areas of Computer Graphics (CG). A polygonal surface is typically represented as a triangular mesh where geometry is encoded by three scalar values (x, y, z) per vertex and the connectivity of vertices is often irregular. As a result of recent developments in technology, the sizes of triangular meshes are growing rapidly. Sheer sizes and irregular connectivity of such meshes put a threat to manipulation, transmission, storage, animation, rendering and visualization of 3D information. Alternatively, a polygonal surface can be represented as a displaced subdivision surface, which consists of a control mesh and a set of scalar offsets that define the polygonal surface as a scalar displacement map over the smooth domain surface generated from the control mesh by subdivision. This surface representation offers a number of advantages over the mesh representation[15]. Expressing terrain data as a height field over a plane is a simple example of displaced surface. Generalization of this concept to arbitrary surfaces involves the challenging problem of defining efficiently the underlying smooth domain surface that locally closely fits the geometry of the given polygonal surface.

Lee *et al*.[13] define domain surface using Loop subdivision scheme[15]. They employ a very slow and memory consuming heuristic approach to simplify the mesh followed by a time consuming energy minimization technique to optimize the vertex positions so as to closely fit the original surface. An alternative technique was proposed by Hussain *et al*.[10] to define smooth domain surface, which exploits √3-subdivision technique[11], a memory efficient decimation approach and a linear time optimization method. Although it reduces significantly memory overhead and time complexity, the underlying simplification approach for creating raw control mesh overestimates the geometric error and the optimization technique for optimizing vertex positions results in creating small wiggles in the smooth domain surface, which causes scalar offsets of greater magnitudes and so the compression ratio is not so good. The proposed technique deals with this issue without noticeable increase in the complexity of the conversion process. The main differentiating features of the proposed algorithm are:

- Reduced memory overhead
- Computational efficiency
- Better quality of generated surfaces and better compression ratio

Related work: Semi-regular or subdivision connectivity meshes offer many advantages, like simple data structures and efficient processing for their manipulation, over the irregular setting where the

connectivity of the vertices of a polygonal model is not regular. Similar to the algorithms by Eck *et al*.[4], Krischnamurthy and Levoy[12] and Lee *et al*.[13], the proposed algorithm constructs a semi-regular mesh from an irregular connectivity input mesh.

The idea of representing a surface as a displacement function was first introduced by Cook[1]. Schroder *et al*.[17] used this idea to represent functions of sphere. Lee *et al*.[13] exploiting the idea of displacement map, developed displaced subdivision surface representation for polygonal surfaces, which are usually encoded by polygonal meshes and validated its usefulness for efficient manipulation, storage, editing and transmission of polygonal surfaces in various application areas. The conversion process proposed by Lee *et al*.[13,14] is not only computationally complex but also involves high memory overhead. They employ a time-consuming and memory extensive heuristic approach based on quadric error metrics[5] and half-edge collapse for decimating the original polygonal surface to generate initial control mesh. Furthermore, their method for optimizing the vertex positions being based on sampling a dense set of points from the original mesh and minimizing their squared distances to the subdivision surface is also computationally very complex.

Similarly to Lee *et al*.[13,14] and Guskov *et al*.[7] presented an algorithm for representing a polygonal surface as a normal mesh by applying successively a hierarchy of displacement maps to the underlying control mesh as it is subdivided. Though their construction encodes most part of the geometry of a polygonal model as scalar offsets, however a small fraction of vertices needs vector displacements to prevent surface folding.

Hussain *et al*.[10] proposed a conversion method for transforming polygonal mesh representation into displaced subdivision surface; this method differs from the technique proposed by Lee *et al*.[13] in the construction of smooth domain surface. It exploits a simple and efficient heuristic based simplification method and linear time optimization approach to push the vertices of the raw control mesh. Though it drastically reduces computational and memory overhead, it creates small wiggles in the smooth domain surface and thus results in scalar offsets of larger magnitudes and so the compression ratio is not good. In this study, an alternative technique has been proposed, which overcomes this problem.

Won and Chang[19] presented an algorithm that directly reconstructs displaced subdivision surface from unorganized points. This method is fairly efficient but it is limited only to a small class of surfaces having simple topology. Also the representation of the surface as control mesh together with displacement map is not straight forward.

