Mesh Simplification using Ellipsoidal Schema for Isotropic Quantization of Face-Normal Vectors

Ganesan Subramaniam and Kenneth Ong

Department of Electronics and Computer Engineering
National University of Singapore
Block E4, Level 5, Room 48, 4 Engineering Drive 3, Singapore 117576

engp1661@nus.edu.sg, eleongk@nus.edu.sg

Abstract

In this paper, we present a method for simplification of arbitrary 3D meshes that is based on Isotropic Quantization of face-normal vectors. There are three stages. Firstly, a codebook that contains the unique face-normal vectors of the 3D mesh is generated using our Ellipsoidal Schema. Secondly, the polygons of the mesh are grouped into patches: based on the codebook vectors and the locality information of the polygons. Polygons that have isotropic and geographical similarities are grouped together. And the resulting patch is approximately a flat plane with its corresponding codebook vector as its normal. In the last stage, our mesh simplification technique re-triangulates the patch, in which the algorithm only considers the vertices on the borders of the patch. We demonstrate that our technique yields better results when applied with multiple iterations than when using a single iteration. Thus using patch-wise quantization, our technique is able to simplify 3D meshes.

Keywords: Mesh Simplification, Ellipsoidal Schema, Isotropic Quantization, Codebook

1 Introduction

The performance of graphics subsystems has improved enormously in the last few years. However the complexity of 3D scenes and graphics applications has also increased greatly. A very critical aspect in interactive 3D graphics is the complexity, in terms of number of triangles, of the meshes to be processed and rendered. A large number of meshes can be easily produced in many applications: by fitting isosurfaces on large data-sets, that is by converting surfaces into triangulated meshes, by 3D scanning real-world objects. All 3D scanners perform a regular sampling of the surface of an object, returning triangle meshes of complexity directly proportional to the scanner's sampling resolution and the object surface area. The resulting meshes typically have a few million triangles. These complex meshes introduce severe overheads in transmission, rendering, processing and storage.

Thus the visualization of such models cannot be realized in real-time without mesh simplification techniques to reduce the number of triangles that are rendered while maintaining the quality of the mesh. For digital geometry processing, most scanned models must undergo complete re-meshing before processing. Geometry-processing algorithms such as smoothing and compression can greatly benefit from parameterization-based re-meshing techniques, combined with uniform or curvature adapted sampling.

Our main contribution to the domain of mesh simplification is the formulation of a dynamic codebook schema for isotropic surface sampling. We present in this paper, the Ellipsoidal Schema (see Section 2.1.1 and 2.1.2) and its variants that is dependent on our geometric model called the GAEA [1]. The GAEA is used in the creation of the Ellipsoidal Schema codebook that is based on the face-normal vectors of the GAEA face-normals. It also ensures distinct separation of the regions formed by the codebook vectors (also known as code-vectors). Due to the inherent properties of a GAEA, it is possible to create both uniformly and non-uniformly distributed set of code-vectors. These code-vectors are later used by the Polygon Grouping stage as shown in Figure 1 (see next page).

The generated codebook is applied to the 3D mesh in the Polygon Grouping stage (see Section 2.2). Here the polygons of the 3D mesh are grouped into non-overlapping regions based on orientation and locality of the polygons. In the final stage of our mesh simplification process, we re-mesh the isosurfaces created in the Polygon Grouping stage. Here we employed Edge Collapse algorithm (see Section 2.3) to flatten out the polygons. To test the effectiveness of our mesh simplification technique, we applied our mesh simplification technique several models of varying sizes (see Section 3) and show our findings when the simplification technique is iterated on the same model.

2 Mesh Simplification Technique

Our mesh simplification technique focuses on the generation of a codebook for quantizing a 3D mesh into patches (groups of polygons). There are three steps involved for the creation of the simplified mesh (see Figure 1). They are Codebook Generation (see Section 2.1), Polygon Grouping (see section 2.2), and Patch Re-meshing (see section 2.3). Each of these processes is discussed in greater detail in the following subsections.

[1] GAEA is the name that we have given for the new geometric model that we are introducing and it is not an acronym.

