Journal for Geometry and Graphics
Volume 7 (2003), No. 2, 221–236.

Planar Grouping for Fast Affine Transformation and Clipping

Jonathan Dinerstein[1], Miklós Hoffmann[2], Parris Egbert[1], Kathryn Turner[3]

[1] *Computer Science Department, Brigham Young University,*
3366 TMCB, Provo, Utah 84602, USA
email: jondinerstein@yahoo.com, egbert@cs.byu.edu

[2] *Institute of Mathematics and Informatics, Károly Eszterházy College,*
H-3300 Eger, Hungary
email: hofi@ektf.hu

[3] *Mathematics Department, Utah State University, USA*
email: kturner@math.usu.edu

Abstract. We introduce planar grouping, a technique where planar relationship information is gathered for a set of rigid points. This information is used to accelerate affine transformations and clipping. The planar grouping technique is an optimization problem, implemented in a best-first greedy search. We present two error metrics, one simple and fast, one based on the quadric error metric (QEM) to achieve higher quality planar grouping. We also apply the quadric error metric to linear grouping.

Key Words: linear grouping, planar grouping, quadratic error metric

MSC 2000: 68U05

1. Introduction

A major bottleneck in real-time rendering of polygon meshes is geometry processing [8]. This includes transformation, clipping, etc. Even with rendering hardware that will perform transformations (e.g. nVidia GeForce, ATI Radeon), transformation is still a serious issue. For example, if polygons are too small, the rendering will probably be transform-bound.

The leading approach to reducing transformation overhead is level of detail (LOD). While effective in reducing computational requirements, LOD is limited because it does not eliminate any *required* transformations. In other words, once a mesh has been optimized through LOD to a minimum acceptable quality, transformation of the remaining vertices is not sped up at all.

ISSN 1433-8157/$ 2.50

In [3], a new rendering optimization technique named *linear grouping* was introduced. It was discovered that, when rendering most polygon meshes, it is not necessary to perform a full transformation on every vertex. Many of the vertices can be transformed with simpler computations using linear relationship information. Further, the linear relationship information can be used to speed up clipping.

In this paper, we present a new technique, *planar grouping*, which is fundamentally similar to but very distinct from linear grouping. In particular, planar (rather than linear) relationship information is gathered for a set of rigid points. Further, we replace the original simple error metric with the quadric error metric (QEM) [5, 6, 9, 10, 12]. This provides much higher quality grouping, as we will show later.

This paper is organized as follows. In Section 2 we review linear grouping and present its primary weaknesses. In Section 3 we introduce planar grouping, and give a brief comparison of planar and linear grouping. In Section 4 we review QEM, and in Section 5 present an application of QEM to planar and linear grouping. Finally, in Section 6 we present experimental results of planar grouping, and conclude in Section 7.

2. Review of linear grouping

2.1. Linear groups

A *linear group* is a set of three or more collinear vertices, as shown in Fig. 1. Any affine transformation can be applied to the vertices and they will still be collinear, and the ratio of distances between them will still be the same. Based on the knowledge that these three vertices are collinear, it is unnecessary to perform a full matrix multiply on all three to transform them. Rather, two of the vertices can be transformed as usual, and then the third vertex reconstructed using the line defined by the two transformed vertices (requiring less computation than a matrix multiply). This is the fundamental idea behind *linear grouping*.

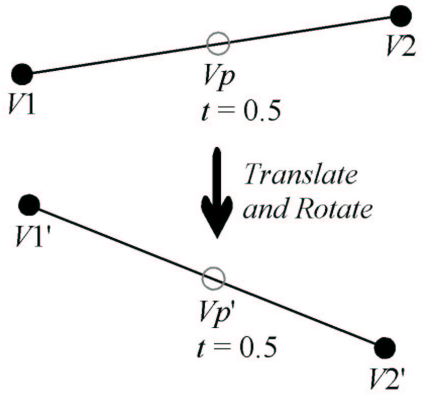

Figure 1: Example of a linear group of three vertices. Vertices $V1$ and $V2$ must be retained to define the line, but Vp can be reduced to a parameter on the line (t). After transforming $V1$ and $V2$, Vp can be rapidly reconstructed-transformed using the equation of the line $V1'V2'$.

Surprisingly, for most polygon meshes, it is possible to group many vertices together in (nearly) collinear sets of three or more. Each of these sets can be represented by keeping two vertices, $V1$ and $V2$, to define the line, and replacing every other vertex W by its parameter

t [11]:

$$W(t) = t * (V2 - V1) + V1.$$

Of course, it is rare for three or more vertices to be exactly collinear. Therefore, an error threshold ε is used, representing the maximum distance between a line and the vertices associated with it. A good measure for the value of ε is the size of the polygon model being considered. $0.5\% \leq \varepsilon \leq 1.5\%$ of the object coordinate space is effective. This is the original error metric of linear grouping, which we will improve upon later in this paper.