Subdivision surfaces: A subdivision surface is generated by iteratively refining a coarse control mesh driven by a refinement operator. In each iteration, refinement operator not only refines the topology of the underlying mesh but also smoothes the geometry and as such it can be assumed to be composed of two types of operators: topological or split operator and geometry smoothing operator. Split operator performs refinement by introducing new vertices and thus new faces in the mesh. Smoothing operator smoothes the vertex positions by taking averages of neighboring vertex positions motivated by some smoothness criteria. The newly inserted vertices are called odd vertices, while the old vertices are referred to as even vertices. A subdivision scheme that relocates only the odd vertices is known as interpolating scheme, e.g., butterfly scheme[3], whereas the one that relocates not only odd vertices but even vertices as well is termed as approximating scheme. The Loop subdivision[15], 4-8 subdivision[16] and $\sqrt{3}$-subdivision[11] are examples of approximating schemes that perform on triangle meshes; in this case the subdivision surface does not go through the control mesh but approximates it.

In Loop subdivision scheme, refinement is performed uniformly by a 1-4 split operator and so the number of faces increases by the factor of 4. Contrary to this, 4-8 subdivision and $\sqrt{3}$-subdivision schemes are based on 1-2 and 1-3 splits respectively and the faces grow in number by the factors of 2 and 3, respectively, i.e., the process of increasing the number of faces is slower than that in Loop scheme; it follows that employing 4-8 subdivision and $\sqrt{3}$-subdivision, one can have more levels of uniform resolution if a prescribed target face complexity of the generated mesh must not exceed a certain number. This reason leads us to exploit 4-8 subdivision or $\sqrt{3}$-subdivision. Because for 4-8 subdivision, the position mask of infinite position of a vertex is not straight forward, so we employ $\sqrt{3}$-subdivision as a tool to define smooth domain surface for displaced subdivision representation and in the following paragraphs, we give an overview of this scheme, for detailed discussion and analysis of the scheme, please consult[11].

For arbitrary control meshes, $\sqrt{3}$-subdivision yields the limit surface to be C^2 almost everywhere except for the extraordinary vertices (valence $\neq 6$) where the smoothness is at least C^1. This scheme has stencils of minimum size and maximum symmetry[11].

In √3-subdivision scheme, split operator divides each triangle at level l into three new triangles by inserting a new vertex at its center, introducing three new edges that connect the new vertex to each of its three vertices and re-balancing the valence of even vertices by flipping every old edge that connects two even vertices. In the following discussion, p^l means the position of a vertex at level l.

Smoothing operator consists of two rules: one for odd vertices and one for even vertices. The smoothing rule for odd vertices is[11]:

$$p^{l+1} = \frac{1}{3}(p_i^l + p_j^l + p_k^l) \quad (1)$$

i.e., an odd vertex p^{l+1} is simply inserted at the center of the triangle $\Delta(p_i^l, p_j^l, p_k^l)$ at level l. Let $p_.^l$ be an even vertex with valence n and $p_0^l, p_1^l, p_2^l, \ldots, p_{n-1}^l$ being its 1-ring neighbors, its position p^{l+1}at level l+1 is determined by the following rule[11]:

$$p^{l+1} = (1-\alpha_n)p^l + \frac{\alpha_n}{n}\sum_{i=0}^{n-1} p_i^l \quad (2)$$

where, $\alpha_n = \frac{4-2\cos(2\pi/n)}{9}$. The limit position of a vertex p^l means the position that it attains as $l\to\infty$. The limit position of any vertex p^l can be determined directly using the following limit position rule[11]:

$$p^{\infty} = (1-\beta_n)p^l + \frac{\beta_n}{n}\sum_{i=0}^{n-1} p_i^l \quad (3)$$

where, $\beta = \frac{3\alpha}{1+3\alpha}$. The position of a vertex p^l can be determined at subdivision level m where m>l exploiting the following rule[11]:

$$p^m = \gamma_n^m p^l + (1-\gamma_n^m)p^{\infty} \quad (4)$$

where, $\gamma_n^m = (\frac{2}{3} - \alpha_n)^m$. This rule is very useful and will be used in our conversion process to build the coarse control mesh for displaced subdivision representation.

With the help of the technique described in[18], the tangent masks for $\sqrt{3}$ -subdivision can be calculated to be $(c_0, c_1, c_2, \cdots, c_{n-1})$ and $(c_{n-1}, c_0, c_1, \cdots, c_{n-2})$, where $c_i = \cos(2\pi i/n)$

MATERIALS AND METHODS

Now, we elaborate the main techniques used by the proposed conversion process.

