9-1995

A Mesh Reduction Approach to Parametric Surface Polygonization

M. Asif Khan
Iowa State University

Judy M. Vance
Iowa State University, jmvance@iastate.edu

A Mesh Reduction Approach to Parametric Surface Polygonization

Abstract
Surface polygonization is the process by which a representative polygonal mesh of a surface is constructed for rendering or analysis purposes. This work presents a new surface polygonization algorithm specifically tailored to be applied to a large class of models which are created with parametric surfaces. This method has particular application in the area of building virtual environments from computer-aided-design (CAD) models. The algorithm is based on an edge reduction scheme that collapses two vertices of a given polygon edge onto one new vertex. A two step approach is implemented consisting of boundary edge reduction followed by interior edge reduction. A maximin optimization is used to determine the location of the new vertex. The concept of a visible region as the location space of the new vertex is introduced. The method presented here differs from existing methods in that it takes advantage of the fact that for many models, the exact surface representation of the model is known before the polygonization is attempted. Because the precise surface definition is known, a maximin optimization procedure, that uses the surface information, can be used to locate the new vertex. The algorithm attempts to overcome the deficiencies in existing techniques while minimizing the number of polygons required to represent a surface and still maintaining surface integrity in the rendered model. This paper presents the algorithm and testing results.

Disciplines
Computer-Aided Engineering and Design

DE-Vol. 82, 1995 Design Engineering Technical Conferences
Volume 1
ASME 1995

A MESH REDUCTION APPROACH TO PARAMETRIC SURFACE POLYGONIZATION

M. Asif Khan
Iowa Center for Emerging Manufacturing Technology
Iowa State University
Ames, Iowa

Judy M. Vance[1]
Iowa Center for Emerging Manufacturing Technology
Iowa State University
Ames, Iowa

ABSTRACT

Surface polygonization is the process by which a representative polygonal mesh of a surface is constructed for rendering or analysis purposes. This work presents a new surface polygonization algorithm specifically tailored to be applied to a large class of models which are created with parametric surfaces. This method has particular application in the area of building virtual environments from computer-aided-design (CAD) models. The algorithm is based on an edge reduction scheme that collapses two vertices of a given polygon edge onto one new vertex. A two step approach is implemented consisting of boundary edge reduction followed by interior edge reduction. A maximin optimization is used to determine the location of the new vertex. The concept of a visible region as the location space of the new vertex is introduced.

The method presented here differs from existing methods in that it takes advantage of the fact that for many models, the exact surface representation of the model is known before the polygonization is attempted. Because the precise surface definition is known, a maximin optimization procedure, that uses the surface information, can be used to locate the new vertex. The algorithm attempts to overcome the deficiencies in existing techniques while minimizing the number of polygons required to represent a surface and still maintaining surface integrity in the rendered model. This paper presents the algorithm and testing results.

MOTIVATION

This research is driven by the need to rapidly render complex models used in real-time simulations, specifically in Virtual Reality (VR) applications. Two important aspects of a VR simulation include visual realism and frame re-fresh rate (Aukstakalnis and Blatner, 1992). Today, techniques are available to produce nearly photo-realistic virtual models that can be used in virtual environments to achieve high levels of visual realism. However, these models often contain so many polygons that even today's high speed graphics-oriented workstations have difficulty rendering these highly detailed models at sufficiently high frame rates. It is the approach taken here that the designers will always be able to create models with more complexity than can be displayed with sufficient speed to achieve the illusion of reality. Therefore, a method to reduce the displayed representation of a model, without loss of surface integrity, is essential for the smooth integration of highly detailed models into virtual environments.

One approach to managing geometric complexity that is suitable for use in virtual environment building is to preprocess models before importing them into the virtual world. The preprocessing reduces the quantity of data required to describe the model while preserving the model's geometric features. This is the approach taken in this research. The algorithm presented here operates on the exact parametric surface definition of the model and constructs a polygonal mesh that is easily imported into a virtual environment. This approach has important applications in the area of engineering design where numerous existing CAD models created with parametric surfaces already exist.

LITERATURE REVIEW

A number of techniques for tessellating parametric surface have been presented in the past. These methods can generally be classified as being in one of two groups: rectangular topology methods or recursive subdivision methods.

The first group of methods is characterized by the vertices of the resultant polygonal mesh having a rectangular topology. The simplest approach to generating these vertices is uniformly sampling the parametric domain (Snyder, 1992). Although this method is theoretically very simple, the main drawback is that a large number of polygons are required to capture the features of any curved surface.

