International Journal of Computational Vision and Biomechanics
Vol.1 No. 2 (July-December, 2015)

MUK PUBLICATIONS
Open Access Publisher

Shape-preserving Mesh Simplification based on Curvature Measures from the Polyhedral Gauss Map

G. Echeverria & L. Alboul

Materials and Engineering Research Institute, Shefeld Hallam University, Shefeld, S1 1WB, UK

This paper presents an improvement on methods for simplification of triangular meshes, based on discrete curvature measures. The basic tool is the Polyhedral Gauss Map *(PGM), which is computed directly from a mesh and provides a detailed description of curvature associated with each individual vertex. Taking into consideration this description, we determine the* Total Absolute Curvature*, abbreviated as TAC, for each vertex. The TAC measure is more precise than the discrete analogue of the Gaussian curvature obtained using the Angle Deficit (AD) method, as it reflects the complexities in the surface shape around a vertex. We apply the TAC to attach a* relevance weight *to each vertex in a mesh and select the vertices to be decimated. The weight of a vertex is also based on measurements of its neighbourhood. To this end we introduce the concept of weighted total absolute curvature (WTAC). The TAC is also used in an additional step to the decimation algorithm to optimise the curvature of the simplified mesh. Taking into consideration the theoretical background and numerical experiments, we can also conclude that the TAC decimation algorithm is* shape-preserving.

Keywords: *Mesh decimation, Curvature, Polyhedral Gauss Map, Spherical geometry*

1. INTRODUCTION

The most common representation for an object in computer aided applications, specially in 3D graphics, is still the polygonal mesh, where a collection of points in space are joined together by edges and faces to form a shape. This provides great flexibility in representing a variety of shapes, from simple to very complex ones, and in varying degrees of detail. Modern methods for generating 3D objects involve high definition scanning of data, providing very detailed meshes of the target objects. Often this produces an enormous amount of data, which is difficult to handle by current computer systems for applications in real time. Meanwhile, the data acquired with these methods may be redundant or excessive to portray the basic shape of the object.

For these reasons, it is desirable to simplify the polygonal geometry in a model. *Mesh simplification* is the process of decreasing the number of components from a polygonal mesh, reducing its overall combinatorial and geometric complexity, while at the same time still providing a good visual representation of the original object. The necessity to simplify a model may come from limited resources, either storage, transmission bandwidth, processing power or display in real time. All of these constraints benefit from having a smaller mesh that still appropriately represents the original object.

Several different techniques already exist to simplify a polygonal mesh. The approaches taken vary greatly from one method to another, however all of them must identify which parts of the object are important to the shape, and which can be safely removed. The curvature of an object is a good measure of the behaviour of the shape, and thus an important characteristic to consider when modifying a mesh model. Curvature can be calculated for the whole object, specific regions, or individual vertices or edges. The curvature of a vertex is a measure of how far its neighbourhood is 'pulled away' from a plane. It is a good estimate of how relevant is the vertex for the general shape of the object.

The *Angle Deficit* (AD), known also as the *discrete Gaussian curvature*, is regularly used as the measure of the curvature associated with a vertex. However, it does not fully reflect the local shape structure around the vertex and therefore is not fully suited for decimation based on curvature.

In this paper we present a new decimation method based on the estimation of the *Total Absolute Curvature* (abbreviated as TAC) as a better measure to guide mesh simplification methods. The TAC is obtained for each individual vertex using the *Polyhedral Gauss Map* (PGM). The Polyhedral Gauss Map for a vertex consists of a set of spherical polygons, which areas are each equipped with a '+' or '-' sign and correspond to the positive and negative parts of the curvature concentrated at this vertex. Therefore it permits the full characterisation of the curvature at a vertex and identification of hidden features not recognised by other methods. Thus the TAC, computed by summating the absolute values of curvatures parts, is ideal for correctly determining the importance of a vertex for the shape of the mesh. The paper (Alboul and Echeverria 2005) provides a detailed theoretical background of both the AD and TAC curvature measures and provides more references. The method to compute and visualise the Total Absolute Curvature for complex polyhedral surfaces is also presented in more detail there.

We present a series of weight values for the vertices of a polygonal mesh to specify how relevant each vertex is to

the overall shape of the object. All of these weights are based on the TAC and in most cases also include the area estimation around the vertex. These measures weigh the complexity of the vertex with the visual impact it has on the overall surface of the object.We present the reasoning behind the selection of four different weights, and compare the results obtained from using them to guide the simplification process.

We also use the TAC to optimise the corresponding region in the simplified mesh after a vertex removal by using edge flip technique, first presented in (Lawson 1972) and later applied in (Alboul 2003). This procedure ensures a further simplification of the mesh with respect to curvature, since the higher TAC yields more complex mesh geometry. As the TAC of a mesh region is computed by summating the TAC of the vertices belonging to this region, our method can be considered also as *shape-preserving*. It emphasises domains of the most prominent curvature of the region, that is either positive or negative, while decreasing or completely eliminating curvature domains of the opposite sign.

We also compare the results obtained from using either the common AD method or the new TAC. We show how this new curvature calculation improves the obtained results when using the AD to estimate the curvature.We produce simplified meshes using both measures, and the results are compared numerically against the original models to draw a conclusion. Figure 1 shows an example of decimation using the TAC.

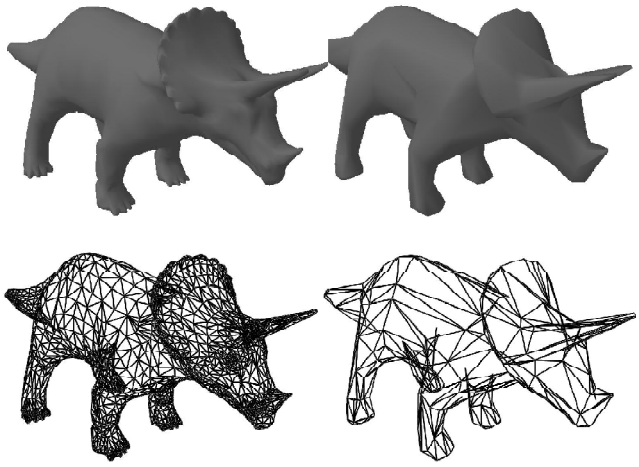


Figure 1: ***Triceratops*** **model: shaded (top), wireframe (bottom). Left: original (2,832 vertices); Right: 88.3% decimated (332 vertices)**

The measurement of vertex weights presented here can be applied to various mesh simplification algorithms based on decimation or edge collapse to improve their selection of the vertices to simplify. To demonstrate the advantages of using the TAC in a simplification algorithm we have implemented a simple vertex decimation program to test and compare the results obtained on different models.

The main contribution is the application of the TAC measure to a mesh simplification algorithm, proving its advantages over decimation algorithms based on existing curvature measurement methods. Other novel contribution is the procedure of updating the mesh after a vertex removal, which is based on the minimisation of the TAC. The research in this paper has been partially presented in (Echeverria and Alboul 2006).

The structure of the document is as follows: Section 2 presents the existing research in the field of mesh simplification. Section 3 presents the method used to compute the TAC of a vertex based on its PGM. Section 4 shows the selection of the vertex weight measures and compares them. The details of our decimation program are presented in Section 5, followed by experimental results in Section 6. Section 7 presents the conclusions and future research directions are pointed at in Section 8.