Linear grouping is a two step process: 1) constructing the linear groups, and 2) transforming and reconstructing the vertices from the groups. Step 1 is performed only once, at either creation or initialization. Step 2 is performed each time the model is to be rendered.

To transform all points on the line, we first transform $V1$ and $V2$, computing

$$\begin{aligned} V1' &= M * V1 \\ V2' &= M * V2 \\ L' &= V2' - V1', \end{aligned}$$

at the cost of two homogeneous matrix multiplications plus a vector subtraction. The remaining points can be transformed by

$$W(t) = t * L' + V1', \tag{1}$$

at a cost of one scalar-vector multiplication and one vector-vector addition — significantly less than a matrix multiplication.

Linear grouping can also be used to speed up vertex clipping, as detailed in [3]. In short, linear groups can be constructed using line segments between the two defining vertices. If the line segment can be trivially accepted/rejected in clipping, it is known that all grouped vertices will also be accepted/rejected.

2.2. Constructing linear groups

We want to find a linear grouping (set of groups) that eliminates as many vertices as possible; finding a good (or best) grouping is an optimization problem. The technique presented in [4] is a best-first greedy search of all possible groups. The best-first order is computed before searching, building a table of all possible vertex pairs, according to how many vertices fall within ε of each line. The linear groups are constructed by greedily grouping all free vertices within ε of the line.

Note that each vertex can be in one of three states during the search: *free* (unused), *defining* (defines one or more lines), and *grouped* (reduced to a parameter on a line). Initially, all vertices are in the free state. Defining vertices can define more than one line, but grouped vertices can only be associated with one group.

This algorithm is $O(n^2)$ — there are $n^2/2$ groups to test (where n is the number of vertices) since each vertex pair defines one possible group. Using this algorithm, a 300-vertex dataset can be linear grouped in a few seconds on an 800 MHz processor. A computational boundary is reached at several thousand vertices. To linear group larger data sets, the vertices must be partitioned and grouped separately.

Linear grouping, with $\varepsilon = 0.5\%$ of the object coordinate space, can eliminate on average 55% of the vertices in a 3D polygon mesh with little to no visible quality loss. This results in an overall savings of about 50% on transformation computation. Linear grouping

is even more effective on 2D datasets (since the Euclidean distance between points and lines in 2D is naturally less than in 3D), eliminating about 76% of vertices on average. This is especially applicable to texture transformation, as texture coordinates are usually 2D. 3D clipping overhead is reduced by about 32%.

It should be noted that linear grouping is limited to rigid point data. However, since it is not dependent on polygon structure, linear grouping can even be applied to an unorganized cloud of points.

2.3. Weaknesses of linear grouping

Linear grouping is very interesting, as there are no other known techniques like it. In particular, it is able to speed up rigid point transformation without having to remove any points (such as LOD). This allows linear grouping to speed up rendering of meshes that have already been optimized through LOD to a minimum acceptable quality level.

However, linear grouping suffers from some significant weaknesses:

1. Only 55% of vertices are eliminated on average (there could potentially be more savings).
2. The $O(n^2)$ search algorithm is very slow.
3. Linear grouping is lossy, especially in areas of high geometric and surface-property detail.

In this paper, we present a new technique called *planar grouping*. Rather than grouping points on lines, points are grouped on planes. We will show in this paper how planar grouping alleviates all of the aforementioned weaknesses.

3. Planar grouping

We have developed a fast and effective technique for grouping and reconstructing vertices on planes. Under our approach, planar grouping is capable of speeding up transformation more than linear grouping. Further, the planar grouping (search) algorithm is much faster than linear grouping. We now present this technique.

3.1. Planar groups

A planar group is a set of two or more vertices (from the original, non-transformed model) that are in an arbitrary plane. Any affine transformation can be applied to the vertices and they will still be coplanar, and the ratio of distances between them will still be the same. Based on the knowledge that these vertices are coplanar, it is unnecessary to perform a full matrix multiply on all of them to transform them. Rather, one of the vertices can be transformed as usual and the new orientation and scale of the plane computed; then the other vertices can be reconstructed using the definition of the plane (requiring less computation than a matrix multiply) — see Fig. 2.

For simplicity and speed, we prefer to limit grouping to planes parallel to the coordinate planes. However, this can make planar grouping dependent on the orientation of a given model. We will discuss this more later.