Creating initial control mesh: The first step is to build initial control mesh $\tilde{M}^0$; it must be built with the goal that its normal space locally closely approximates the normal space of the given polygonal surface because it ensures that the resulting subdivision surface locally closely fits the given surface. For this purpose, we use a fast, simple, memory efficient and feature preserving error metric that automatically preserves locally the normal space of the original polygonal model[9] and a sequence of half-edge collapse transformations to simplify the triangle mesh. Here we give a brief description of the algorithm, for detailed account of the algorithm please consult[9]. The cost of an arbitrary half-edge collapse transformation $\bar{e}_{st}: v_s \to v_t$:

$$\text{Cost}(\bar{e}_{st}) = \sum_{t \in T_{v_s} - T_{e_{st}}} (\Delta_t - \vec{n}_t.\hat{n}_{ot})$$

Where:

Δ_t = The area of the triangle t

$\vec{n}_t$, $\hat{n}_{ot}$ = The current normal (not unit normal) and the original unit normal to the triangle t

T_{v_s} = The set of all triangles incident on vertex v_s

$T_{e_{st}}$ = The set of triangles which share the edge e_{st}, (Fig. 1)

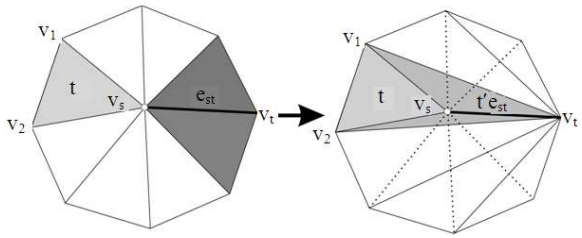

Fig. 1: Edge collapse operation $v_s \to v_t$ eliminates the triangles $T_{e_{st}}$ incident on e_{st} (shown in dark gray) and changes the orientation of the remaining triangles $T_{v_s} - T_{e_{st}}$. Collapse of the edge e_{st} maps typically the triangle $t = \{v_s, v_1, v_2\} \in T_{v_s} - T_{est}$ onto the triangle $t' = \{v_t, v_1, v_2\}$; for computing the distortion caused by t the deviation of the normal to t' (which is current normal to t) is calculated with respect to the original normal to t

Half-edge collapse transformations are applied iteratively in a greedy manner until the error is below a threshold value; the greedy process is guided by the measure of geometric error described above. There is no need to employ any heuristic to ensure the local preservation of the normal space of the original model during the decimation process because the error metric described above automatically does this job, for more detail, please consult[9]. The normal at each vertex is approximated as the average of the normal vectors to the faces in T_{v_s}. Original normal of each vertex is computed using the original polygonal model at a preprocessing stage and is stored for further processing. The normal vectors can also be computed using tangent masks for √3-subdivision scheme but we found that it does not make any difference.

Optimization of the initial control mesh: Subdivision of the initial control mesh $\tilde{M}^0$ causes the generated subdivision surface to shrink because √3-subdivision is an approximating scheme, for detail consult[10]. The smooth subdivision surface locally closely fits the original surface only when it passes through the vertices of the initial control mesh because these vertices lie on the original surface. To force the subdivision surface to pass through these vertices, there is a need to readjust the positions of these vertices; this objective is accomplished by exploiting a kind of optimization technique; in the sequel we present the detail of this technique.

The optimization technique for readjusting the vertex positions of the initial control mesh is based on √3-subdivision rule described by Eq. 4. This rule, in particular, for m = 1, l = 0 and $p^0 = p$ can be expressed as:

$$p^1 = \gamma_n^1 p + (1-\gamma_n^1)p^\infty \tag{5}$$

But according to the smoothing rule for even vertices specified by Eq. 2:

$$p^1 = (1-\alpha_n)p + \frac{\alpha_n}{n}\sum_{i=0}^{n-1} p_i \tag{6}$$

From (5) and (6), we obtain the following equation:

$$\frac{1}{3}p + \frac{\alpha_n}{n}\sum_{i=0}^{n-1} p_i = (1-\gamma_n)p^\infty \tag{7}$$

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

For each $p_i \in \tilde{M}^0$, there exists an equation similar to (7) and so we get a large sparse system of order r, where r is the number of vertices in the control mesh. Solving this system for p_i′s, we get their positions which cause the subdivision surface to pass through p_i^∞'s. The subdivision surface must interpolate the vertices of $\tilde{M}^0$ so that it can locally closely fit the given polygonal surface because the vertices of $\tilde{M}^0$ are on the polygonal surface as it has been created using half-edge collapses. So in the above mentioned system of linear equations, p_i^∞'s are taken to be the vertices of $\tilde{M}^0$ and the solution of the system yields their optimal positions p_i′s.