[1] send all correspondence to this author (e-mail: jmvance@iastate.edu)

A more desirable tessellation scheme would generate few polygons in regions of low curvature and many polygons in regions of high curvature. Examples of such techniques can be found in Kosters (1991) and Bajaj (1990). While these methods are sensitive to curvature considerations, they still produce a mesh having a rectangular topology. This is a severe restriction to place on the configuration of the final mesh. These methods cannot take advantage of a flat region in the surface that does not run the entire length of the parametric domain.

A second group of polygonization methods includes recursive subdivision schemes. These methods can be described in terms of two integral parts: a mechanism to determine how to subdivide the parameter space and a criterion to determine if any additional subdivision is required. One popular subdivision scheme is the recursive quadtree method. In this approach the parameter space is initially divided into four equal regions. The subdivision criterion is in turn tested on each region. The regions that do not meet the criterion are then divided into four subregions. This process is repeated until all the regions satisfy the subdivision criterion. Variations of this method are described in Snyder (1992) and Von Herzen (1987).

Although recursive subdivision methods are often more complicated to implement than the rectangular topology methods described earlier, they generally result in smaller polygonal meshes due to their less restrictive topology. However, one difficulty that arises when these methods are applied to a parametric surface is that surface cracking exists. As can be seen in Figure 1, as a result of recursive subdivision, adjacent regions of the models are divided such that they often vary in size. Cracks appear in the polygonized surface wherever smaller regions adjoin larger regions. Each subregion in the mesh must be further triangulated to eliminate the surface cracking. This results in a major drawback with this method: that the criterion is tested against rectangular regions that are significantly larger (at least twice as large) than the final triangulated regions. This produces a resultant triangular mesh that has considerably more polygons than the subdivision criterion dictates.

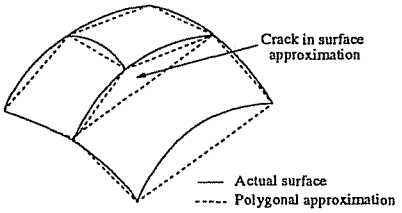

FIGURE 1: SURFACE CRACKING

The surface polygonization algorithm presented in this research is based on a mesh reduction scheme. Geometric models consisting of large numbers of triangles are routinely generated in computer graphics applications. A mesh reduction scheme attempts to reduce the number of triangles required to represent the model while retaining its geometric features.

In the past few years, a number of prominent papers have been published in this field. Schroeder et al. (1992) presents an algorithm that iteratively removes an interior vertex and re-triangulates the resulting hole. The criterion for vertex removal is the distance from the vertex to the plane approximating its surrounding vertices. A decimation scheme developed by Renze (1994) is based on a similar concept. Other mesh decimation techniques have been developed by Turk (1992) and Hoppe et al. (1993).

The preceding algorithms were designed to be used for general mesh decimation problems where the only data available is the original polygonal mesh. They assume that the precise surface definition is unknown. This is not the case when polygonizing a given parametric surface. The mesh simplification algorithm developed here takes advantage of the fact that the exact surface definition is known.

THE MESH REDUCTION ALGORITHM

The general principles of this mesh reduction algorithm can be applied to any parametric surface. CAD software such as Pro/ENGINEER (1993) and IDEAS (1991) use Non-Uniform Rational B-Spline or NURBS surfaces, as the basis for geometric modeling. Because of this standard, without loss of generality, the implementation of the algorithm is applied to NURBS surfaces. Details on NURBS curves and surfaces can be found in Piegl (1991).

The following is a brief step-by-step description of the mesh reduction algorithm. Two processes called *boundary edge reduction* and *interior edge reduction* are introduced in the algorithm and are explained in the subsequent sections. The steps include:

1) Construct subregions in the parametric domain by creating internal boundaries along lines of surface discontinuity.
2) Select an initial grid size for each subregion.
3) Subdivide the boundaries according to the grid size and apply *boundary edge reduction* to decrease the number of line segments along each boundary.
4) Construct the interior mesh based on the initial grid size and apply *interior edge reduction* to reduce the number of triangles in each subregion.

Constructing subregions

The subregions are fractions of the total parametric domain in which the surface is known to have certain continuity properties. For a NURBS surface of p and q degrees in the u and v directions, all points on the surface are C^p and C^q continuous in the respective directions. Exceptions to this rule are u and v locations that appear in the knot vectors. The surface continuity at a knot is a function of the knot multiplicity, which is the number of times the knot appears in the knot vector. If the knot multiplicity is equal to the degree in the given parametric direction, the surface is only C^0, or point, continuous. Such knot repetition often results in a crease in the surface. To detect these known locations of slope discontinuity, the knot vectors are scanned for degree-repeated knots. Internal boundaries are constructed at locations where degree-repeated knots are found.