Each planar group can be represented by keeping one vertex, $V1$, and two unit vectors, E_s and E_t, to define the plane. Every other vertex W is replaced by its parameter pair (s, t):

$$W(s,t) = V1 + sE_s + tE_t.$$

The unit vectors, E_s and E_t, are oriented according to the plane they define. As an example, if the group is parallel to the XY plane, then $E_s = (1\ 0\ 0)$ and $E_t = (0\ 1\ 0)$.

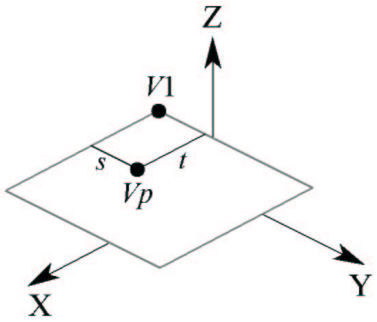

Figure 2: Example of a planar group of two vertices parallel to the XY plane. Note that the plane is infinite, but is drawn bounded to show the geometry. Vertex $V1$ must be retained to define the plane, but Vp can be reduced to a parameter pair (s, t) on the plane. After transforming $V1$ and unit vectors representing the s and t dimensions, Vp can be rapidly reconstructed-transformed using the equation of the plane.

Of course, it is rare for many vertices to be exactly coplanar. The error metric used in linear grouping can also be used for planar grouping: an error threshold ε, representing the maximum distance between a plane and the vertices associated with it. This metric is simple and fast. However, it does not minimize geometric error. We will improve upon it later in this paper.

Planar grouping is a two step process: 1) constructing the planar groups, and 2) transforming and reconstructing the vertices from the groups. As with linear grouping, step 1 is performed only once, at either reation or initialization. Step 2 is performed each time the model is to be rendered.

To transform the planar grouped model, we first transform the unit vectors E_s and E_t, at the cost of two matrix multiplications. Since this must only be performed once per rendering, the cost is negligible and may be ignored.

To transform all points on a plane, we first transform $V1$, at the cost of one homogeneous matrix multiplication. The remaining points can be transformed by

$$W'(s,t) = V1' + sE'_s + tE'_t, \tag{2}$$

at a cost of two scalar-vector multiplications and two vector-vector additions — significantly less than a homogenous matrix multiplication.

3.2. Constructing planar groups

We want to find a planar grouping (set of groups) that eliminates as many vertices as possible; finding a good (or best) grouping is an optimization problem. We use a best-first greedy search of all possible groups. The best-first order is computed before searching, building a table of all possible planes, according to how many vertices fall within ε of each plane. Note that to make this search fast, we prefer to allow each vertex to define only three planes, each parallel to the XY, XZ, or YZ planes. Therefore, there are $3n$ possible planes (where n is the number of vertices). Alternatively, we could allow more planes to be associated with each vertex. For example, we could use planes parallel to the faces of a dodecahedron, etc. However, this will increase grouping time (but not transformation time using these groups).

All planes defined by one vertex should be considered a single group in the best-first table (this helps minimize the number of vertices that must be retained to define groups). Once the

best-first table has been constructed, the planar groups are constructed by greedily grouping all free vertices within ε of each plane in order. Groups whose defining vertex has already been grouped are ignored, as well as groups that eliminate no vertices.

This $O(n)$ algorithm is very fast, especially in comparison to linear grouping which is $O(n^2)$. With our non-optimized experimental implementation of planar grouping, a 300-vertex dataset can be grouped in just a few milliseconds on an 800 MHz processor; several seconds were required for linear grouping. A 3000-vertex data set can be planar grouped in about 300 milliseconds, but requires about twenty minutes with linear grouping.

Planar grouping, with $\varepsilon = 0.5\%$ of the object coordinate space, can eliminate approximately 93% of the vertices in an average 3D polygon mesh with little to no visible quality loss. This results in an overall savings of about 65% on transformation computation. Compare this to linear grouping, which only eliminates approximately 55% of vertices. Later in this paper we will show how the quadric error metric can be used with more aggressive planar grouping to achieve even higher vertex elimination ratios while keeping error virtually invisible.

Like linear grouping, planar grouping can also be used to speed up vertex clipping. Recall that if a line segment (defining a linear group) can be trivially accepted/rejected through clipping, then all grouped vertices can be accepted/rejected. This same principle applies to planar grouping, if it is known that a planar region can be trivially accepted/rejected. This is a more expensive test than line segment clipping, but planar groups usually contain far more vertices than linear groups.

3.3. Comparison of planar and linear grouping

Planar and linear grouping are similar in some fundamental aspects, but are markedly different with respect to the choice of grouping geometry. Linear grouping has the advantage of using the simpler geometry, a line. This allows for faster reconstruction of vertices than in planar grouping. However, since lines (groups) are defined by vertex pairs, linear grouping has $n^2/2$ groups to test. Alternatively, planar grouping has only $3n$ groups, since we only allow each vertex to define three planes.