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 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 (8). We exploit the energy functional proposed in[8], which is defined as:

$$E = \sum_{e \in E} D_e^{\,2} \tag{8}$$

Where:

E = The set of all edges of the control mesh

D_e = The difference between the normal vectors to the triangular faces incident on the edge e and can be 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:

$\{p_i, p_j, p_k, p_l\}$ = Vertices of the triangles incident on 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} = The signed area of the triangle (I, j, k)

l_e = The length of the edge e

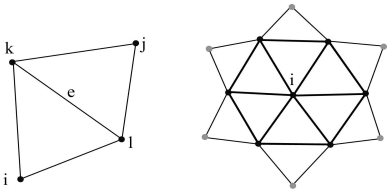

Fig. 2: (Left) Stencil of edge e, V(e) = {p_i, p_j , p_k, p_l} is the set of vertices in the stencil. (Right) The sets of vertices V(i) and VF(i) in 1-ring neighborhood (dark shaded) and in flapped 1-ring neighborhood (dark and light shaded without p_i), respectively, of the vertex p_i and E(i) is the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

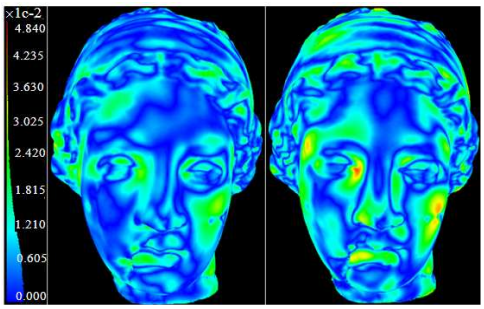

Fig. 3: Error distribution showing the difference between the original model and the smooth domain surface created by proposed method (left) and the one (right) published in[10]. Left of the figure shows error ramp, error increases from zero (blue) to maximum (red)

The areas and the length are computed after applying hinge map on the stencil of the edge e (Fig. 2) taking e as hinge and rotating one of the triangular faces so that it lies in the plane of the other face. We choose this energy functional because it not only discourages excessive undulations in the smooth domain surface, but also smoothes and preserves locally the normal space of the control mesh and so it results in better compression ratio.

Now the problem of optimizing the vertex positions of $\tilde{M}^0$ is equivalent to the problem of optimizing the energy functional (8) subject to the linear constraints (7). Employing the method of Lagrange multipliers, this problem is equivalent to the solution of the following system of linear equations:

$$\eta_i p_i + \sum_{j\in VF(i)} \delta_{ij} p_j + \sum_{k\in V(i)} \frac{\alpha_{n_k}}{n_k}\lambda_k + \frac{1}{3}\lambda_i = 0 \qquad (9)$$

$$\frac{1}{3}p_i + \frac{\alpha_{n_i}}{n_i}\sum_{l\in V(i)} p_l - (1-\gamma_{n_i})p_i^{\infty} = 0 \qquad (10)$$

where, $\eta_i = \sum_{e\in E(i)} (\omega_e^i)^2$, $\delta_{ij} = \sum_{\{e\in E(i)|j\in V(e)\}} \omega_e^i \omega_e^j$, for the interpretation of E(i) and V(e) Fig. 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 (Fig. 2) and (λ_1, λ_2, λ_3, … , λ_r) is the vector of Lagrange multipliers, r being the number of vertices in $\tilde{M}^0$. Eq. 9 and 10 are two rows of a large linear system of order 2r. Since the number of nonzero entries in each row is much smaller than r, so the system is a sparse linear system. In view of this, the problem of optimizing the vertex positions decomposes into the solutions of three independent sparse linear systems, one for each of x-, y- and z-coordinates. We solve each of the system using biconjugate gradient method[20], which yields an acceptable solution in O(r) time.