2. PREVIOUS RESEARCH ON MESH SIMPLIFICATION

Extensive research has already been done in the field. There are several different approaches to simplifying a polygonal mesh. The papers (Cignoni *et al*. 1998 – Comparison) and (Luebke 2001) review the most important of these methods, and compare their advantages and shortcomings. Cignoni *et al*. do a comparison of the error differences, from the original models to the simplified ones, and the time taken to obtain the results, using several different algorithms. To do this they created the *Metro* tool (Cignoni *et al*. 1998 - Metro), which has since been used by others to compare any new algorithm with the existing ones. As explained in these papers, simplification techniques vary in: *optimisation goal*; *local* or *global optimisation*; *preservation of the original object's topology*; *maintaining the original vertices* or *re-meshing the model*. Some simplification methods also perform *view dependant* decimation. That is, a model is more decimated in areas that are far away or hidden from the current perspective. This permits a larger reduction of the mesh, but requires that the mesh be re-simplied if the view direction changes. The algorithms tested in the aforementioned works include Mesh Decimation, Simplification Envelopes, Multiresolution Decimation, Mesh Optimization, Progressive Meshes and Quadric Error Metric Simplication. (We refer the reader to the cited works for the precise descriptions of these methods). In the tests, Mesh Decimation produces the largest error with respect to the original model, although it is by far the fastest algorithm. Other techniques perform better in most respects, in particular Quadric Error Matrix (QEM) (Garland and Heckbert 1997), which is generally considered as a very effective simplification method and used as a parameter of comparison for newer techniques.

The first method for *decimation* was proposed by (Schroeder *et al*. 1992), and is based on progressive removal of specific vertices from a mesh. All vertices are classified according to their local topology, and are handled accordingly. Vertices are labelled as 'simple', 'complex' or 'boundary'. Complex ones are generally non-manifold

vertices, and are left untouched. Simple and boundary vertices are treated differently. Simple vertices are further classified according to the number of feature edges connected to the vertex. Vertices are selected for decimation according to a distance metric. For simple vertices the parameter is the distance from an average plane; for border vertices, it is the distance to the line connecting the neighbours along the border. The vertex with the shortest distance is selected for removal, along with the triangles surrounding it. The hole produced in the mesh is filled using a recursive splitting of the remaining region into triangles. This method generates a subset of the original vertices, not adding any new ones. It also preserves the topology of the object.

A scheme to re-tile a polygonal mesh with less vertices was proposed by (Turk 1992). First, a new set of vertices is distributed over the original model, the new vertices will repel one another to adequately cover the whole surface. Next the surface is re-triangulated using both new and old vertices to preserve the shape, and later the old vertices are removed. An extra step is also included which uses an estimation of the curvature to aid in the distribution of the new vertices over the surface.

A method to preserve more accurately the appearance of a simplified model is developed in (Cohen *et al*. 1998), by using texture and normal maps, in addition to the polygonal mesh. They initially compute the shading colours and the normals of the full model, and convert this data into maps that contain this important visual information. Then the mesh can be simplified to several levels of detail. Applying the maps with the original information will make the model look very similar to the full detail version. The main improvement for this method is the *texture deviation metric* which is used to assign texture coordinates to the remaining vertices in their corresponding position with respect to previous vertices. This metric can also be used to guide the simplification algorithm.

(Lindstrom and Turk 1998) implemented an algorithm using edge collapses where the position of new vertices is obtained from the optimisation of a few simple geometrical properties of the local neighbourhood of the edge. They use preservation and optimisation of the volume and boundary related to the edge. The same optimisations are used to give weights to the edges for the selection of the collapses.

A probabilistic approach is employed by (Wu and Kobbelt 2002) to reduce resource requirements of a simplification algorithm. They use a Multiple-Choice Algorithm to randomly select a few candidate edges and select from those the best option. This avoids having to keep an updated queue of the best possible options at all times. Doing so they significantly speed up the simplification process and reduce the memory requirements, while obtaining a good degree of simplification, comparable to the QEM method.

In (Kim *et al*. 2002) a measure based on curvature is used to assign a cost to the edges, and to select the ones to be collapsed. The authors make use of both the Gaussian and mean curvatures on the mesh. After deciding on an edge to collapse, a new vertex is generated in place of the two edge vertices. The location of this new vertex is found using a butterfly subdivision mask. Using curvature to decide on the geometry to eliminate, they prove that important features of the object are preserved after heavy simplification.

(Hussain *et al*. 2004) propose a simplification method driven by half-edge collapses that keeps at least one of the vertices of the edge removed. They use a metric based on the angle difference from the original faces to the ones that will be created after the collapse; which is effectively an approximation of the curvature of the region, although not very accurate, but nevertheless useful in preserving important geometry. This implementation competes in performance with QEM but claims to require less memory to store data.

3. DESCRIPTION OF THE POLYHEDRAL GAUSS MAP

In many applications only the Gaussian curvature of a vertex is measured, using the Angle Deficit method. This technique is only capable of identifying two kinds of vertices: convex or saddle. Vertices which do not fit in these categories will present problems to the algorithm, and their true curvature cannot be found. This is the case, for example, when a non-saddle vertex has concavities (folds) in its neighbourhood.

The new method presented to compute the curvature of a polyhedral surface is based on the area of the Gauss Map constructed for a vertex. It is an analogous technique to compute the integral curvature on smooth surfaces (Banchoff 1970). The curvature of a whole mesh is computed as the sum of the curvatures of the individual vertices that belongs to the mesh.

In what follows we assume that a mesh M represents an orientable surface, and therefore we can determine a coherent orientation on the whole mesh (Alboul 2003). Then the direction of the outwards pointing normals is chosen to be positive. The edge will be called *convex* if the two lines determined by the normals to its adjacent faces (triangles) intersect in the negative direction; otherwise it is called *concave*.

If V is the set of all the vertices in the mesh M, for every vertex $v \in V$, we define the *star of the vertex* as the set of faces incident on v, ordered in counter clockwise direction with respect to the chosen orientation, and denote it as $star(v)$. The Polyhedral Gauss Map of v is computed using the normal vectors of the faces in $star(v)$. These vectors are translated to the same origin, which will become the centre of a sphere. The endpoints of the vectors are joined by geodesic arcs in their corresponding order, creating what is called the *spherical indicatrix*, shown in Figure 2. The ordering of the vectors from their corresponding faces is used to determine the sign of the curvature.

If the star of the vertex is shaped as a convex cone, *i.e.* all its edges either convex or concave, or a simple saddle or its generalised version, the arcs in the spherical indicatrix