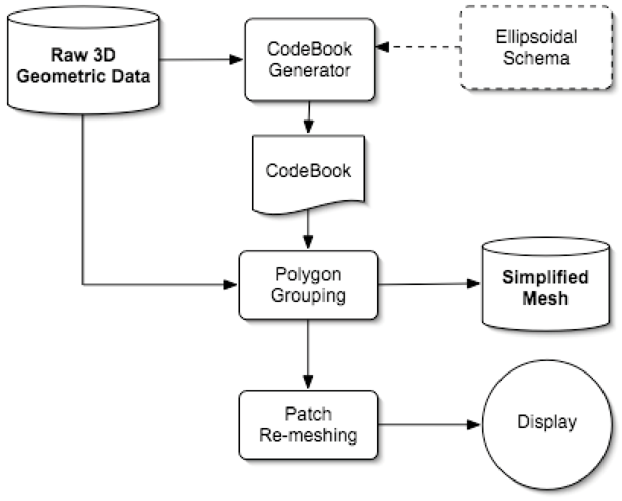

Figure 1: Shows the overall process of our mesh simplification technique.

2.1 Codebook Design

The Polygon Grouping process requires a codebook to quantize the face-normal vectors of a 3D mesh. However designing a codebook can sometimes requires an exhaustive search for the best possible code-vectors in space; and the search increases exponentially as the number of code-vectors increases. Therefore we resort to a suboptimal codebook design schema.

We introduce the Ellipsoidal Schema that uses a pre-defined general model called GAEA-*n* (see Section 2.1.1 for the explanation of -*n*) whose code-vectors are uniformly distributed about the origin. This schema can be extended to create a non-uniformly distributed codebook, GAEA-*xyz* (see Section 2.1.2 for the explanation of -*xyz*) using the pre-defined general model.

2.1.1 Uniform Ellipsoidal Schema

To produce our Uniform Ellipsoidal Schema codebook, we introduce a new geometric model called GAEA-*n*. A GAEA-*n* is basically a sphere, which is symmetrical about ***n/2*** different directions (much like a cube) and has ***n*** number of subdivisions along each axis. In fact, we can create a GAEA-*n* by expanding all the vertices of a cube to fit a sphere's *profile* (see Figure 2).

A GAEA-*n* model provides a uniformly[2] distributed set of code-vectors that shall be used in the quantization of the orientation angles of the face-normal vectors of a 3D mesh. The ***n*** in GAEA-*n* refers to the number of subdivisions along the each axis where $n > 0$. For example, GAEA-3 means that this object is created from a cube that has 3 subdivisions along each axis.

A GAEA-*n* object has two notable properties. One of which is that a GAEA-1 is actually a cube with 1 subdivision. This means that a GAEA-*n* object shall have at least 6 unique face-normals and each one lies along one of the three axes (i.e. X, Y, and Z axes). The second property of the GAEA-*n* is that all its face-normals are unique and are

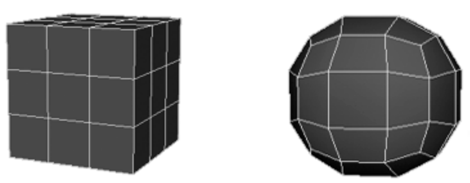

Figure 2: Shows that a cube with 3 subdivisions can be transformed into a GAEA-3 whose face-normals form the vectors for the codebook of the Ellipsoidal Schema.

uniformly distributed. This makes it a suitable candidate for creating a uniformly distributed set of code-vectors. We call this schema the Uniform Ellipsoidal Schema.

Since our goal is to quantize the face-normal vectors of a 3D mesh, we would need to choose the right number of subdivisions for the GAEA-*n* to create a comprehensive codebook. To aid in this decision, it is important to note that the number of code-vectors obtained through this schema is directly related to the number of subdivisions based on the following equation:

$$V = 6 * n^2$$

The equation above shows that the codebook size has a quadratic relationship with the number of subdivisions. In other words, a small increase in the number of subdivisions shall yield a large number of code-vectors. For example, GAEA-3 will yield 54 code-vectors while GAEA-4 will yield 96 code-vectors. As the codebook size increases, the resolution (refers to the number of polygons) of the simplified mesh will also increase with the number of quadrilaterals in the simplified mesh.