Polygonal surface as a displacement map: Optimized control mesh M^o is subdivided up to level k using refinement operators of √3-subdivision. Then, each vertex of the resulting surface M^k is pushed to its limit position using limit position rule of √3-subdivision to obtain the smooth surface M^k, which serves as a domain for representing the given polygonal surface model as a displacement map. The normal at each vertex is computed. The position of each vertex p_i on M^k and its normal n_{pi} define the straight line $p(t) = p_i + tn_{pi}$; the length of the line segment between the vertex p_i and p_i' (the vertex where the line pierces the original surface) is the magnitude of the scalar offset, 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 point, then we pick the one that is closest to the domain surface. In the second case we reject the intersection. To compute the intersections efficiently, we make use of OBB tree data structure[6].

RESULTS

Now, we describe the details of the algorithm that converts polygonal surface into the corresponding displaced subdivision surface representation (DSR). Following is the detail of the algorithm:

Conversion_to_DSR()
Input: Polygonal surface model M (V,F)
Output: Displaced subdivision surface DSS = (M^0, D^k), where M^0 is the optimized control mesh and D^k offset values at subdivision level k

$\tilde{M}^0 \leftarrow$ Create_Initial_Control_Mesh (M)
$M^o \leftarrow$ Optimize_Initial_Control_Mesh (M, $\tilde{M}^0$)
$M^k \leftarrow$ Subdivide_Optimized_Control_Mesh (M^o, k)
$D^k \leftarrow$ Compute_Offset_Values (M, M^k)

The optimized control mesh M^0 and the offset values D^k form the displaced subdivision surface representation (DSR):

$$DSR = S^{\infty}S^kM^0 + D^k$$

where, S^{∞} and S^k are matrix forms of limit position rule and refinement operators, respectively, of √3-subdivision.

Figure 4 demonstrates the different stages of the algorithm and the output of the algorithm for Venus model. Figure 5 shows the conversion of horse model.

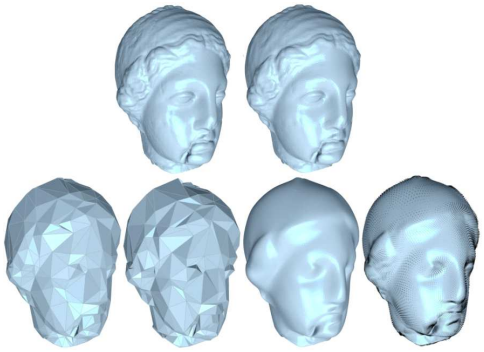

Fig. 4: (Top row: left) Original Venus model M (# faces 268,686; size on disk 9.67 MB) and (Top row: Right) Displaced subdivision surface (#faces 64476, size on disk 460-24 KB for control mesh and 336 KB for displacement map). Bottom row demonstrates the four phases of the conversion process: (bottom row: Left) raw control mesh $\tilde{M}^0$ (#faces 796), (bottom row: Middle left) optimized control mesh M^0, (bottom row: Middle right) smooth domain surface S^k, k = 4 obtained by subdividing M^0 and (bottom row: right) displacement field that encodes the difference of M and S^k

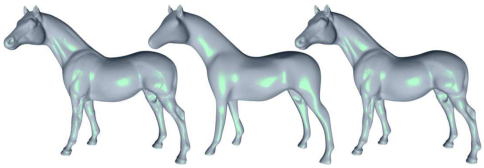

Fig. 5: Horse model (courtesy cyber-ware): (Left) Original T: 96966, (middle) smooth domain surface, S^k = 4, T: 64476, (right) displaced subdivision surface

DISCUSSION

Our implementation of the proposed algorithm has been tested on many public domain polygonal surface models. Here, we discuss the performance of the algorithm using only two typical benchmark models.

Compression ratio: The √3-subdivision technique increases the number of vertices by a factor of 2, whereas Loop subdivision increases this number by a factor of 3 after each subdivision step. Lee *et al.*[13] use Loop subdivision. So the size of the set of offset values- D^k-produced by the proposed method is less by 33% as compared to the one generated by the method of Lee *et al.*[13,14] Also, it is obvious from error distribution shown in Fig. 3 that most of the offset values are close to zero and the range of these values is small, so the proposed method results in higher compression ratios.