Planar grouping alleviates the first two weaknesses of linear grouping listed in Section 2.3. First, using the simple/fast error metric with $\varepsilon = 0.5\%$ (virtually lossless), planar grouping eliminates 93% of vertices on average in polygon meshes, as compared to 55% eliminated in linear grouping. Even though reconstruction of vertices is more expensive in planar grouping than in linear grouping, enough additional vertices are grouped such that a higher overall savings is achieved (about 65% vs. 50%).

Second, planar grouping is $O(n)$, as compared to $O(n^2)$ for linear grouping. Planar grouping can be performed on a 3000-vertex data set in about 300 ms on an 800 MHz processor, but linear grouping requires about twenty minutes.

As for the third linear grouping weakness listed in Section 2.3 (introduction of visible error), planar grouping can also suffer from this weakness. This is because planar grouping can use the same simple/fast error metric used in linear grouping. While this error metric is useful if grouping at high quality levels (e.g. $\varepsilon \leq 0.5\%$), error may become visible when grouping with more aggressive settings (e.g. $\varepsilon > 0.5\%$). There is need for a more robust error metric when performing aggressive planar and linear grouping. In the next two sections we review QEM and present an application of it to both planar and linear grouping. After introducing this new error metric we will present experimental results and conclusions.

It is important to note that planar grouping does not eliminate the usefulness of linear grouping. For example, as discussed in Section 2.2, linear grouping works very well for 2D

data sets. However, planar grouping does not necessarily work well in 2D, because it provides very little speed improvement.

4. Review of QEM

A full review of QEM is beyond the scope of this paper, but we will briefly mention the major points of the technique. Also, we will not review the details of using QEM for LOD, as it is not pertinent to our planar grouping application. Refer to [6, 9, 10] for further details.

QEM was introduced by Garland and Heckbert [5], and is based on the metric of Ronfard and Rossignac [12]. The fundamental idea is to measure the sum of the squared distances of a point from a set of planes. This provides an excellent error metric for LOD. One of the strengths of QEM is that it is very inexpensive to compute.

QEM defines a quadric on each face of the model. The quadric defines the squared distance of a point from the plane. The quadrics are in quadratic form, and are stored in 4×4 matrices denoted $\mathbf{Q}_i$. To compute the sum of squared distances to a set of planes, we only need one quadric (matrix) which is the sum of the quadrics (matrices) defined by each of the individual planes. Each vertex of the model is assigned the sum of the quadrics on its adjacent faces weighted by face area. Eq. (3) shows how to compute the vertex quadric, where $\mathbf{p} = [n_x\ n_y\ n_z\ d]^{\mathrm{T}}$ is the plane definition of face f (with unit normal). Eq. (4) shows how to compute the sum of squared distances using homogenous vertex $\mathbf{v} = [x\ y\ z\ 1]^{\mathrm{T}}$ and quadric $\mathbf{Q}$.

$$\mathbf{Q} = \sum_{\mathbf{p}\in\text{faces(v)}} \text{area}(f)\cdot(\mathbf{p}^{\mathrm{T}}\mathbf{p}) \tag{3}$$

$$Q(\mathbf{v}) = \mathbf{v}^{\mathrm{T}}\mathbf{Q}\mathbf{v}. \tag{4}$$

Unlike most QEM papers, we prefer this homogenous quadratic matrix form for quadrics. This is because it is the most convenient form for our use with planar grouping that we will present later. However, we will make some references to the non-homogenous form. In the non-homogenous form, the quadric $\mathbf{Q}$ is defined as

$$\mathbf{Q} = (\mathbf{A}, \mathbf{b}, c) = (\mathbf{n}\mathbf{n}^{\mathrm{T}}, d\mathbf{n}, d^2)$$
$$Q(\mathbf{v}) = \mathbf{v}^{\mathrm{T}}\mathbf{A}\mathbf{v} + 2\mathbf{b}^{\mathrm{T}}\mathbf{v} + c,$$

where $\mathbf{n}$ is the unit normal of the face. $\mathbf{A}$ is the 3×3 matrix in the top-left corner of the homogenous 4×4 quadric matrix.

4.1. Interpretation of quadrics

QEM quadrics possess some useful geometric properties, which are important to our application of QEM in planar grouping. Refer to [2, 6] for further details.