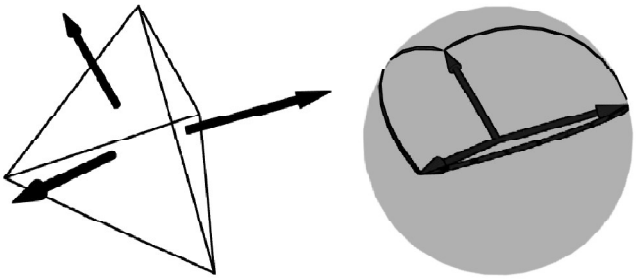
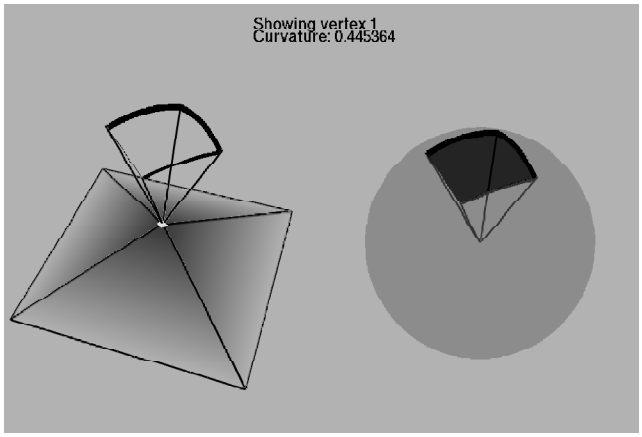
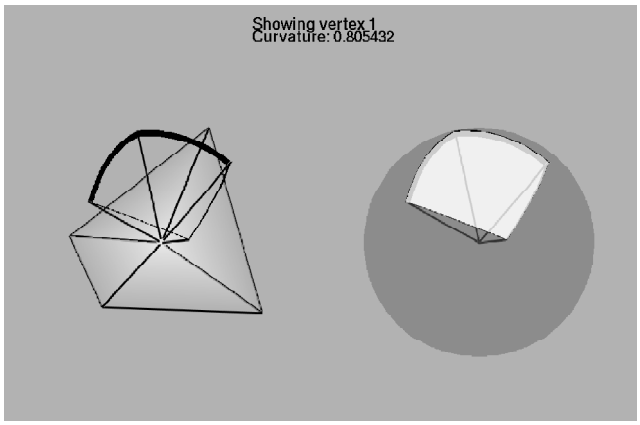


Figure 2: Using the normal vectors of faces in star(v) to create the Gauss Map

will not intersect and will draw a single spherical polygon on the surface of the sphere. Under a simple saddle we understand a saddle vertex whose star consists of two concave and two convex edges in alternated order. Recall that a saddle vertex does not posses a supporting plane. A generalised simple saddle is a saddle where a concave or convex edge are substituted by several edges of the same type. A convex and a generalised simple saddle can be seen in Figure 3.



(a) Convex cone



(b) Saddle

Figure 3: Vertices with intersections in their spherical indicatrix

In all other cases, there will be self-intersections in the indicatrix. It is necessary to identify where these intersections occur and separate the indicatrix into individual spherical polygons. It is possible for two or more spherical polygons to overlap. Figure 4 shows a vertex whose star is a non-convex cone. In this case, the concavity causes the normals of the faces to switch directions temporarily and go clockwise with respect to the vertex, opposite to the direction of their corresponding faces. When this occurs, two of the arcs intersect, signifying a change in the sign of the curvature. The intersection point is where two separate spherical polygons of opposite signs meet. The first step in the Polyhedral Gauss Map computation consists of identifying all such spherical polygons.

The next step is to determine the orientation, positive or negative, of the spherical polygons. Each one of them splits the surface of the sphere into two areas, only one of which corresponds to the vertex curvature. The correct area chosen is based on the ordering of the faces and the corresponding normals. Positive polygons result from the ordering of the normal vectors which complies to the ordering of their corresponding faces. If the ordering of the normals goes in the opposite direction, then the area of the polygon represents a negative curvature. The Total Absolute Curvature is obtained by adding together the absolute values of the areas of both positive and negative polygons.

In the visualisation of the Gauss Map shown in Figure 4 the areas of positive curvature are shown in black, while negative areas are shown in white. On the mesh each vertex is also given a colour based on its curvature: it will be black if it has only positive curvature, white if it has only negative curvature and grey if its Gauss Map has spherical polygons of both positive and negative orientation.

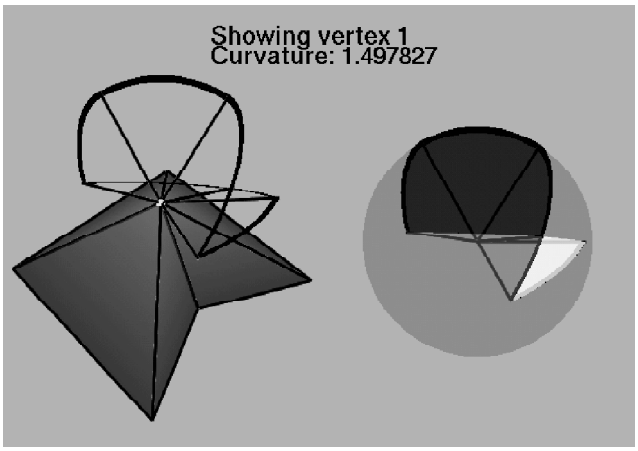
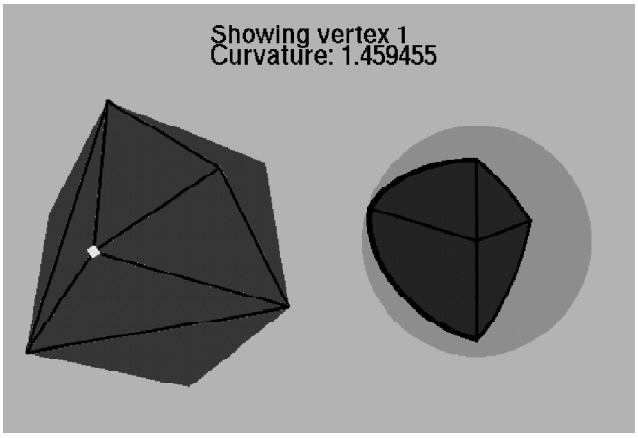


Figure 4: A vertex with non-convex neighbourhood (showing the indicatrix) and its corresponding Gauss Map

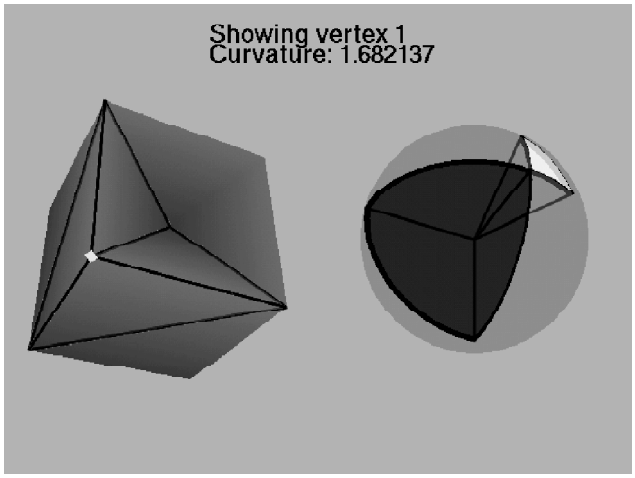
The algorithm is fast as all computations are nearly linear with respect to the number of vertices in the mesh. All relevant information about each vertex is gathered simultaneously with the computation of the Polyhedral Gauss Map: mean and Gaussian curvature (using Angle Deficit), the number of neighbours and incident triangles in *star*(v). All of this information is stored in linked data structures for its later use.

The PGM algorithm can also handle non-manifold vertices, correctly identifying the curvature of vertices with self intersecting faces. These vertices will produce larger than normal areas in the Gauss Map. The Polyhedral Gauss Map obtained gives a complete characterisation of the vertex. All irregularities of the mesh can be determined based on the Gauss Map technique proposed, without resorting to computationally expensive methods.

The PGM provides a better understanding of the geometry of an object than the Gaussian curvature. It can identify more detailed features of the shape such as hidden regions of negative curvature that exist in the concavities of a cone and appear only when analysing both the positive and negative components of the curvature of the vertices. The Gauss Maps for two different vertices are shown in Figure 5. While Angle Decit methods would conclude that the two vertices are equal, using the PGM it is immediately evident that both vertices have different curvature.