Quality of the generated surfaces: For objective evaluation of the displaced subdivision surfaces generated by the proposed algorithm, we use mean square geometric error -L^2- and compute it using well-known IEI-CNR metro tool[2]. Column 5 of Table 1 shows L^2 as the percentage of the diagonal of the bounding box for various models. It is apparent that the surfaces generated by the proposed method compare well with the original polygonal surfaces. Because the implementation of the algorithm by Lee *et al.*[13] is not available in public domain, we cannot make comparison directly with it in terms of quality. Anyhow, according to the error statistics reported in[13] (Table 1), L^2 error is 0.027 for Venus model with control mesh consisting of 748 faces and displaced subdivision surface having 191488 faces, whereas this error in our case is 0.011 (Table 1) in spite of the displaced subdivision surface being smaller in size. Though this is not exact comparison, it gives the idea that quality of the displaced subdivision surfaces generated by the proposed method is better. It is obvious from Fig. 3 that the proposed method performs better even than the one proposed in[10] in terms of the quality.

Table 1: Sizes of the original model, corresponding Displaced Mesh (DM) and control mesh (CM) are given as the number of triangle faces

Model	Size of M (# faces)	Size of S^k (# faces)	Size of M^0 (# faces)	L^2 (%)
Horse	96966	64476	796	0.032
Rabbit	134074	64476	796	0.015
Venus	268686	64476	796	0.011
Ball joint	274120	85536	1056	0.015

Table 2: Execution times (sec)

Process	Horse	Rabbit	Venus	Ball joint
Simplification	16.58	23.20	49.83	51.10
Optimization	0.41	0.45	0.55	1.02
Sampling	14.05	14.88	27.89	28.28
Total (sec)	31.05	38.53	78.27	80.40

Table 3: Execution times (min) taken from[13]

Process	Armadillo	Venus	Bunny	Dinosaur
Size (#F)	210,944	100,000	69,451	342,138
Simplification	61	28	19	115
Optimization	25	11	11	43
Sampling	2	2	1	5
Total (min)	88	41	31	163

Time complexity: For creating initial control mesh, the method proposed by Lee *et al*.[13] involves computing the parameterization of the original polygonal surface using MAPS[14] during the simplification process and a large number of comparisons to ensure that the normal space of the original polygonal surface is locally preserved. And they optimize the initial control mesh by solving a nonlinear optimization problem. In comparison, our approach doesn't involve any constraint to insure local preservation of normal space while simplification and our optimization technique is based on the linear problem of solving three sparse linear systems. This analysis shows that the running time of the proposed algorithm is far more less than that by Lee *et al*.[13]. The empirical results shown in Table 2 and 3 demonstrate this fact. The execution times of Table 2 have been reported on a 550 MHz pentium III machine. The execution times of the original displaced subdivision surface scheme, shown in Table 3 (taken from[13]), also have been obtained on 550 MHz pentium III PC. Although in our experiments, we have used different models and a machine that might have different architecture, even then we can have an overall idea about the running efficiency.

Space complexity: Apart from the memory required for the optimization process, the method proposed in[13] for building initial control mesh needs 64 r bytes (assuming that floats are used), where r is the number of vertices in M, for simplification and parameterization procedures in addition to necessary space requirements for storing the geometry and connectivity of M. The proposed technique requires only 24 r bytes, so it significantly cuts off the memory overhead. Note that the quadric error metric needs 40 bytes of memory per vertex and the parameterization requires 4 bytes per vertex and 8 bytes per vertex are needed for management of priority queue. The error metric which we employed consumes only 4 bytes per vertex and 12 bytes are utilized to store the original normal at each vertex; the simplification approach adapted in this proposal stores original normal for the calculation of error metric whereas it is stored by the method in[10] for heuristic used for locally preserving the normal space. Note that the memory requirements of the proposed scheme are almost the same as those of the method proposed in[10].

CONCLUSION

An efficient technique has been presented for building smooth domain surface for displaced subdivision surface representation, which is not only fast and memory efficient but also generates displaced surfaces of better quality and results in higher compression ratio. As our main contribution is an efficient method for the construction of smooth domain surface, so the displaced subdivision surfaces generated by the proposed method offer all those benefits as have been demonstrated by Lee *et al*.[13], i.e., compression, editing, animation and scalability. The only limitation of this algorithm is that at the moment it is applicable only to closed polygonal surface models.