2.1.2 Non-Uniform Ellipsoidal Schema

GAEA-*n* provides a general solution to quantizing orientation angles of a mesh and thus, can be applied to all meshes. In some cases however, a uniformly distributed codebook may not be adequate enough to describe the 3D mesh with the desired accuracy. This might become apparent in cases where only certain face-normal vectors are predominant in the mesh. A uniformly distributed set of code-vectors tends to over generalize this mesh, resulting in the loss of key details in the final simplified mesh.

To address this problem, we extend GAEA-*n* such that certain ranges of face-normal vectors are sampled at a higher frequency than others. This property can be achieved by varying the number of subdivisions in each axis independently. To cater to this modification, we introduce a new model called GAEA-*xyz*.

The *xyz* in GAEA-*xyz* denotes the number of subdivisions along the respective axes . Since each axis is now controlled separately, the formula for finding the number of code-vectors of a GAEA-*xyz* object shall be given as follows:

$$V = (x * y + y * z + z * x) * 2$$

[2] Uniform distribution here means that the angles between the face-normals of the GAEA-*n* are equal.

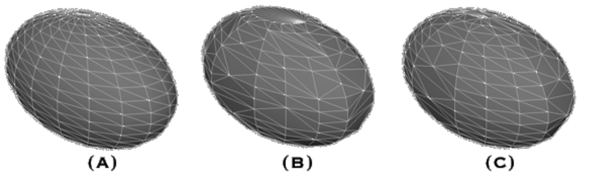

Figure 3: Shows (A) original mesh (B) after applying GAEA-3 codebook (C) after applying GAEA-5-4-3.

By varying the subdivisions along each axis, GAEA-*xyz* creates a codebook for the 3D mesh as shown in Figure 3. Unlike GAEA-*n*'s codebook, this codebook is able to relax quantization on areas that have fewer features and emphasizes quantization on areas that have more features. The values for *x*, *y*, and *z* can be found by determining the bounding box dimensions of the 3D mesh. Then, for a desired codebook size, one can determine the values of the subdivisions for a GAEA-*xyz* based on a pre-determined filter.

Having determined the initial codebook for the 3D mesh, we can now proceed to the Polygon Grouping stage (see Section 2.2). We refine the codebook to accommodate the locality criteria for our mesh simplification technique. We accomplish this by first grouping the polygons based on the orientation angles of the code-vectors. In other words, all polygons that face a certain (approximate) angle shall be grouped together. This angle is provided by the vectors of the codebook. Then using the adjacent polygon information from the 3D mesh, we break down the groups into smaller sets of connected polygons. Finally the averages of the face-normal vectors of each set of connected polygons shall form the code-vectors for the new codebook.

2.2 Polygon Grouping

Polygon Grouping is a process that maps a 3D mesh's face-normal vectors (in the vector space R^k) into a codebook, $Y = \{y_i\!: i = 1, 2... N\}$ where y_i is a finite set of code-vector. Associated with each code vector, y_i, is a nearest neighbor region, and is defined as:

$$V_i = \{x \in R^k : \|x - y_i\| \le \|x - y_j\|, \quad for\ all\ j \ne i$$

The representative code-vector is determined to be the closest in Euclidean distance from the input vector. Euclidean distance accounts for the locality of the polygons in a 3D mesh. However for the purposes of Polygon Grouping, we have decided that orientation is also important. In other words, the polygons that are close together in Euclidean space and are facing in the same direction shall be clustered together to form a group. To accommodate the latter criterion, the orientation-based grouping, we redefine our neighbor region as follows:

$$V_i = \begin{cases} x \in R^k : \|x - y_i\| \le \|x - y_j\|, & for\ all\ j \ne i \\ cos\left(\dfrac{x \cdot y_i}{\|x\| \cdot \|y_i\|}\right) \le cos^{-1}\left(\dfrac{x \cdot y_j}{\|x\| \cdot \|y_j\|}\right), & for\ all\ j \ne i \end{cases}$$

Thus for Polygon Grouping, the representative code-vector is determined to be the closest in Euclidean distance from the input vector and closest in orientation to the input vector. During the Polygon Grouping process, an input face-normal vector is matched against the code-vectors generated from an Ellipsoidal Schema and the index to the code-vector that offers the least distortion is output. In this case the lowest distortion is found by evaluating the Euclidean distance and the orientation angle between the input vector and the code-vector in the codebook.