(a) Convex object



(b) Nonñconvex object

Figure 5: Gauss Map of a vertex in two different objects. Left: convex vertex (only positive curvature). Right: non-convex vertex (positive and negative curvature components)

The PGM of a vertex represents, therefore, a set of spherical polygons equipped with a positive or negative sign. The algebraic sum of these polygons is equal to the *discrete Gaussian curvature*, or *Angle Decit*, of the vertex; and the sum of the absolute areas represents the measure of *Total Absolute Curvature* associated with this vertex. Figure 6 shows a vertex whose PGM has various areas both positive and negative.

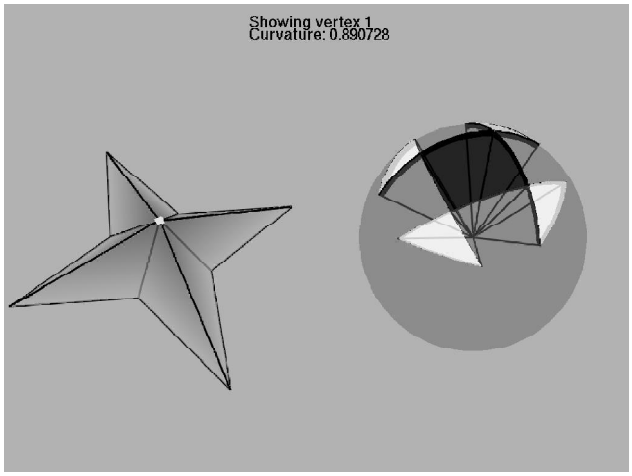


Figure 6: A vertex with the Gauss Map consisting of several spherical polygons of positive and negative signs

4. CURVATURE AS DECIMATION MEASURE

Most mesh simplification algorithms based on vertex decimation assign a weight to each individual vertex, further referred to as the *relevance weight*, that signifies its importance to the shape of the object. If the value is small the vertex can be removed without significantly altering the mesh, while if the weight is larger the vertex must be kept.

The values used as weights for the vertices are different for each implementation, but generally are based on the geometrical properties of the surrounding region, such as distances, areas or measures of curvature. We present a set of new vertex weight measurements to guide decimation based on the Total Absolute Curvature. We refer to this measurement set as the *Weighted Total Absolute Curvature*, denoted as WTAC. The WTAC measures consist of multiplying the TAC by some factor related to some geometric shape parameter associated with each vertex. Four different weight measures are tested and compared in our decimation program. We first present definitions of shape parameters used in the WTAC.

For any vertex v, as well as the star *star*(v), we define *neighbours*(v) as the set of vertices in *star*(v) other than v and *link*(v) as the collection of the n edges joining the vertices in neighbours(v), which forms a polygonal closed line in space. $\mathbf{N}_v$ is an artificial vector, computed as the average of the sum of normals and areas of the incident triangles. It can be considered to be the normal to the vertex.

Then, the *projected link* of v is the polygon P obtained by orthogonally projecting the list of vertices in *neighbours*(v)

on a plane that passes through v and is perpendicular to the vector $\mathbf{N}_v$. The projected link can also be defined for saddle-like vertices; in this case the vertices will be projected from both half-spaces determined by the plane.

We define T_v as the list of the n triangles in *star*(v), so that $t_i \in T_v$, then the *cone area* of vertex v is the sum of the areas of all the triangles in star(v), and is denoted as A_C(v). Thus,

$$A_C(\nu) = \sum_{i=1}^{n} area(t_i).$$

The *projected area* of vertex v is computed as the planar area of the polygon P_v, and is denoted as A_P(v). It is possible, however, that the projection of the link into the plane chosen may produce self-intersections, specially when dealing with very complex saddle type vertices. In these cases the area computed will be unreliable. This problem is recognised in (Lee *et al*. 1998), and an alternative projection into a conformal map is proposed.

In order to overcome the aforementioned difficulties and shortcomings related to the determination of the flat projection, we also use the well-known isoperimetric inequality (Osserman 1978) $L^2 = 4\pi \geq A$ (where L is the length of the figure in the plane, and A is its area) in order to get an approximated area of the flat projection. We define E_v as the list of the n edges in *link*(v), so that $e_i \in E_v$, then by denoting with *length*(e_i) the sum of the lengths of edges, we determine the shape parameter *length-area* of vertex by using the formula:

$$A_L(\nu) = \frac{\sum_{i=1}^{n} length(e_i)^2}{4\pi}.$$

Using these definitions of the areas around a vertex, the four versions of WTAC are as follows:

4.1 TAC

In this case we use exclusively the Total Absolute Curvature of the vertex v as its relevance weight, denoted as TAC(v). All other properties of the vertex are discarded, and only the curvature has an effect on its importance.

4.2 ATAC

Multiplication of the TAC by the cone area around vertex v.

$$ATAC(\nu) = TAC(\nu) \times A_C(\nu)$$

This measure takes into account the complexity of the vertex and its visual importance in terms of the area and curvature simultaneously. It gives more priority to vertices with large incident triangles making them less susceptible to elimination, but those with small stars are more likely to be removed.

4.3 PTAC

Knowing the cone and projected areas we can use a *normalised area* of v as the parameter to multiply TAC:

$$PTAC(\nu) = TAC(\nu) \times \frac{A_P(\nu)}{A_C(\nu)}.$$

An important feature of this parameter is that the values of factor will always lie in the half-open interval (0, 1]. It is clear that if the star of a vertex represents a flat region, both the cone and projected areas will be the same, making the factor 1. If the vertex has a very sharp cone, the projected area will be much smaller than the cone area, making the factor close to 0 (zero).

The PTAC weight thus considers the local properties of the neighbourhood around the vertex, but disregards the size of the region. It ensures that vertices with similar stars at different scales will be treated in the same way. This may result in the removal of a vertex in a relatively flat region that leaves a very large hole in the mesh. It can be difficult to properly re-triangulate such a hole and may lead to a more expensive optimisation of the initial triangulation.

As mentioned before the computation of the projected area is not always reliable, depending on the plane chosen for the projection of the vertices.

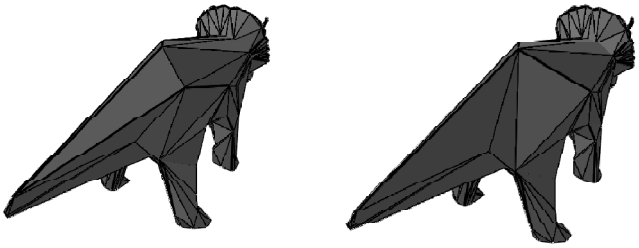
4.4 LTAC

Using the projected area and the length-area inequality, we establish an alternate version of the normalised area.

$$LTAC(\nu) = TAC(\nu) \times \frac{A_L(\nu)}{A_C(\nu)}.$$

This weight measure attempts to solve the shortcomings of PTAC, by computing a planar area by means of the perimeter of the star instead of the projection into a plane. This avoids any self intersections in the projection. In this case the factor is not guaranteed to lie in the interval (0, 1]. However this weight also disregards the scale of the neighbourhood of the vertex.

The *Metro* tool was also used to compare the different weights used to guide the decimation. The results from this comparison, both visually and numerically indicate that the best results are produced by using the ATAC weight. Using normalised areas, as in PTAC and LTAC, preserves better finer details in the mesh, but produces much larger errors on areas of small curvature. This is due to the fact that areas of relatively small curvature will eventually be reduced to a single vertex with small curvature and several very large incident triangles. This single vertex represents the shape previously represented by the whole region. If it is removed, the appearance of the object is altered signicantly. This is exemplified in Figure 7.