QEM quadrics are (possibly degenerate) ellipsoids. They characterize the local shape of the surface, as is shown in Fig. 3. Matrix $\mathbf{Q}$ is symmetric positive semidefinite. Matrix $\mathbf{A}$ (upper-left 3×3 of $\mathbf{Q}$) defines the shape of the ellipsoid. The eigenvectors of $\mathbf{A}$ define the principal axes of the ellipsoid, and the eigenvalues define the extent of the ellipsoid on the principal axes (as shown in Fig. 4). In a simplified form, this relationship is

$$\lambda_1 x^2 + \lambda_2 y^2 + \lambda_3 z^2 = 1. \tag{5}$$

The possible degenerate forms of the quadrics are cylinders and parallel planes — but the quadrics are rarely degenerate. If the ellipsoid is degenerate, $\mathbf{A}$ is singular. Since $\mathbf{Q}$ is positive semidefinite, an isosurface (ellipsoid) is defined by $Q(\mathbf{v}) = \varepsilon$, for some positive scalar ε. This isosurface is the set of all points whose error with respect to $\mathbf{Q}$ is ε. If $\varepsilon = 0$, and $\mathbf{Q}$ defines an ellipsoid, the isosurface is a point (i.e. the ellipsoid has no volume). A vertex can be moved anywhere within the ellipsoid and its error will be $\leq \varepsilon$.

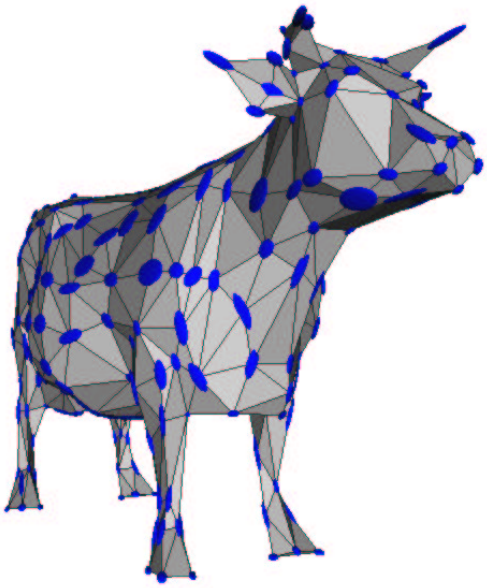

Figure 3: Cow object model rendered with QEM ellipsoids. The ellipsoids characterize the local shape of the surface — they are narrow in directions of high curvature and wide in directions of low curvature.

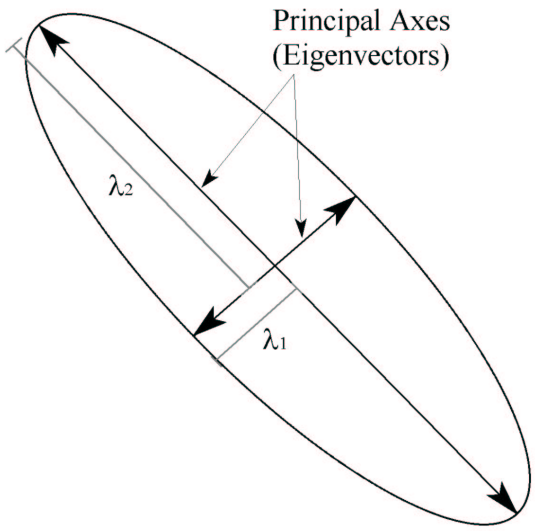

Figure 4: The principal axes of an ellipsoid (quadric) are defined by the eivenvectors of matrix $\mathbf{A}$. The extent of the ellipsoid on the axes is defined by the eigenvalues of $\mathbf{A}$.

5. QEM for planar and linear grouping

One of the weaknesses of linear grouping (number 3 in Section 2.3) is that it has a fast and simple but poor error metric. According to this metric, each vertex can only move some Euclidean distance ε. This distance is the same for all vertices. Conceptually, this means that there is a sphere of radius ε centered at each vertex, and each vertex can move to any location inside the associated sphere. This is a poor error metric because the spheres do not characterize the local shape of the surface. Further, no concern is taken for surface properties, such as texture, color, and normals.

QEM quadrics are a natural extension to linear grouping's original fast/simple error metric. The quadrics define error ellipsoids, inside of which the associated vertex can be moved anywhere and be within error ε. Linear grouping's error spheres are simply a special case of the ellipsoids. Interestingly, QEM quadrics only become sphere-like when the local surface has high curvature in all directions (i.e. vertex motion in any direction results in about the same error). If a quadric is a sphere, its error is a squared Euclidean distance. Otherwise, the error is not a squared Euclidean distance, but the sum of squared distances from the planes.