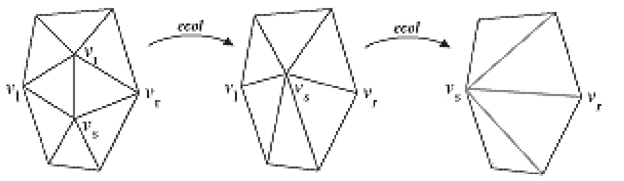

Figure 4: Illustration of the flattening process of a patch using a series of edge collapse transformation.

2.3 Patch Re-meshing

After grouping polygons into patches, we are now left with the task of flattening the patches. Since the patches consist of polygons that are isotropic, each patch can be treated as a flat 2D plane. Thus removing the vertices in the interior of the patch should not affect the fidelity of the simplified mesh to the original mesh too greatly. The remaining boundary vertices shall be triangulated to form the new patch. Although any polygon triangulation algorithms can be used, we have implemented edge-collapsing technique to flatten a patch.

As shown in Figure 4, an edge collapse transformation $ecol(\{v_s, v_t\})$unifies 2 adjacent vertices v_s and v_t into a single vertex. The vertex v_t and the two adjacent faces $\{\ v_s\ v_t\ v_l\ \}$and $\{\ v_t\ v_s\ v_r\ \}$ vanish in the process. A position *s* is specified for the new unified vertex.

To begin an edge-collapse operation, we first have to select an edge based on a cost-function. Our cost function shall be the shortest edge with an *interior vertex*. An interior vertex is one that does not lie on the boundary of a patch. Since our goal is to flatten a patch, the interior vertex shall collapse into a boundary vertex. Thus for a set of $\boldsymbol{\beta}$ boundary vertices of a patch, the cost function for the selection of an edge can be given as follows:

$$C = \{min\,[\,|v_s - v_i|\,],\ for\ i \in \beta$$

Using this equation, we can then proceed to flatten a patch and thus complete our simplification process. Note however that other polygon triangulation techniques can be used to achieve our goal. However we have chosen this technique due to its simplicity and ease of implementation.

3 Results and Discussion

The mesh simplification technique described in this paper has been implemented as an interactive software. The user shall be able to control the simplification via the selection of parameters for our codebook generation schema. Models in PLY formatted files can be loaded into the program directly and the Ellipsoidal schema can be applied onto them. Figure 5 shows the result of the applying our mesh simplification schema using our software.

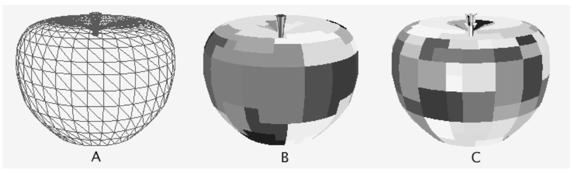

Figure 5: Shows (A) original mesh (B) after applying GAEA-3 codebook (C) after applying GAEA-5.

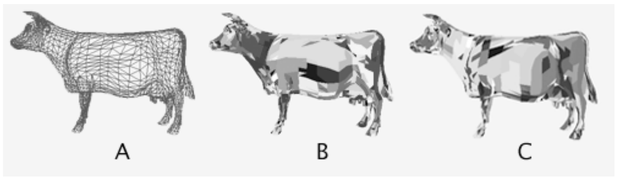

Figure 6: Shows (A) original mesh (B) after applying GAEA-3 codebook (C) after applying GAEA-5 to GAEA-1 iteratively.

We have run our technique on a variety of models of arbitrary complexity. Figure 5 illustrates simplification of a polygonal model with 1704 polygons. After applying our simplification technique, the new model has 1548 polygons by using GAEA-3 codebook and 1672 polygons by using GAEA-5 codebook. The following table shows all the results for the polygonal models on which we have used our technique.