(a) Vertex with many large incident faces **(b) Removal of the vertex changes the shape**

Figure 7: *Triceratops* model simplified with LTAC to 84.4%.

The graphs in Figure 8 show a comparison of the *Metro* results. They plot the Hausdorff distances obtained when comparing the original model with different decimated meshes using the various vertex weights. The *Decimation %* parameter shown refers throughout this document to the percentage of the original vertices removed during the decimation.

5. IMPLEMENTATION OF A VERTEX DECIMATION ALGORITHM

For the purpose of testing the curvature measure on simplification, a simple vertex decimation algorithm is used, similar to the one described in (Schroeder *et al*. 1992).

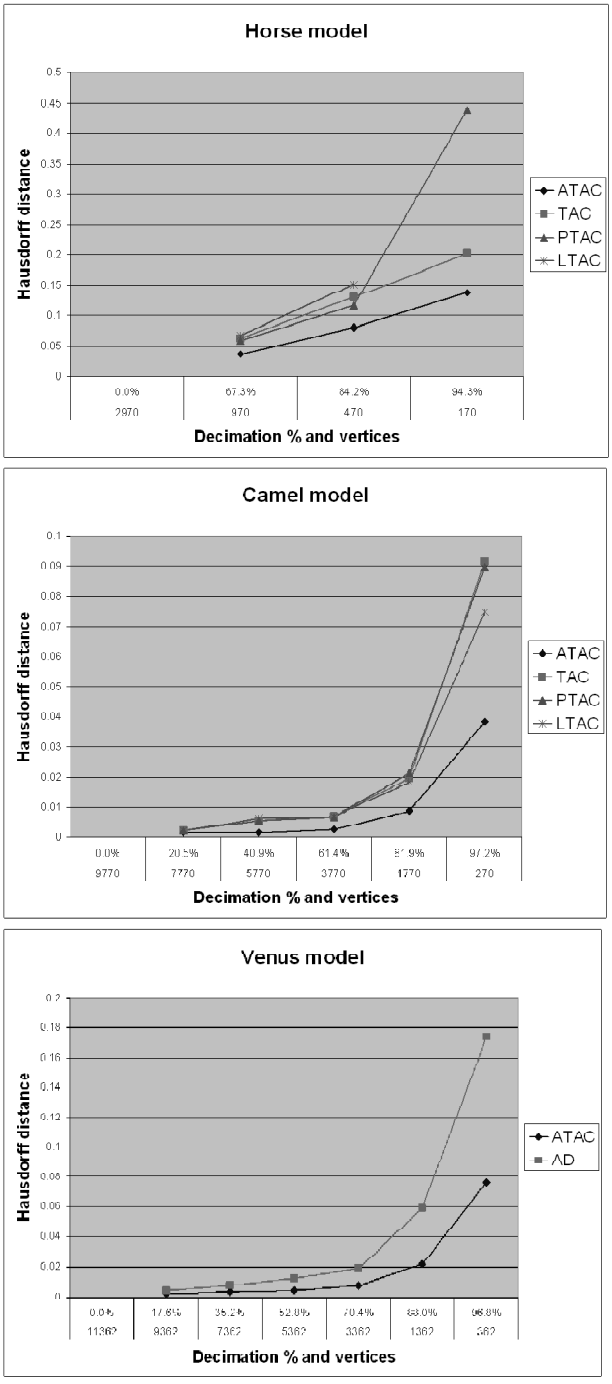


Figure 8: Comparison of the vertex weight parameters: TAC, ATAC, PTAC and LTAC

The algorithm makes several passes over the whole dataset, removing each time the vertex with the smallest weight. After removing the vertex it updates the mesh by re-triangulating the obtained 'hole' in the mesh in such a way that the curvature of the mesh does not increase. In order to achieve this we apply to the initial re-triangulation the optimisation procedure based on the minimisation of the TAC by means of the edge flip as explained in (Alboul *et al*. 1999) and (Aichholzer *et al*. 2002). This preserves topology as well as the shape up until very high levels of decimation.

The process of vertex decimation is as follows:

- The polygonal mesh to be decimated is analysed, and lists are created for the vertices and faces. The lists are then converted into arrays that make reference to each other by their indices. This simplies the search for specific vertices or triangles.

  Initially, the TAC is computed for all vertices in the mesh, obtaining along the way all other required information for each vertex: the Angle Deficit, list of triangles in the star, list of neighbours, artificial vector normal to the vertex, projection of the neighbours and feature edges. All of these are stored in an array of data structures.

- After gathering all the above-mentioned information, an extra parameter is assigned to each vertex: the relevance weight, which is described in general as the WTAC. All vertices are sorted by increasing value of WTAC, to simplify the process of selecting the next vertex for decimation. To this end, two new arrays are created. One of them has the vertices ordered by WTAC, and includes the vertex ID number and the value assigned to it. The other array is ordered by vertex ID numbers, and contains the index location of each vertex in the other array. This second array is used to simplify searching for other vertices. Vertices to delete are taken from the top of the array, and after each removal the array is updated with the new values for the affected neighbour vertices.

- A vertex v is selected for removal if it has the smallest value for the WTAC measure being used. For the special case of vertices on the boundary of the surface of an open object, the parameter used for selection also considers the Angle Deficit, since the Gauss Map area in these vertices can be equal to zero.

- The region around the removed vertex is updated using the technique of half–edge collapse. A vertex v_c is chosen from the neighbours of the deleted vertex v. Only the two triangles that share the edge $\overline{vv_c}$ are deleted from the mesh, and the rest of the triangles incident on v are updated to be now incident on v_c. This method is likely to create some degeneracies, mainly in the form of adjacent triangles facing opposite directions but lying in the same plane. The program will check that two identical faces are not facing in opposite directions.

There is no check in the case of non-identical but overlapping triangles, however this case is taken care of by optimising the triangulation.

- Once the initial re–triangulation is complete, the curvature data for all the neighbour vertices v_j is recalculated using the new triangles, and afterwards the triangulation is optimised using edge flips. The objective of this optimisation step is to reduce the curvature of the proposed triangulation. A list of edges is created only for the triangles that now cover the previous *star*(v), the edges on the borders of the polygon are then discarded and only the edges lying inside of the polygon are kept. Each edge structure contains the vertices that define it, the two neighbour triangles and the two opposite vertices. All of the edges are tested by flipping them and recomputing the curvature of the 4 vertices involved. If the sum of curvatures is smaller than previously the flip is accepted, otherwise the edge and incident triangles and vertices are returned to the previous state. This process is repeated until none of the edges will flip to a smaller curvature configuration.