Construction of these internal boundaries ensures that in the final tessellation: (1) a series of triangle edges lies along the slope discontinuity and, (2) no triangle straddles a slope discontinuity. Dividing the parametric domain into subregions is the only element of

the algorithm that is specific to NURBS surfaces. When applied to other types of parametric surfaces, this step is omitted.

Selecting an initial mesh size

To assemble an initial mesh on the NURBS surface, the user specifies a grid size for each subregion. If the grid size in adjacent subregions varies, the shared internal boundaries take on the larger value of the two subregions. Each boundary is uniformly discretized to build a list of vertices which are placed in a vertex list. These vertices form the endpoints of line segments or edges that are placed in an edge list.

Applying boundary edge reduction

The boundary edge reduction process is used to decimate the edge list. This process is implemented along all the internal and external boundaries of the subregions once edge lists for each boundary have been constructed. The algorithm operates on one boundary at a time. The two vertices at the extremes of a boundary are classified as being fixed and the remaining vertices are classified as free or removable. The two fixed vertices ensure that the discretized representation of the boundary spans the entire length of the original boundary.

The two types of vertices give rise to two types of edges; fixed-free edges and free-free edges. The number of edges along a boundary is decreased by iteratively deleting edges when the boundary edge reduction criterion is satisfied. The boundary edge reduction criterion evaluates whether a straight line segment is an acceptable approximation of a three-dimensional space curve.

Since the exact surface definition is known, the approach taken here is to examine the angle between the surface tangents and the line segment approximation. The tangents are in the general parametric direction of the boundary, i.e., for a boundary parallel to the u-axis, the tangents in the u direction are used. Figure 2 illustrates the vector locations of the exact surface tangents, $\mathbf{T}_i$ and $\mathbf{T}_j$, the exact surface normals at the vertices $\mathbf{N}_i$ and $\mathbf{N}_j$, and the surface approximation tangent, $\mathbf{T}_e$.

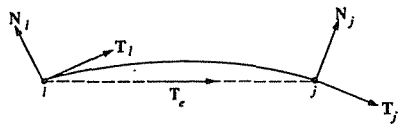

FIGURE 2: EDGE VECTOR DEFINITIONS

The user selects an acceptable boundary edge angle error, ϕ_B and a surface angle error, ϕ_I. Smaller values of ϕ_B result in a finer discretization of the boundary. The advantage of this approach is that the boundary discretization is a function of two, user-controlled variables. The surface angle error ϕ_I will be explained more fully in the next section, however, it is needed here to ensure that the initial triangles formed along the boundary satisfy the interior edge reduction criterion that will be applied in the next step of the algorithm.

The boundary edge angle criterion is defined by

$$f(\mathbf{V}_i,\mathbf{V}_j) = min\{|\cos(\phi_{max})| - \cos(\phi_B), |\mathbf{N}_i \bullet \mathbf{N}_j| - \cos(2\phi_I)\} \quad (1)$$

where

$$\cos(\phi_{max}) = max\{\mathbf{T}_i \bullet \mathbf{T}_e, \mathbf{T}_j \bullet \mathbf{T}_e\} \quad (2)$$

$$\mathbf{T}_e = \frac{(\mathbf{V}_j - \mathbf{V}_i)}{\|\mathbf{V}_j - \mathbf{V}_i\|} \quad (3)$$

and

$\mathbf{V}_i,\mathbf{V}_j$ = location vectors for nodes i and j.

The location of the vertices i and j are fixed when the boundary line segment meets the criterion. This occurs when $f(\mathbf{V}_i,\mathbf{V}_j)$ is greater than zero. Choosing a ϕ_B that is smaller than the selected ϕ_I results in boundaries that are more finely discretized than the surface interior.

A fixed-free edge is deleted by collapsing the edge onto the fixed vertex (see Figure 3). This extends the neighboring edge. The criterion is tested on this extended edge. If the criterion is met, this edge deletion is accepted and the algorithm moves on to the next edge in the list. If the criterion is not satisfied, the edge deletion is not performed.