REFERENCES

1. Cook, R., 1984. Shade trees. ACM. SIGGRAPH. Comput. Graph., 18: 223-231. http://portal.acm.org/citation.cfm?id=964965.808602
2. Cignoni, P., C. Rocchini and R. Scopigno, 1998. Metro: Measuring error on simplified surfaces. Comput. Graph. Forum, 17: 167-174. DOI: 10.1111/1467-8659.00236
3. Dyn, N., D. Levin and J.A. Gregory, 1990. A butterfly subdivision scheme for surface interpolation with tension control. ACM. Trans. Graph., 9: 160-169. http://portal.acm.org/citation.cfm?id=78956.78958
4. Eck , M., T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery and W. Stuetzle, 1995. Multiresolution analysis of arbitrary meshes. Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, (ACCGIT' 95), ACM Press, New York, USA., pp: 173-182. http://portal.acm.org/citation.cfm?id=218440
5. Garland, M. and P. Heckbert, 1997. Surface simplification using quadric error metrics. Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, (ACCGIT' 95), ACM Press/Addison-Wesley Publishing Co., New York, USA., pp: 209-216. http://portal.acm.org/citation.cfm?id=258849

6. Gottschalk, S., M. Lin and D. Manocha, 1996. OBB-tree: A hierarchical structure for rapid interference detection. Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, (ACCGIT' 96), ACM Press, New York, USA., pp: 171-180. http://portal.acm.org/citation.cfm?id=237244
7. Guskov, I., K. Vidimce, W. Sweldens and P. Schroder, 2000. Normal meshes. Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, (ACCGIT' 00), ACM Press/Addison-Wesley Publishing Co., New York, USA., pp: 95-102. http://portal.acm.org/citation.cfm?id=344831
8. Guskov, I., W. Sweldens and P. Schroeder, 1999. Multiresolution signal processing for meshes. Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, (ACCGIT' 99), ACM Press/Addison-Wesley Publishing Co., New York, USA., pp: 325-334. http://portal.acm.org/citation.cfm?id=311577
9. Hussain, M. and Y. Okada, 2005. LOD modeling of polygonal models. Mach. Graph. Vision, 14: 325-343. http://portal.acm.org/citation.cfm?id=1140437
10. Hussain, M., Y. Okada and K. Niijima, 2006. An efficient method for converting polygonal models into displaced subdivision representation. IEICE Trans. Fundament. Elect. Commun. Comput. Sci., E89-A: 807-816. http://portal.acm.org/citation.cfm?id=1184600
11. Kobbelt, L., 2000. √3−Subdivision. Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, Aug. 2000, ACM Press/Addison-Wesley Publishing Co., New York, USA., pp: 103-112. http://portal.acm.org/citation.cfm?id=344835
12. Krishnamurthy, V. and M. Levoy, 1996. Fitting smooth surfaces to dense polygon meshes. Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, Aug. 1996, ACM Press, New York, USA., pp: 313-324. http://portal.acm.org/citation.cfm?id=237270
13. Lee, A., H. Moreton and H. Hoppe, 2000. Displaced subdivision surfaces. Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, Aug. 2000, ACM Press/Addison-Wesley Publishing Co., New York, USA., pp: 85-94. http://portal.acm.org/citation.cfm?id=344829
14. Lee, A., W. Sweldens, P. Schroder, L. Cowsar and D. Dobkin, 1998. MAPS: Multiresolution adaptive parameterization of surfaces. Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, (ACCGIT' 98), ACM Press, New York, USA., pp: 95-104. http://portal.acm.org/citation.cfm?id=280814.280828
15. Loop, C., 1987. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, Department of Mathematics. http://research.microsoft.com/apps/pubs/default.aspx?id=68540
16. Velho, L. and D. Zorin, 2001. 4-8 subdivision. Comput. Aided Geometr. Des., 18: 397-427. http://cat.inist.fr/?aModele=afficheN&cpsidt=1004012
17. Schroder, P. and W. Sweldens, 1995. Spherical wavelets: efficiently representing functions on the sphere. Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, (ACCGIT' 95), ACM Press, New York, USA., pp: 161-172. http://portal.acm.org/citation.cfm?id=218439
18. Stam, J., Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, July 1998, ACM Press, New York, USA., pp: 395-404. http://portal.acm.org/citation.cfm?id=280945
19. Won, K. J. and C.H. Kim, 2001. Direct reconstruction of displaced subdivision surface from unorganized points. Proceedings of 9th Pacific Graphics, Oct. 16-18, Tokyo, Japan, pp: 160-168. http://portal.acm.org/citation.cfm?id=883419
20. William, H., A. Saul, T. William and P. Brian, 2002. Numerical Recipes in C++, the Art of Scientific Computing. 2nd Edn., Cambridge University Press, Cambridge, ISBN: 10: 0521750334, pp: 1002.