To use QEM quadrics with planar and linear grouping, we need a technique to build the quadrics. This includes ensuring that the ellipsoids are properly sized so that no geometric errors can be introduced to the surface, such as fold-over. Further, the technique needs to compute an appropriate ε based on the desired model quality/vertex elimination ratio. Finally, an optimal placement scheme is needed for placing the vertices on the lowest-error locations of the lines or planes. We will now present solutions to each of these requirements.

5.1. Creation of ellipsoids and determining ε

The ellipsoids for planar and linear grouping are constructed much like traditional QEM, as in equation (3). The only difference is that the quadrics are not weighted by the face area. We also use the same mesh boundary preservation technique in [6]. Any degenerate quadrics are replaced with very small spheres. This is a simple solution to this problem, and produces good results.

Like [3], we use the dimensions of the object (which will be planar/linear grouped) to compute a proper value for ε. It was recommended in [3] to use $0.5\% \leq \varepsilon \leq 1.5\%$ of the object coordinate space. However, QEM is a far better error metric, and allows for larger values of ε. We have found $0.5\% \leq \varepsilon \leq 2.5\%$ to be effective, and recommend that 1% usually be used. Note that ε needs to be squared, as QEM measures squared distances.

5.2. Avoiding geometric error

QEM ellipsoids do not have a limited maximum size (regardless of the value of ε). The less the curvature of the surface, the larger the ellipsoids are. For surfaces with no curvature along an axis, the ellipsoids degenerate to an infinite size on that axis. An example of large ellipsoids is given in Fig. 5.

The problem with large ellipsoids is that, since a vertex can move anywhere within the ellipsoid, too much motion can take place. This can result in visually displeasing geometry as well as mesh fold-over. To solve this problem, we need to reduce the size of ellipsoids that are too large for the selected value of ε. An ellipsoid can be scaled down by either reducing its ε, or multiplying $\mathbf{Q}$ by a scalar > 1.

There are many possible techniques to determine which ellipsoids are too large and scale them to a proper size. However, this can be a mathematically complex problem. We have

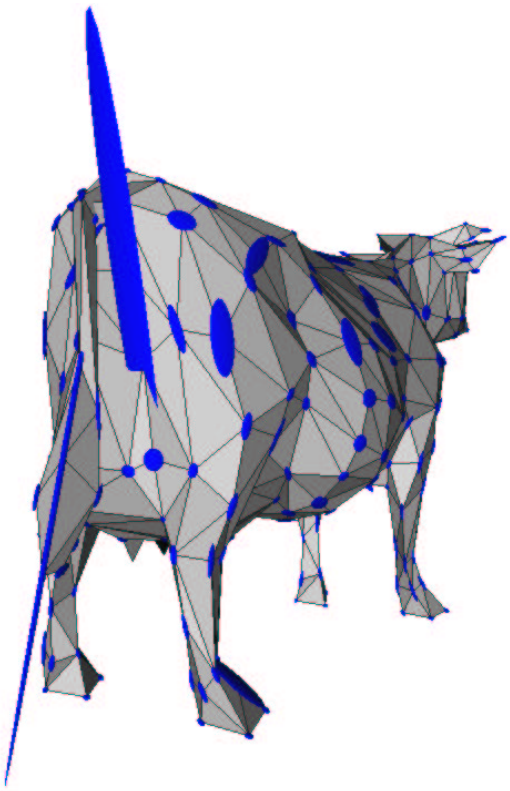

Figure 5: Ellipsoids are large on an axis when the surface (along that axis) has very low curvature. The two large ellipsoids in this figure are mathematically valid, but are too large for use in planar and linear grouping as they allow the associated vertices to move too far.

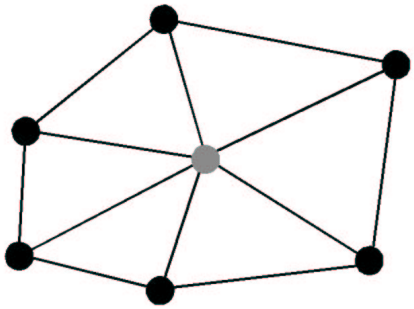

Figure 6: Example of a neighborhood region. The current vertex is the gray circle, and the other vertices are the black circles. The current vertex's ellipsoid is not allowed to extend beyond half the distance along each polygon edge radiating from the current vertex to the other vertices in the neighborhood region.

selected a very simple technique, which has proven effective. Slightly better results might be achieved by using a more accurate technique.

Our error-avoidance technique is based on the realization that, to keep the current vertex from moving too far, the associated ellipsoid should not pass beyond the *neighborhood region* (the area formed by the polygons that share the current vertex). An example neighborhood region is shown in Fig. 6. Further, two vertices should not be allowed to swap places. Our simple error-avoidance technique ensures that the current vertex will not move more than

half the distance along any of the polygon edges that radiate from the current vertex to the other vertices in the neighborhood region.