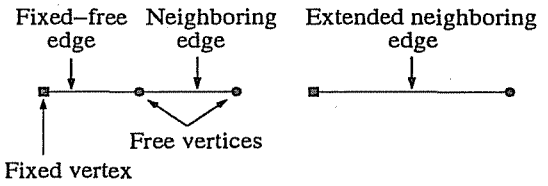

FIGURE 3: BOUNDARY EDGE REDUCTION OF A FIXED-FREE EDGE

A free-free edge is deleted by replacing the two vertices of the edge by one new vertex. As illustrated in Figure 4, this results in extending the two neighboring edges. The location of the new vertex is determined by solving a maximin optimization problem. Maximin optimization maximizes the minimum of a set of objective functions. In terms of edge reduction, the goal is to maximize the minimum value of the criterion that occurs on the extended neighboring edges which maximizes the probability that neighboring edges will be deleted in subsequent steps. This can formerly be stated as:

$$\text{maximize } min_{i \in I_f}\{f_i(u,v)\} \quad (4)$$

where $I_f = \{1,2\}$ is the set of criterion values on the extended neighboring edges. The code used in this research to solve the maximin

optimization problem is Craig Lawrence's CFSQP (Lawrence et al., 1994), C code for Feasible Sequential Quadratic Programming.

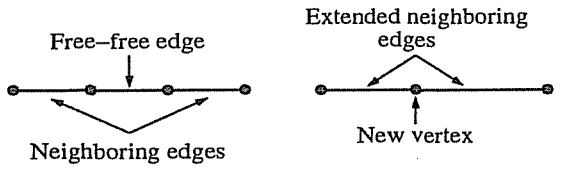

FIGURE 4: BOUNDARY EDGE REDUCTION OF A FREE-FREE EDGE

The new vertex is located such that the smaller of the two values of the criterion in the neighboring edges is maximized. This vertex can be placed anywhere within the bounds of the surrounding edges. If the new vertex results in both the neighboring edges satisfying the criterion, then the new vertex is placed and the free-free edge is deleted. Otherwise the original edge is left unchanged.

All the edge vertices remaining in the vertex list after completing boundary edge reduction are re-classified as being fixed. These vertices are not modified in the remainder of the algorithm.

Applying interior edge reduction

A uniform grid of vertices is generated in the interior of each subregion once boundary edge reduction has been completed. This set of vertices is triangulated to create an initial mesh using a constrained form of the greedy triangulation method (Preparata and Shamos, 1985). Interior edge reduction deletes edges in the interior of the initial mesh. As before, vertices in the mesh are classified as being either fixed or free. All the vertices along the boundaries are fixed and the remaining vertices are free. This vertex classification results in three types of edges: fixed-fixed, fixed-free and free-free edges. As their name suggests, fixed-fixed edges are not modified by the algorithm.

To delete a fixed-free edge, the edge is collapsed onto the fixed vertex. As illustrated in Figure 5, this results in the lengthening of neighboring triangles. The edge deletion is accepted if and only if the fixed vertex lies within the visible region and all the remaining triangles meet the edge reduction criterion. The construction of the visible region will be explained in a later section.

The interior edge reduction criterion determines the degree to which the surface is polygonized. The criterion must be such that by varying an input parameter, the accuracy of the mesh representation can be controlled. The tessellation process approximates nonplanar surface patches by flat triangle elements. Hence for interior edge reduction the criterion must judge when a planar representation of a nonplanar surface patch is acceptable. A number of attributes can be used to measure representation quality. Two candidates are distance and surface normals.

The edge reduction criterion implemented in this research is based on surface normals. Normals determine the visual characteristics of a surface as they are used in light reflection and shading algorithms. The curvature of a surface is also indicated by changes in the direction of surface normals, therefore preserving accurate surface normal information is critical.

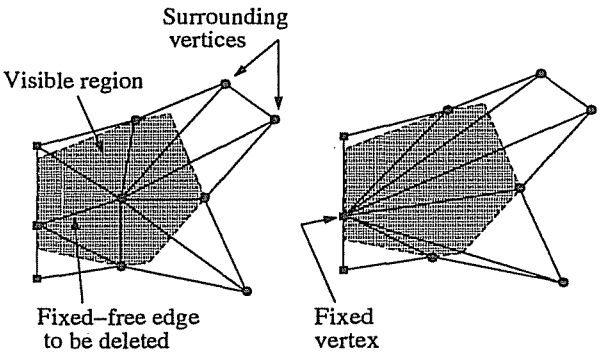

FIGURE 5: INTERIOR EDGE REDUCTION OF A FIXED-FREE EDGE

The criterion developed here is referred to as the interior edge angle criterion and can be stated as follows:

> Given a triangular surface patch defined by three vertices, $\mathbf{V}_i$, $\mathbf{V}_j$, $\mathbf{V}_k$, the angle between the normal of the triangle formed by these vertices, $\mathbf{N}_t$, and the normal of any point on the original surface, $\mathbf{N}_p$, must be less than ϕ_I.

In the testing it was found that assuming a high resolution initial mesh has been defined, it is only necessary to compare the triangle normal with the normals at the three corner vertices, $\mathbf{N}_i$, $\mathbf{N}_j$, and $\mathbf{N}_k$. The location of these normals is shown in Figure 6.

