Decimation of meshes for BEM calculations.

Manuel A. Sobreira Seoane
E.T.S.I. de Telecomunicación, University of Vigo; 36200 Vigo, Spain e-mail: msobre@gts.tsc.uvigo.es

Three dimensional surface meshes are used in several computer graphics and calculation applications. One of the fields we can find these models in is Acoustics. In regards to the acoustic field calculations, often the models we get from CAD programs or 3D laser scanners are excessively detailed for low frequency calculations. A decimation process of triangular meshes is presented in this paper. Decimation is the concept of removing a large number of polygons from the original mesh, maintaining certain quality in the resulted new mesh. Many solutions and different approaches have been tested. The acoustics implications of this process will be discussed.

1 Introduction

Following the requirements of a specific application, we could find that the level of detail of the surface meshes used during the process is quite high. That may result in an over-consumption of our storage memory and simulation time. For that concrete reason we need to reduce this level, simplifying the surface meshes.

During the last decade, mesh decimation has been studied for different reasons and utilities. Many different approaches have been investigated, viewing in detail how they work with decimation, how fast and accurate they are, and finally which ideas or features we can pick from them to apply in our solution.

Moreover, we must look at the final purpose of our work. We have focused on obtaining meshes with lower level of detail to reduce the mass storage and simulation time requirements working in different acoustic test cases. On the revision of algorithms and its implementation, the selection is based on several characteristics:

1. Efficiency: obtaining these simplified versions of the complex original model in a relative short time.

2. Quality: taking care that also a high fidelity to the original mesh is maintained, preserving the primary features even after significant simplification.

3. Acoustic Field Calculations: assuming the model to consist only of triangles, try to keep the relation $L < \lambda/6$ for every segment length [6].

Mixing up the ideas taking from the several revised works and our final goals, we finally get to three different solutions:

1. The fastest and easiest solution, mainly based on Schroeder’s Mesh Decimation work [9], but with some additional features and changes.

2. The one concerning the local and global error criterion for removing vertices, mainly following the JADE paper [2].

3. The one based on the Surface Simplification using Quadric Error Metrics (QEM) [6], adding the Permission Grid [7], to guide the simplification into generating an approximation with a guaranteed geometric error bound.

2 Tested Solutions

2.1 Mesh decimation

The first implemented algorithm is based on the Mesh Decimation work by Tim Garthwaite and Jason Rapose [5], the theory of Schroeder [9] and the Multiple-Choice paper [10]. The process can be reviewed following the next subsections.

2.1.1 Data structure

A general overview of it may begin thinking about the data structure to deal with the model information. The easiest way to implement the decimation algorithm is to create a ring structure. This consists of a data structure in which triangles know which vertices they use, and vertices know which triangles they are a part of. This leads to a ring structure (see figures 1 an 2) that allows the decimation algorithm to remove vertices and reconstruct the triangle mesh with ease.

Now we have the data structure to work, we go into the classification of the vertices, following Schroeder theory [9], what will make us penalize the removal of vertices in sharp edges and corners. Despite these classification, the algorithm itself, with the distance criterion, makes difficult to delete vertices from edges or corners. So, we manage to preserve some basic geometry of the original mesh.
2.1.2 Multiple-choice technique

While producing the same expected quality of the output meshes, the Multiple-Choice [4] approach leads to a significant speed-up compared to the well-established standard framework for mesh decimation as a greedy optimization scheme. Moreover, Multiple-Choice decimation does not require a global priority queue data structure which reduces the memory overhead and simplifies the algorithmic structure.

Instead of doing greedy optimization (which requires to find the best choice among all candidates) we are using a Multiple-Choice paradigm (which requires to find the best choice only among a small subset of the candidates). The motivation for using this probabilistic optimization strategy is the fact that when decimating high resolution meshes, most of the vertices will be removed anyway - usually 90% to 99% of the original data. Hence it is not necessary to look at all possible candidates in every decimation step. Let us e.g. assume that we decimate a given 3D model down to 5% of the original complexity. The basic idea of Multiple-Choice techniques is to pick a small random subset of candidates, say 8 possible edge collapses, and then perform the best of them. This strategy leads to a wrong decision only in the rare case when all 8 collapses do not belong to the 95% majority of candidates that are supposed to be removed. The probability for a wrong decision is hence \( \left( \frac{5}{100} \right)^8 \approx 10^{-11} \) which implies that a reliable decision if a certain atomic decimation step is part of the optimal decimation sequence or not can be based on a small subset of candidates. Notice that for the above estimate we exploit the fact the most atomic operations are independent from each other and

the exact order of the decimation steps matters only for direct neighbors.