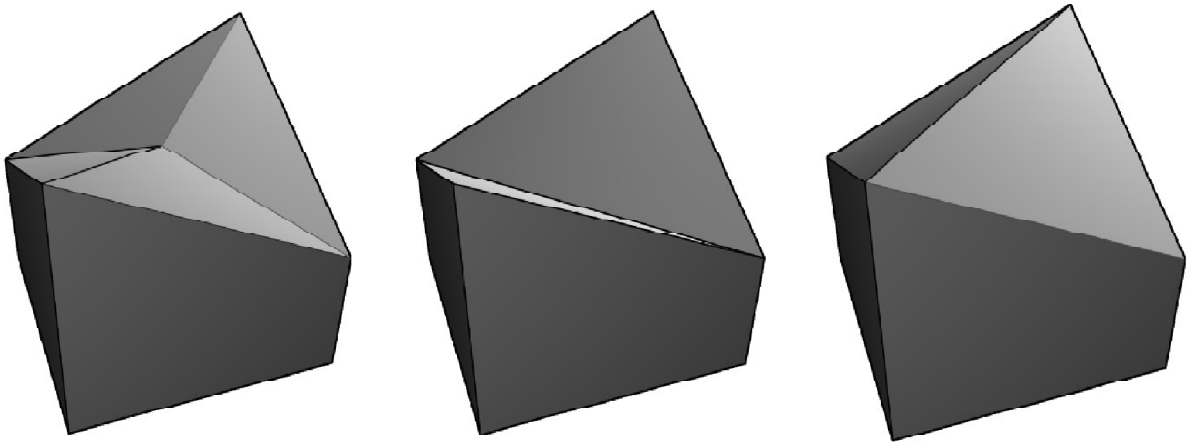
Figure 9 shows an example of the removal of a vertex and how edge flipping can reduce the curvature of the resulting object. In the figure the original mesh is shown to the left, followed by two possible re–triangulations after the removal of a vertex. Regardless of what the initial triangulation may be, both results are tested using edge flips, and ultimately the program will select the third mesh because of the smaller curvature.

Once a valid triangulation has been obtained, for every vertex v_j the curvature and areas are updated according to the new mesh. The decimation procedure then continues until a termination condition is met. Otherwise, the algorithm continues until the shape of the object is completely lost. For an input mesh of genus 0, the algorithm can continue until the object is simplified to a tetrahedron. Termination condition for the decimation can be selected according to the requirements of the final mesh. It can be set as the maximum value of WTAC of the vertices to be deleted. Since all vertices are sorted by their WTAC value, once a certain threshold is reached, the decimation stops. Alternatively, the termination condition can be a maximum error difference between original and decimated model, to avoid losing detail in the mesh. If the requirement is to reach a certain file size for the model, the limit can be set to a decimation percent or a specified number of faces/vertices in the final model.

6. EXPERIMENTAL RESULTS

Figure 10 shows the results of using both curvature measures to decimate a polygonal mesh. The top row has the original mesh, while the centre row has the mesh decimated using the TAC and the mesh in the bottom row was obtained using the AD. In the right column of images all vertices are colour coded according to the value of the TAC, shown in black for positive curvature, white for negative, and grey for mixed (both positive and negative) curvature. Regions of similar curvature can be distinguished where several vertices of the same kind are linked together. It is noticeable that the model decimated using the AD possesses more regions of mixed curvature (grey colour), which represents irregular geometry. The mesh decimated with the use of the TAC has more clearly defined areas of either positive or negative curvature that mimic those in the original mesh. This produces a more smooth and visually pleasant surface.

We evaluate the benefit of using the TAC instead of the AD to guide a simplification algorithm. The same decimation algorithm was applied to various models using both curvature measures to assign the importance weight to vertices and to select optimal edge flips. In both cases the formula to compute the weight for the vertices is the ATAC, but by substituting the AD with the TAC and multiplying by the cone area $A_C(v)$. The resulting simplfiied models were compared using the *Metro* tool. Figure 11 shows the results of using the decimation program on three different meshes using both curvature measures.



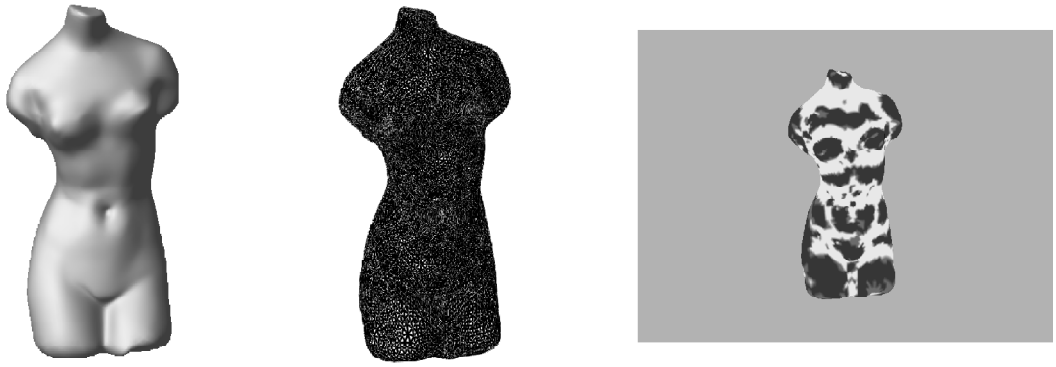
(a) Original mesh TAC = 13.817166 **(b) Before flip TAC = 14.177234** **(c) After flip TAC = 12.566371**

Figure 9: Comparison of the triangulations obtained after removal of a saddle vertex

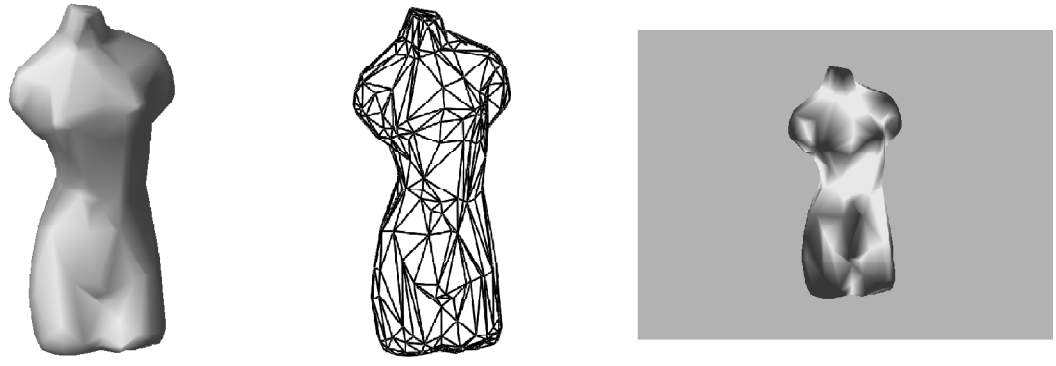
As can be seen from the results, the use of TAC instead of AD as an estimation of curvature produces a smaller error, and the difference in performance increases as the decimation goes further. We have not done tests to compare the time difference to do similar levels of decimation using both measures. Computing the TAC is more complex than Angle Deficit, but it still has almost linear complexity, once all the information required has been extracted from the model and stored in appropriate data structures. Currently our application computes both the TAC and AD simultaneously, and thus the time taken is the same in both cases.

During decimation the geometric properties of the mesh are altered in differing ways. Table 1 and Table 2 show the values of the curvature as computed with the TAC and AD, total surface area of the object and Hausdorff distance to the original mesh. TAC and area both decrease with the loss of geometry, but the change in TAC is more dramatic, because it is more sensitive to the changes in the geometry of the object. On the other hand, the Gaussian curvature of the whole mesh, as represented by the AD remains the same, at the value equal to 4π for closed meshes of genus 0.