We iteratively perform our test on each ellipsoid. First, the eigenvectors (principal axes of ellipsoid) and the eigenvalues (extent of ellipsoid on axes) are computed. Next, the actual Euclidean lengths of the ellipsoid's axes are computed from the eigenvalues, using eq. (6). Finally, the lengths are compared of axes and edges that point in generally the same direction (within about $\pi/4$). If the ellipsoid may extend beyond the midpoint of any edge, its matrix $\mathbf{Q}$ is multiplied by a scalar determined by (7).

$$a = \sqrt{\varepsilon/\lambda_1}, \quad b = \sqrt{\varepsilon/\lambda_2}, \quad c = \sqrt{\varepsilon/\lambda_3} \tag{6}$$

$$s = (\text{axis length} \;/\; (\text{edge length}/2))^2. \tag{7}$$

Note that we perform uniform scaling of the ellipsoid, which does not maximize the potential volume. Also, our half-edge test does not determine that the ellipsoid is actually too large, only that it may be. More accurate techniques can be used and may increase the percentage of vertices eliminated, but would require a more complex implementation and greater computational requirements.

5.3. Grouping of vertices

To group vertices, it is necessary to find which vertices are within error ε of a plane or line. We do this by finding the optimal location of a vertex on the plane or line and then computing the error. However, the linear case is somewhat simpler than the planar case, so we present it first.

The parameter on the line of the optimal placement position can be computed as

$$t = \frac{-\mathbf{V}^{\mathrm{T}}\mathbf{y}}{\mathbf{L}^{\mathrm{T}}\mathbf{y}}, \quad \text{where } \mathbf{y} = \mathbf{Q}\mathbf{V}, \tag{8}$$

where $\mathbf{V} = [x_o \; y_o \; z_o \; 1]^{\mathrm{T}}$ and $\mathbf{L} = [i \; j \; k \; 0]^{\mathrm{T}}$ (origin of line and direction of line, respectively) define the parametric line. This is fairly inexpensive at a cost of one matrix-vector multiplication, two vector-vector multiplications, and one scalar division. In fact, this is about the same cost as computing arbitrary point-line distance. The new vertex location can be computed using the parameter t and the equation of the line (1). The error at this point can then be computed using (4). If the error is less than ε, the vertex can be included in the group.

The planar case is somewhat more complex than the linear case. The location on the plane of the optimal placement position can be computed as

$$\mathbf{V} = \frac{b}{\mathbf{a}^{\mathrm{T}}\mathbf{y}}\mathbf{y}, \quad \text{where } \mathbf{y} = \mathbf{Q}^{-1}\mathbf{a}, \tag{9}$$

where $\mathbf{V} = [x \; y \; z \; 1]^{\mathrm{T}}$ is the optimal location, and the plane is defined as $\mathbf{a}^{\mathrm{T}}\mathbf{v} = b$. This costs one matrix-vector multiplication, one vector-vector multiplication, one scalar-vector multiplication, and one scalar division. A matrix inversion is also required, but this can be computed once and reused for each group tested. Overall, this is only somewhat more computationally expensive than the linear case. Moreover, since our grouping planes are simple (axis-aligned), equation (9) can be further simplified in practice. Once the optimal location has been computed, the error at this point can be computed using (4). If the error is less than ε, the vertex can be included in the group.

The grouping of vertices is the most time-intensive process in planar and linear grouping, as all vertices are tested for inclusion in all groups. However, this process is very optimizable, as the average plane- and line-quadric distance is large. Standard ray tracing acceleration techniques [1, 7] are very appropriate for speeding grouping up. For example, a bounding box or sphere can be computed for each quadric (or set of quadrics) and tested for intersection with the plane or line. Space partitioning techniques may also be effective, especially for planar grouping as the planes are axis-oriented (and therefore trivial to compare with volume hierarchies).

In our non-optimized experimental implementation, planar grouping generally takes about three times as long when using QEM. However, planar grouping is fast enough that this may not be an issue. Further, an optimized implementation (i.e. with space partitioning) should be able to avoid most of this additional overhead.

6. Experimental results and findings

Planar grouping has proven to outperform linear grouping in our experiments. Please refer to Section 2.3 for linear grouping's weaknesses. Planar grouping alleviates weaknesses 1 and 2 by grouping far more vertices than linear grouping (93% vs. 55% on average), and its search algorithm is O(n) vs. O(n^2). We solve weakness 3 by applying QEM to both planar and linear grouping. For a more thorough comparison of planar and linear grouping see Section 3.3.