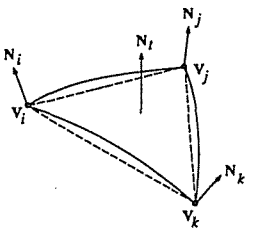

FIGURE 6: SURFACE VECTOR DEFINITIONS

The angle criterion is defined mathematically by,

$$f(\mathbf{V}_i,\mathbf{V}_j,\mathbf{V}_k) = \left|\cos(\phi_{max})\right| - \cos(\phi_I) \tag{5}$$

where

$$\cos(\phi_{max}) = max\{\mathbf{N}_i \bullet \mathbf{N}_t, \mathbf{N}_j \bullet \mathbf{N}_t, \mathbf{N}_k \bullet \mathbf{N}_t\} \tag{6}$$

$$\mathbf{N}_t = \frac{(\mathbf{V}_j - \mathbf{V}_i) \times (\mathbf{V}_k - \mathbf{V}_i)}{\left\|(\mathbf{V}_j - \mathbf{V}_i) \times (\mathbf{V}_k - \mathbf{V}_i)\right\|} \tag{7}$$

ϕ_I = acceptable surface angle error

A triangle is said to satisfy the criterion if $f(V_i,V_j,V_k)$ is greater than zero. The user selects the angle ϕ_I to determine the desired accuracy of the surface polygonization. In this manner the user can create several level-of-detail models by selecting various values of ϕ_I. A favorable property of a mesh constructed using this angle criterion is that the angle between the normals of adjacent triangles is guaranteed to be less than or equal to $2\phi_I$. This property ensures that there are no artificial creases in the rendered object.

To delete a free-free edge, the two edge vertices are replaced by one new vertex, thereby deleting two neighboring triangles (See Figure 7). The new vertex is positioned in the visible region such that it maximizes the smallest value of the criterion that occurs among the neighboring triangles. This location is determined by solving a maximin optimization problem of the form:

$$\text{maximize} \quad min_{i \in I_f}\{f_i(u, v)\} \text{ s.t. } (u, v \in V) \qquad (8)$$

where f_i are the values of the criterion on each of the extended neighboring triangles, $I_f = \{1,...,n_f\}$, in the visible region V.

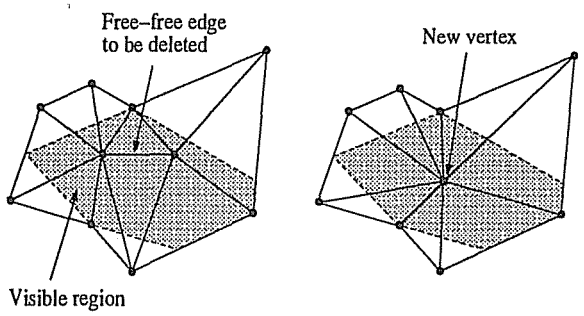

FIGURE 7: INTERIOR EDGE REDUCTION OF A FREE-FREE EDGE

The optimization algorithm implemented here is a gradient-based technique. A property of gradient techniques is that they are sensitive to the initial location and tend to converge to locally optimal solutions. To overcome the difficulties posed by this limitation, multiple starting points are used. To attempt to find the global solution, the optimization method must be started from a number of locations in the search space. If the remaining triangles meet the edge reduction criterion and satisfy a local geometry check, the edge reduction is accepted and the algorithm moves on to the next edge in the edge list. Otherwise the original edge is left unchanged. This optimization method was chosen over other non-gradient methods, such as genetic and simulated annealing, because the overall performance is faster.

VISIBLE REGION CREATION

The visible region of an edge is a subset of the parametric space bounded by the neighboring vertices of the edge. This region defines the feasible domain for the location of a new vertex when attempting to delete an edge.This area is named the visible region since any vertex placed within this space is visible to all the neighboring vertices. That is, a line connecting the new vertex and any neighboring vertex (in parametric space) will not intersect the surrounding edges.

Visible regions were constructed using heuristic means for a number of surrounding vertex sets of arbitrary topology. It was observed that the corners of the visible region are defined by either surrounding vertices or the intersection points of extended surrounding edges. Based on this observation, the following algorithm was developed for constructing visible regions:

1. Place all the surrounding vertices in a vertex list.
2. Extend all the surrounding edges out to infinity as shown in Figure 8.
3. Cycle through each extended surrounding edge and compute the intersection of this edge with all other extended edges. Note that some edges may not have suitable intersection points due to being parallel or collinear. Add all the intersection points to the vertex list from step 1.
4. Check the visibility of each point in the vertex list with respect to the surrounding vertices. If any vertex in the list is not visible to any of the surrounding vertices, remove it from the list.
5. Remove coincident vertices from the vertex list. The remaining vertices define the bounds of the visible region (Figure 9). Sort these vertices such that they traverse the visible region in a counterclockwise direction.