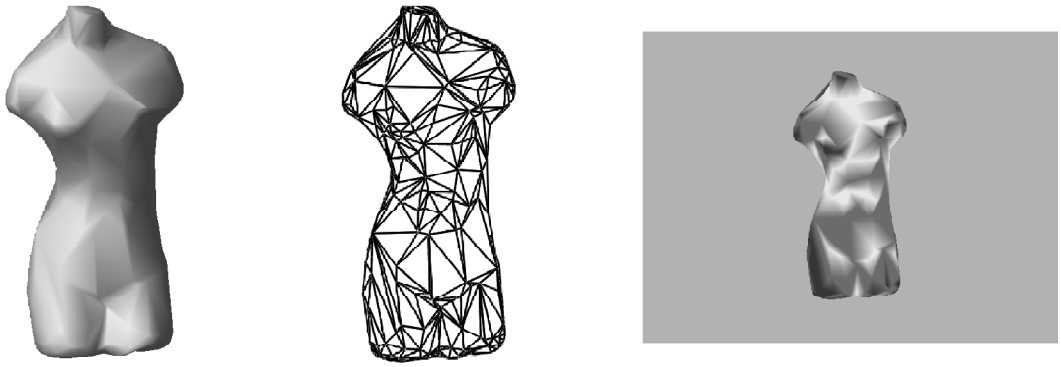
Some of the existing mesh simplification algorithms produce a mesh with evenly distributed triangles around the final model, particularly those that re–sample the mesh and use new vertices. This presents some advantages, specially for more even texture mapping. However, some of the finer detail is lost from the original shape. The use of curvature to drive a simplification algorithm ensures that the reduced mesh will have more triangles where the shape changes more rapidly, and less triangles in flatter areas. As an example, we use the rockerarm model in Figure 12. As can be seen in the close-ups of the decimated rockerarm in Figure 13 the finer detail of the model is preserved even after several steps of decimation.



(a) Original (11,362 vertices)

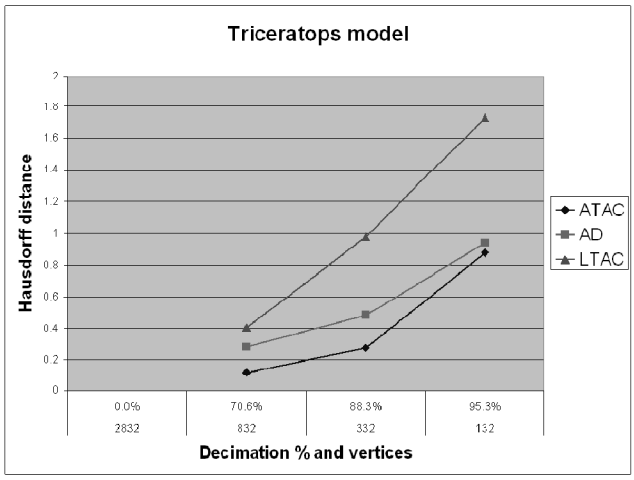


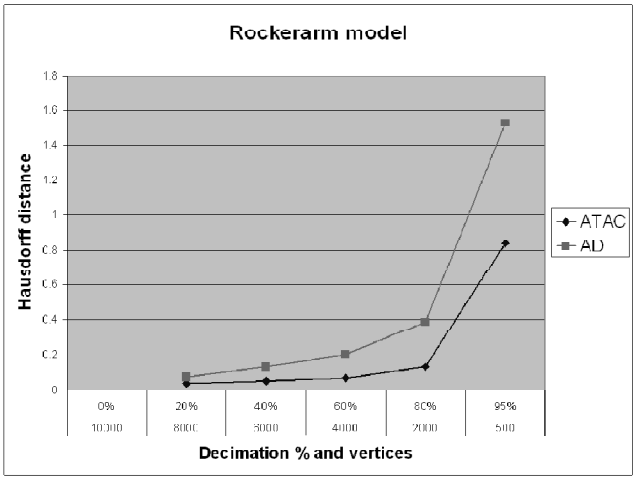
(b) 96.8 % decimation with PGM (362 vertices)



(c) 96.8 % decimation with AD (362 vertices)

Figure 10: ***Venus*** **model: shaded (left), wireframe (centre), curvature coded (right)**





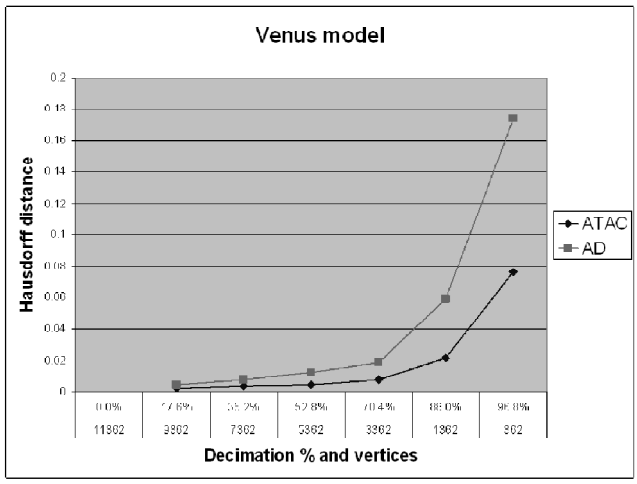


Figure 11: Comparison of the results obtained using TAC or AD as a measure for curvature in decimation

Table 1
Comparison of the Curvature of the *venus* Model with Increasing Decimation

Decimation	*Vertices*	*TAC*	*AD*	*Surface Area*	*Hausdorff*
0%	11,362	77.045602	12.566380	30.136994	0
17.60%	9,362	69.976640	12.566385	30.136968	0.002301
35.21%	7,362	68.334118	12.566394	30.136956	0.003741
52.81%	5,362	66.487840	12.566390	30.136331	0.004303
70.41%	3,362	63.640547	12.566383	30.136502	0.007639
88.01%	1,362	57.489905	12.566375	30.123395	0.021712
96.81%	362	47.290207	12.566365	30.005538	0.076089

Table 2
Comparison of the Curvature of the *igea* Model with Increasing Decimation

Decimation	*Vertices*	*TAC*	*AD*	*Surface Area*	*Hausdorff*
0%	33,587	730.119954	12.566346	23505.330791	0
11.91%	29,587	706.423968	12.566354	23505.398398	0.108923
23.82%	25,587	703.362744	12.566344	23504.541003	0.153921
35.73%	21,587	652.604403	12.566359	23503.028226	0.164336
47.64%	17,587	587.653313	12.566370	23501.645275	0.164336
59.55%	13,587	513.781304	12.566373	23500.950160	0.242121
71.46%	9,587	442.974448	12.566368	23499.417764	0.247055
83.37%	5,587	340.123710	12.566366	23498.766032	0.390726
95.27%	1,587	175.379923	12.566375	23450.253691	1.292738
98.25%	587	71.259997	12.566376	23343.561350	1.499396

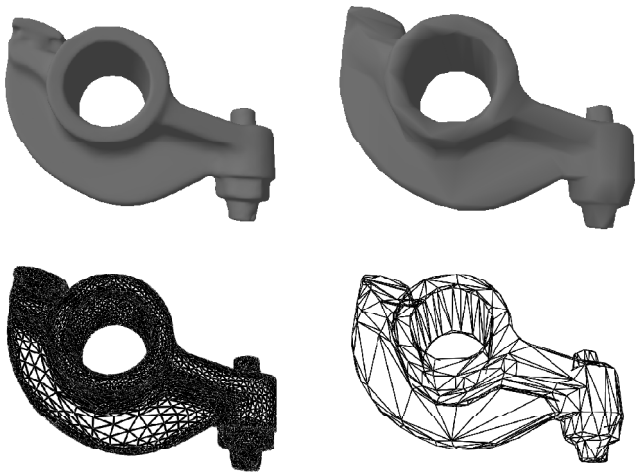


Figure 12: Rockerarm model: shaded (top), wireframe (bottom). Left: original (10000 vertices); Right: 95% decimated (500 vertices)

7. CONCLUSIONS

We have implemented a vertex decimation program that bases its selection of vertices for removal on the curvature and area of the vertex. The program progressively removes vertices from the mesh, choosing the next vertex with the smallest importance weight. The changes made to the local geometry are optimised using again the curvature measures of the affected vertices.