2.1.3 The distance to average plane criteria and removing vertices

The decimation algorithm works by removing vertices from areas of the mesh that are relatively flat. A vertex has a list of vertices called neighbors. Neighbors are defined as other vertices in the entire vertex list that are connected to the vertex by an edge of a triangle. An average plane can be calculated from these neighbors, as shown in figure 3. This is the plane that is closest approximation to all of the neighbors in point-normal form. The point "p" is the average point of the neighbors, and the normal "N" is the average of their normals. "p" is found by averaging each component of the neighbors (x, y, and z); "N" is found by averaging each component of their normals (x, y, and z).

After the average plane has been found, the distance between the removal candidate vertex (from which the neighbors were found, and which the average plane did not take into account) and the plane is calculated. The distance is found using

\[
d = V.N - p.N
\]

where \(V\) is the coordinates of candidate vertex for removal, \(p\) is the coordinates of the projection of this point onto the plane, and \(N\) is the coordinates of the unit normal of the plane. We can see the distance "d" between \(p\) and \(V\) in the figure 4.

2.1.4 Validation Process

Going into details on the validity process, we have three significant issues:

1. illegal movements
2. normal flipping
3. L acoustic rule
Figure 4: The distance between the candidate vertex "V" and its projection on the average plane "p", using plane normal vector.

The illegal movement (see figure 5) occurs when at least one of the two triangles containing the edge collapse are complex, that is, if they are subdivided. We could find this condition and removed all triangles and nodes inside, but this procedure goes out completely from the vertex order selection (distance criteria, local and global error approximation, collapse cost). So we end to disallow the movement totally. It can be seen also as a $180^\circ$ rotation of the normal of one of the faces, so it can be found also by verifynormals function (we include its explanation in the third algorithm).

Figure 5: The original patch, with the illegal movement (collapse of a-b edge, with 10-11-12 triangles conforming on of the adjacent faces), and the new patch (with new coordinate node c).

As we can see, if we substitute node "b" by "a", triangle 9 will change its normal $180^\circ$, so the movement will be rejected. Otherwise, we can calculate a medium new coordinate "c" (we use this in the third solution).

Pair contractions do not necessarily preserve the orientation of the faces in the area of the contraction. For instance, it is possible to contract an edge and cause some neighboring faces to fold over on each other. It is usually best to try to avoid this type of mesh inversion, although the OpenBEM package used for acoustic calculations includes a powerful function to check the orientation of the surfaces. We use essentially the same scheme as others have before. When considering a possible contraction, we compute the angle between normals of old faces and new ones. If the normal flips (rotation of normal greater than a defined threshold), that contraction can be either heavily penalized or disallowed. We also compare normals between adjacent faces, to check for new sharp edge illegal creation.

The L acoustic rule deals with the rule of thumb that says that 6 nodes per wavelength are enough to get accurate results from polygonal meshes [6].

To finish the process, if the vertex is valid to be removed, the program re-triangulates the patch (using the coordinates of the other vertex in the edge collapse for the new node), and obviously two things happen: the model remains almost unchanged and two triangles are removed. The data structure must be updated and the process repeated.

Summarizing, the steps of the algorithm are:

1. Find the best candidate in the model, looking for the one with the minimum distance to the plane.

2. Check if removing the vertex will violate some condition (area, illegal movement, normal flipping, L rule).

3. Re-triangulate: it replaces the best candidate with one of its neighbors.

4. Repeat the process until we met the maximum number of removed nodes or all of them are rejected.

So, the whole loop will become the decimation algorithm.

2.1.5 Some results

The following figures 6 and 7 show the result of the mesh decimation. The geometry used is the HATS dummy head from Brüel and Kjaer.

Figure 6: Most significant nodes after classification, for HRTF BEM calculation purposes.

It can be noticed how the area of nodes that are over-weighted are the most significant in the topology, as the nose and neck. When proceeding to decimate, these geometry tend to be preserved, as the figure 7 shows.

The shape is degraded as the decimation gets larger and the final error increases as the number of nodes gets
The general algorithm flow might be: we have an input

tIME.
write \( v_1 - v_2 \rightarrow v_n \), moves the vertices \( v_1 \) and \( v_2 \) to the new position \( v_n \), connects all their incident edges to \( v_1 \), and deletes the vertex \( v_2 \). Subsequently, any edges or faces which have become degenerate are removed. The effect of a contraction is small and highly localized. In order to select which contraction to perform during a given iteration, we need some notion of the “cost” of a contraction. To define this cost, we attempt to characterize the error at each vertex. To do this, we associate a symmetric \( 4 \times 4 \) matrix \( Q \) with each vertex, and we define the error at vertex \( v \) to be the quadratic form \( \Delta v = v^T Q v \).