This algorithm was developed independently but a survey of related work is documented in O'Rourke (1987).

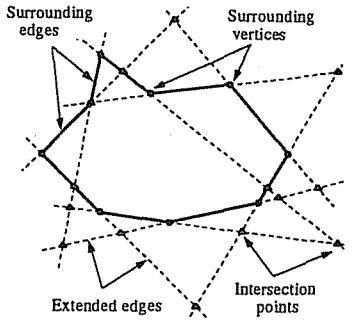

FIGURE 8: INTERSECTION POINTS OF EXTENDED SURROUNDING EDGES

Limiting the new vertex location to the visible region ensures that edge reduction produces non-intersecting edges in parametric space. This however does not guarantee non-intersecting edges on the surface mesh. Before accepting an edge reduction step, the geometry of the remaining triangles must be checked. The interior edge reduction criterion has the property that the angle between the normals of adjacent triangles (that satisfy the criterion) will be less

than or equal to $2\phi_I$. If the triangles remaining after edge reduction overlap or result in intersecting edges, the normals of the triangles computed by traversing their vertices in a counter-clockwise direction (in parametric space) will not display the above stated property. Hence these triangles are detected and the associated edge reduction is rejected.

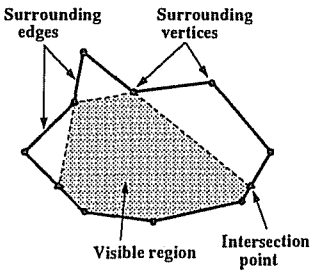

FIGURE 9: FINAL VISIBLE REGION

ALGORITHM TESTING

A number of tests were carried out using the algorithm developed in this research. Three initial surfaces and the resulting reduced meshes are shown in Figures 10 and 11. The NURBS surfaces initially were created in three sections and the initial mesh is shown after edge reduction has been completed. The results of tests on these surfaces and others were analyzed to determine the properties and characteristics of the surface polygonization technique.

A set of input parameters must be specified to implement the algorithm. These values may affect the size and the quality of the final reduced mesh. The following are the primary input variables:

1. Specify the number of optimization locations to try when attempting to delete a free-free edge.
2. Specify the location in the mesh where to start the edge reduction process.
3. Specify the initial grid size in each subregion of the parametric domain.
4. Specify the angle criteria values for both boundary and interior edge reduction.

Number of optimization starting locations

When attempting to delete a free-free edge, the location of the new vertex is determined by solving a maximin optimization problem. The code used to solve this problem is a gradient based technique called CFSQP. A property of gradient techniques is that the choice of the initial location can result in convergence on locally optimal solution. To attempt to find the global solution, the optimization method must be started from a number of locations or guesses in the search space.

Since the optimization process can be time consuming, the option to implement edge reduction with no optimization is also available. In this case, the new vertex is arbitrarily placed at the centroid of the visible region and the angle criterion is evaluated on the surrounding triangles. If all the surrounding triangles satisfy the angle criterion, the edge deletion is accepted. Arbitrarily locating the new vertex in this manner is similar to general mesh decimation algorithms (Schroeder et al., 1992) and hence is provided for comparison purposes.

The effect of the number of optimization guesses was examined in a series of tests on models ranging up to 10,000 triangles. The results related to the three surfaces presented here are summarized in Table 1.

Table 1: FINAL MESH SIZE AS A FUNCTION OF THE NUMBER OF OPTIMIZATION GUESSES

Optimization guesses	Average final mesh size (in triangles)		
	Surface *A*	Surface *B*	Surface *C*
0	50	754	1427
1	44	717	1447
2	44	600	1320
3	44	592	1245
4	44	592	1242
5	44	588	1241

It was found that implementing mesh reduction with five optimization guesses compared to no optimization produces final meshes that have 14% to 28% fewer polygons. This however is at a significant cost in terms of run-time. When using a Silicon Graphics Indy 150 MHz computer, mesh reduction with no optimization takes less than a minute for these tests. With five optimization guesses the process can take up to thirty minutes. However, the considerably fewer polygons suggests that the optimization-based algorithm developed in this research, for the specific problem of surface polygonization, can reduce meshes to a greater degree than general mesh decimation techniques. Execution time is also not a major concern as the algorithm is proposed as a off-line preprocessor that reduces the size of geometric models produced by CAD software.