From the Table 1 above, we can deduce that while a smaller order GAEA produces a simpler resulting mesh, it also suffers in terms of quality. However in another observation of the results in the table above, it seems that the algorithm is only able to produce minimal reduction in the number of polygons. To achieve higher rates of polygon reduction, we have realized that a model needs to be passed through our algorithm for several iterations. In each iteration, if the number of polygon does not reduce, we would reduce the GAEA order till GAEA-1 is reached. The results of this experiment is shown Figure 6 where the cow model was put through this test. We started with GAEA-5 and ended the algorithm after GAEA-1.

We have also discovered that instead of using uniform codebooks during the iterations, using non-uniform codebooks yielded more polygon reductions. This is due to the properties of the non-uniform codebooks to "better" fit the model. Based on the shape of the model we have selected appropriate values for the dimensions of GAEA-*xyz*. During the iterations, we reduce the every dimension of GAEA till they are all 1s (i.e. till GAEA-*1-1-1* is reached). The results are as follows:

	apple.ply	**cow.ply**	**bunny.ply**	**dragon.ply**
Original	1704	5804	69451	871414
Using GAEA-1	1216	3074	17993	265776
Using GAEA-3	1548	5274	37660	435814
Using GAEA-5	1672	5614	48229	550340

Table 1: Shows the polygon counts after applying the mesh simplification technique on several models.

	cow.ply	**bunny.ply**	**dragon.ply**
Original	5804	69451	871414
Initial GAEA	5-3-1	5-5-3	5-5-1
Simplified Mesh	2755	10857	130611
Timing	27 secs	65 secs	935 secs

Table 2: Shows the results when iterative simplification and non-uniform codebooks were used.

4 Conclusion

Our Mesh Simplification technique involves three steps as illustrated in Figure 1. The first step creates a codebook based on either of one of our polygon quantization schemas: Uniform Ellipsoidal Schema and Non-Uniform Ellipsoidal Schema. In the second step of our mesh simplification process, we group polygons based on the resulting codebook from step 1. The grouped polygons shall be isotropic and have similarities in locality. This thus allows us to flatten each group of polygons without too much loss of fidelity. As observed in step 3, different codebooks provide different results. Thus the choice of the quantization schemas can affect the size of the final simplified mesh.

5 References

MichaelGarland and Paul S. Heckbert (1997): Surface simplification using quadric error metrics, SIGGRAPH, pages 209–216. http://www.cs.cmu.edu/~garland/quadrics.

Cohen, J., M. Olano, and D. Manocha (1998): Appearance-Preserving Simplification, SIGGRAPH.

Hugues Hoppe (1996): Progressive meshes, SIGGRAPH, pages 99–108, http://research.microsoft.com/hoppe/

W. Sweldens and P. Schr¨oder (2001): Digital Geometry Processing, Course Notes. ACM SIGGRAPH.

X. Gu, S. Gortler, and H. Hoppe (2002): Geometry images, SIGGRAPH, pages 355–361.

P. Alliez, ´ E. Colin de Verdi`ere, O. Devillers, and M. Isenburg (2003): Isotropic surface remeshing, Shape Modeling International.

P. Alliez, M. Meyer, and M. Desbrun (2002): Interactive geometry remeshing. ACM Transactions on Graphics, 21(3):347–354. SIGGRAPH conference proceedings.

M. Desbrun, M. Meyer, and P. Alliez (2002): Intrinsic parameterizations of surface meshes, Eurographics, pages 209–218.

B. L´evy, S. Petitjean, N. Ray, and J. Maillot (2002): Least squares conformal maps for automatic texture atlas generation, SIGGRAPH, pages 362–371.

Pragyana Mishra, Omead Amidi and Takeo Kanade (2004): EigenFairing: 3D Model Fairing using Image Coherence, British Machine Vision Conference, Vol. 1, pp. 17-26

Markus Hadwiger (1998): Mesh Simplification and Multiresolution Data Structures,

http://www.cg.tuwien.ac.at/studentwork/VisFoSe98/msh/