Note that the level surface \( \Delta v = \epsilon \), which is the set of all points whose error with respect to \( Q \) is \( \epsilon \), is a quadric surface. Each level surface \( \epsilon \) is an ellipsoid centered around the corresponding vertex. The interpretation of these ellipsoids is that a vertex can be moved anywhere within its ellipsoid and have an error less than \( \epsilon \). The significant feature of the ellipsoids is that they conform to the shape of the model surface very nicely. They are large and flat on mostly planar areas such as the middle of the hind leg, and they are elongated along discontinuity lines such as the contours through the ear and along the bottom of the leg. In some sense, these ellipsoids can be thought of as accumulating information about the shape of the local surface around their vertex.

### 3.2 Permission grid

Our best results have been obtained combining the QEM technique described in the previous section with the permission grid. The permission grid basically follows Steve Zelinka and Michael Garland work [11]. We introduce the permission grid, a spatial occupancy grid which can be used to guide almost any standard polygonal surface simplification algorithm into generating an approximation with a guaranteed geometric error bound. In particular, all points on the approximation are guaranteed to be within some user-specified distance \( \epsilon \) from the original surface.

Conceptually simple, the permission grid defines a volume in which the approximation must lie, and does not permit the underlying simplification algorithm to generate approximations outside the volume. The technique developed here augments current simplification algorithms to provide such a guaranteed error bound on the approximation produced, at a small additional cost in memory and running time.

The general idea of the permission grid is very simple: it is a 3D spatial occupancy grid surrounding the object to be simplified. Each voxel of the grid may be either occupiable or empty. Before any simplification begins, voxels which are entirely within of the mesh are marked as occupiable. As simplification proceeds, each new triangle generated by a particular simplification algorithm (in our case, the surface simplification by QEM) is tested against the permission grid; if a triangle intersects any empty cells, the operation which would have created the triangle is rejected or the grid is expanded.

It can be viewed as a tolerance volume method, according to a permission grid classification. The shape of the volume is a discretization of the union of the set of a-prisms of the mesh. The a-prism of a triangle is the Minkowski sum of the triangle \( A \) and a sphere of radius \( a \) \( (B) \). The Minkowski sum is the whole range covered when we make one of the polygon \( B \) goes sliding along the boundary of the other \( A \). Thus, the permission grid volume is similar to Gueziec’s tolerance volume if all vertex tolerances are equal.

Figure 8: Decimation with local-glogal error estimation.

a) Original 393 nodes model; b) 22 %; c) 39 %; d) 56 %

The figure 9 shows how the geometry of significant parts for HRTF calculations are preserved (ears).

### 4 Summary

The results tell us that the best solution, regarding to the ease of programming and time simulation, is the first mesh decimation algorithm. But if we want to get an increase of quality in the solution, as we can guess from the wrong results we obtain with this method, we must run the second algorithm, the local and global error approximation method. The third solution, QEM, is too expensive according to time and programming. Despite the error approximation numbers, even taking the best solution (permission grid without expansion), the results look...
Figure 9: a) Modified permission grid. Original 7327 nodes model. "Decimation" results: (b) 82% side view; c) 82 back view

better in quality than the rest.

So we could choose the first solution and then apply the third to the result, to improve the figure topologically. Anyway, if the priority is not the time simulation, we can use the fourth solution, the "decimation", combining local and global error approximation, QEM calculations and permission grid.

Apart from the decision of choosing a specific algorithm, we can guess some particular results, mainly looking at table 12. One result we get is that the simulation time for QEM solution gets lower comparing to the local and global error approximation solution, as the number of nodes in the initial mesh is greater. That is the result of computing the error for all the nodes at each iteration of the process, when the QEM solution uses a multiple-choice set of candidates.

Considering the deepness of the decimation, with almost all the algorithms, and according to the validation criteria applied, we can remove 60% of the original mesh approximately. We increase that number using the "decimation" algorithm, that goes up to 90%.

5 Acknowledgements

This work has been developed during the the project ref. TIC2003-06841-C02-01, "MPEG-4 advanced audio transmission/reception platform", supported by the Spanish ministry for science and education.

References