The results also show that as the number of optimization guesses increases, the size of the reduced mesh asymptotically approaches a value. This indicates that by starting the optimization from several locations, the globally optimal solution is being found. From these tests and all other tests conducted on a variety of surfaces, it has been observed that for practical purposes, three optimization guesses prove to be sufficient.

Starting location

Once an initial list of edges has been constructed, the edge reduction algorithm proceeds to decimate this list by choosing an arbitrary starting position in the list and sequentially work through the list until no more edge reduction occurs. The edge reduction is implemented using both zero and three optimization guesses. As in the previous section, tests on the three surfaces indicate that in all cases optimization produces surfaces with fewer polygons compared to no optimization. It was found that choosing an arbitrary starting location has little effect on the mesh size (see Table 2).

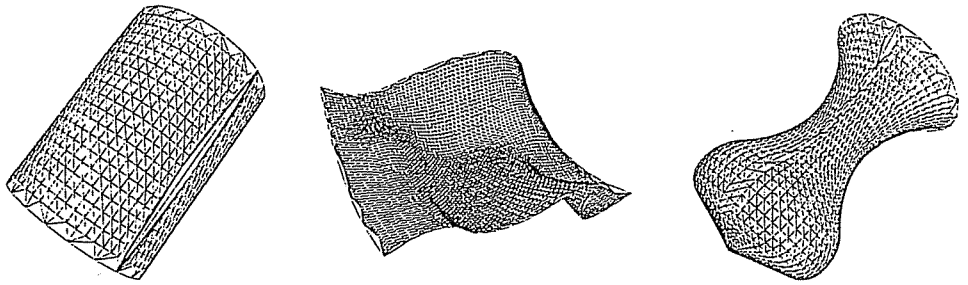

FIGURE 10: INITIAL MESH

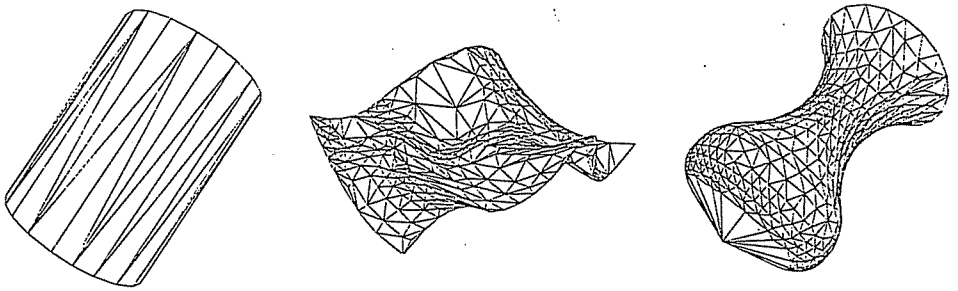

FIGURE 11: REDUCED MESH

Table 2: FINAL MESH SIZE AS A FUNCTION OF STARTING LOCATION

Surface	Optimization guesses	Average final mesh size	Standard Deviation (%)
A	0	48	13.8
	3	42	0.0
B	0	736	3.5
	3	589	1.8
C	0	1414	0.7
	3	1230	1.9

Table 3: FINAL MESH SIZE AS A FUNCTION OF THE INITIAL GRID SIZE

Surface	Optimization guesses	Average final mesh size	Standard Deviation (%)
A	0	46	15.0
	3	42	0.0
B	0	763	5.4
	3	603	1.6
C	0	1454	3.8
	3	1252	0.4

Initial grid size

The size of the initial grid of vertices determines the initial mesh resolution. If all the triangles in the initial mesh satisfy the angle criterion, the edge reduction algorithm guarantees that the final mesh also retains this property. This is because the algorithm rejects all edge deletions that result in triangles which fail the criterion. If a number of initial triangles violate the criterion, the final mesh produced by the reduction process is guaranteed to have the same or fewer number of violating triangles. Though the grid size can be independently chosen for each subregion, in the current implementation the same grid size is used for all subregions.

The results of the testing indicate the favorable property that decimation with three optimization guesses is insensitive to the initial grid size (Table 3). In contrast, the initial grid size does have some influence on the final mesh from zero optimization tests.

Choosing an angle criterion

A number of the current decimation algorithms use a distance-based criterion. A major disadvantage of such criteria is that the value that produces an acceptable mesh varies considerably depending on the size and shape of the surface. The angle criterion developed in this research does not suffer from this drawback. Equal importance is given to small and large features since the criterion is an angular and not a scalar quantity. From extensive tests, it was found that a 5° boundary criterion and a 10° interior criterion produce a good visual surface. The flexibility of this criterion selection is useful when choosing to make several different level-of-detail models for a simulation.