Experimental results have shown how the use of the Total Absolute Curvature of a vertex provides significantly improved results in mesh simplification over the use of the more common Angle Deficit computation. This is due to the

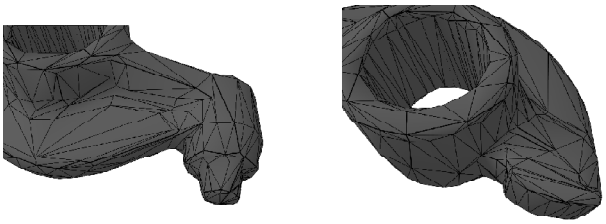


Figure 13: Detail of the decimated rockerarm model at 95% decimation

fact that the TAC provides a more detailed description of the neighbourhood of a vertex and can distinguish features that the AD would not. Both curvature measures are employed to assign relevance weights to vertices and to optimise the triangulation of affected areas using edge flips.

We have experimented with different parameters to determine the importance of a vertex in the mesh. A set of vertex weights based on the TAC were presented, generally referred to as the WTAC. These involve various measures of the area of the neighbourhood of the vertex. From the results obtained, the best overall factor is the one known as the ATAC, where the TAC is multiplied by the surface area of the triangles incident on the vertex. Other weights are better at preserving the smaller details of the object, but alter significantly the shape of the object in more even regions.

The ATAC parameter gives more importance to vertices with larger stars and large curvature. It is more likely to remove vertices with large curvature and small areas, since these represent relatively minor details of the mesh. Large regions of small curvature are better preserved and help retain the overall shape of the object. The parameters that emphasise the curvature (TAC, PTAC, and LTAC) modify the shape of the object by changing the regions of smaller curvature into very large flat areas, but keep the smaller regions with high curvature that make up the sharp details.

8. FUTURE RESEARCH DIRECTIONS

At the moment the algorithm works progressively vertex by vertex. The next step is to move from vertices to regions, by grouping together regions of similar curvature and removing all vertices within that region with minimal WTAC. The Gauss Map allows us to identify such regions. This however implies a graph-theoretical problem to navigate around the mesh analysing the regions, and is outside of the scope of the current work.

After the removal of a vertex, the current policy for the optimisation of the modified region focuses only on minimisation of the local curvature. In some cases this may actually alter the shape of the object, since important feature edges may be flipped for the sake of a smaller curvature. It is necessary to identify these edges and ensure they are not flipped during the optimisation phase. One way to address this problem is to also use the difference between the curvature of the region before and after the removal of the vertex as a parameter to select vertices for decimation. This requires extra computational load to compute the cost of removing every single vertex, but may produce better results. Currently the proposed ATAC weight gives the best results, but at the loss of some detail in the mesh, while the other three weights tested are better at preserving such detail. An implementation that can keep the shape of the object on both large and small features would be desirable. This could be possible by selecting an appropriate weight measure for each individual vertex according to the properties of its neighbourhood.

REFERENCES

[1] Aichholzer, O., L. S. Alboul, and F. Hurtado (2002), On flips in polyhedral surfaces. *International Journal of Foundations of Computer Science* 13(2), 303-311.

[2] Alboul, L. (2003), Optimising triangulated polyhedral surfaces with self-intersections. In M. J. Wilson and R. R. Martin (Eds.), *IMA Conference on the Mathematics of Surfaces*, Volume 2768 of *Lecture Notes in Computer Science*, pp. 48–72. Springer.

[3] Alboul, L. and G. Echeverria (2005), Polyhedral Gauss maps and curvature characterisation of triangle meshes. In R. R. Martin, H. E. Bez, and M. A. Sabin (Eds.), *IMA Conference on the Mathematics of Surfaces*, Volume 3604 of *Lecture Notes in Computer Science*, pp. 14–33. Springer.

[4] L. Alboul, G. Kloosterman, C. R. Traas and R. van Damme (2000), Best data-dependent triangulations. *Journal of Comp. and Applied Math.*, **119**, 1-12.

[5] Banchoff, T. F. (1970), Critical points and curvature for embedded polyhedral surfaces. *American Mathematical Monthly* 7, 475–485.

[6] Cignoni, P., C. Montani, and R. Scopigno (1998), A comparison of mesh simplication algorithms. *Computers & Graphics* 22(1), 37–54. ISSN 0097-8493.

[7] Cignoni, P., C. Rocchini, and R. Scopigno (1998), Metro: Measuring error on simplfiied surfaces. *Computer Graphics Forum* 17(2), 167–174. ISSN 1067-7055.

[8] Cohen, J., M. Olano, and D. Manocha (1998), Appearance-preserving simplification. *Computer Graphics 32*(Annual Conference Series), 115–122.

[9] Echeverria, G. and L. Alboul (2006), Decimation and smoothing of triangular meshes based on curvature from the polyhedral Gauss map. In *Proc. of CompIMAGE Conference. Coimbra, Portugal*, pp. 175–180.

[10] Garland, M. and P. S. Heckbert (1997), Surface simplification using quadric error metrics. *Computer Graphics 31*(Annual Conference Series), 209–216.

[11] Hussain, M., Y. Okada, and K. Niijima (2004), Efficient and feature-preserving triangular mesh decimation. In V. Skala (Ed.), *Journal of WSCG*, Volume 12, Plzen, Czech Republic. UNION Agency-Science Press.

[12] Kim, S. J., C. H. Kim, and D. Levin (2002), Surface simplification using a discrete curvature norm. *Computers and Graphics* 26(5), 657–663.

[13] Lawson, C. (1972), Transforming triangulations. *Discrete mathematics 3*, 365–372.

[14] Lee, A.W. F.,W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin (1998, July 19–24), MAPS: Multiresolution adaptive parameterization of surfaces. In *Proceedings of the ACM Conference on Computer Graphics (SIGGRAPH-98)*, New York, pp. 95–104. ACM Press.

[15] Lindstrom, P. and G. Turk (1998), Fast and memory efficient polygonal simplification. In *IEEE Visualization*, pp. 279–286.

[16] Luebke, D. P. (2001), A developer's survey of polygonal simplification algorithms. *IEEE Computer Graphics and Applications* 21(3), 24–35.

[17] Osserman, R. (1978), The isoperimetric inequality. *Bull. Amer. Math. Soc.* 84, 1182–1238.

[18] Schroeder, W. J., J. A. Zarge, and W. E. Lorensen (1992), Decimation of triangle meshes. In E. E. Catmull (Ed.), *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26, 65–70.

[19] Turk, G. (1992), Re-tiling polygonal surfaces. *Proceedings of SIGGRAPH'92*, 55–64.

[20] Wu, J. and L. Kobbelt (2002), Fast mesh decimation by multiple-choice techniques. In G. Greiner (Ed.), *Proceedings* of the Vision, Modeling, and Visualization Conference 2002 (VMV 2002), Erlangen, Germany, *November 20-22, 2002*, pp. 241–248. Aka GmbH.