Further details of these tests can be found in Khan (1994).

CONCLUSIONS

A new technique for parametric surface polygonization is described and test results presented. This algorithm provides a method to tessellate a surface into a reduced polygonal mesh while retaining all salient geometric features. The research is driven by the need to reduce the polygon count of complex models that must be rapidly rendered. This is of special importance for real-time simulations and virtual reality applications.

This algorithm differs from other surface polygonization techniques presented in the past. The majority of previous work restricts the final mesh to a rectangular topology or is based on recursive subdivision schemes. The method developed in this research models surface polygonization as a specialized form of a mesh reduction problem. Unlike general mesh decimation techniques, this method is able to take advantage of the fact that the precise surface definition is known. The implementation here is demonstrated on NURBS surfaces although the method can be applied to any parametric surface.

The mesh reduction approach begins by dividing the parametric domain into a number of subregions. This is the only element of the algorithm that is specific to NURBS surfaces and does not apply to other forms of parametric surfaces. An initial grid of vertices is constructed in each subregion. The number of vertices defining the surface is reduced by a general process called edge reduction. This process is first applied along the boundaries and then in the interior of the subregions. In edge reduction, an edge in the mesh is deleted by replacing it's two vertices with one new vertex. This new vertex is located by solving a maximin optimization problem or is arbitrarily located at the centroid of the feasible region. Since the algorithm operates on local subsets of the mesh, global minimization of the mesh size and symmetry properties are not guaranteed.

A series of tests are conducted to evaluate the performance of this polygonization technique. The method proves to be insensitive to the initial grid size and the starting location of the reduction process. Optimization-based mesh reduction produces substantially smaller meshes compared to implementation without optimization. As general decimation techniques are similar to the no optimization case, the test results imply that with optimization, the technique developed here is more effective for parametric surface polygonization problems. The longer execution time is not a major concern as this method is proposed as a preprocessor that reduces the size of geometric models. The quality of the mesh is determined by a user-selected angle criterion. Satisfaction of this criterion ensures that a shaded rendering of the polygonal mesh is a close approximation of the original surface.

ACKNOWLEDGMENTS

The authors are grateful to Craig T. Lawrence, Jian L. Zhou, and Andre L. Tits, for permitting the use of their CFSQP routines. This project was funded by the Iowa Center for Emerging Manufacturing Technology at Iowa State University.

REFERENCES

Aukstakalnis, S. and Blatner, D. *Silicon Mirage: The Art and Science of Virtual Reality*, Peachpit Press, Berkeley, CA, 1992, p. 266.

Bajaj, C. L. 'Rational hypersurface display' *Comput.-Aided Design* Vol 24 (1990) pp 117-127.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. 'Mesh optimization' *Computer Graphics* Vol 27 (1993) pp 19-26.

Khan, M., *A mesh reduction approach to parametric surface polygonization*, M.S. Thesis, Iowa State University, 1994.

Kosters, M. 'Curvature-dependent parameterization of curves and surfaces' *Comput.-Aided Design* Vol 23 (1991) pp 569-578, 1991.

IDEAS, Solid Modeling, User's Guide Structural Dynamics Research Corporation (1991).

Lawrence, C., Zhou, J. L. and Tits, A. L. 'User's guide for CFSQP version 2.0: A C code for solving large scale constrained nonlinear minimax optimization problems, generating iterates satisfying all inequality constraints' *Electrical Engineering Department Institute for Systems Research TR-94-16*, University of Maryland, College Park (1994).

O'Rourke, J. *Art Gallery Theorems and Algorithms*, Oxford University Press, New York, 1987.

Piegl, L., 'On NURBS: A survey' *IEEE Computer Graphics & Applications,* Vol 11 (1991), pp 55-71.

Preparata, F. and Shamos, M. *Computational Geometry*, Springer-Verlag, New York, pp. 235-237, 1985.

Pro/ENGINEER Modeling User's Guide Parametric Technology Corporation (1993).

Renze, K. J. 'Unstructured surface and volume decimation of tessellated domains' *Ph.D. dissertation*, Iowa State University (1994).

Schroeder, W. J., Zarge, J. A., and Lorensen, W. E. 'Decimation of triangle meshes' *Computer Graphics* Vol 26 (1992) pp 65-70.

Snyder, J. M. *Generative Modeling for Computer Graphics and CAD* Academic Press, Inc. (1992).

Turk, G. 'Re-tiling polygonal surfaces' *Computer Graphics* Vol 26 (1992) pp 55--64.

Von Herzen, B. and Barr, A. H. 'Accurate triangulations of deformed, intersecting surfaces' *Computer Graphics* Vol 21 (1987) pp 103-110.