Visual results are given in Figs. 8 through 10 at the end of the paper. As can be seen, planar grouping with QEM produces very good results. For moderate 3D vertex elimination ratios (91% to 95%), there is usually no visible difference in the model. Even with high 3D vertex elimination ratios (95% to 98%), the quality is good. Further, for less aggressive planar grouping ($\varepsilon \leq 0.5\%$), the original fast/simple error metric produces good results. Note that our experiments were performed using models already optimized through LOD (using [6]), so the vertices were in generally random positions; therefore planar grouping was not a trivial problem, as can happen with contrived data.

Planar and linear grouping are lossy techniques by design. However, we believe that the best use of planar/linear grouping is to provide benefit after LOD cannot provide any more benefit without unacceptable quality loss. LOD is a more powerful technique for increasing rendering performance since entire polygons are removed, but is visibly destructive to the shape of the original model (especially when considering an already optimized model) as shown in Fig. 8.

In Table 1 we present some performance results using our experimental implementation of planar grouping for a series of models.

Model (verts)	*Verts eliminated*	*Time taken*
Dragon (600)	95.4%	30 ms
Bunny (3000)	97.5%	317 ms
Cow (1000)	96.4%	60 ms
Jet airplane (300)	89.0%	10 ms

Table 1: This table shows planar grouping results using the simple/fast error metric, with $\varepsilon = 0.5\%$. This experiment was performed on an 800 MHz processor. In comparison, linear grouping the 3000-vertex bunny model requires about twenty minutes.

Note that higher elimination ratios are achieved for less optimized (more dense) meshes, because the vertices are naturally closer together. This is examined in more detail in Fig. 7.

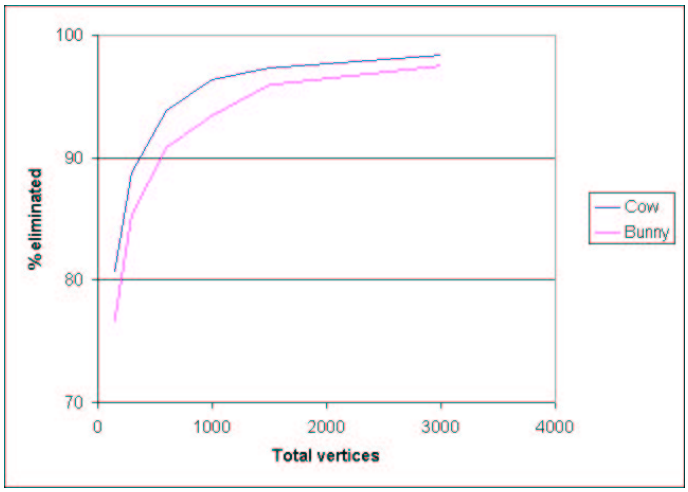

Figure 7: Graph of vertex elimination ratios for the Cow and Bunny models, optimized to different numbers of vertices through LOD (using [6]). This experiment was done with the fast/simple error metric and $\varepsilon = 0.5\%$. Note that the cow model is an example of better than average vertex elimination, and the bunny model less than average. There is generally little variance between how many vertices can be eliminated in meshes of the same number of vertices.

In our non-optimized experimental implementation, the time to planar group a mesh with QEM was generally three times that using the simple/fast error metric. Since planar grouping is very fast, this may not be an issue. Further, an optimized implementation (i.e. with space partitioning) should be able to avoid most of this additional overhead. However, the simple/fast error metric may be preferable if performing conservative planar grouping (i.e. with $\varepsilon \leq 0.5\%$), since it is faster and easier to implement than QEM, and will generally produce results as good as QEM for low values of ε. Further, since high vertex elimination ratios are possible in planar grouping with $\varepsilon = 0.5\%$, QEM may not be necessary unless achieving the highest possible vertex elimination is required.

Note that, as mentioned previously, planar groups do not have to be coordinate plane aligned. The formulation of planar grouping, and both error metrics we have presented, are sufficiently general for arbitrary planes. However, through our experimentation, we have found that there is usually no tangible benefit to using non-aligned planes — very few additional vertices are eliminated, and computational cost of performing the grouping can increase notably. However, if desired to ensure near-optimal grouping, more planes can be used (e.g. planes parallel to the faces of a dodecahedron).

We also experimented with using a more robust search for grouping, e.g. a branch-and-bound search. However, this proved to provide little or no benefit over our greedy best-first search, while being significantly slower.

Planar grouping is very simple to implement, especially if the simple/fast error metric is used. In fact, it can be implemented in just one or two pages of readable C code. This simplicity is a valuable feature.

While equations (8) and (9) have not been explicitly extended for use with augmented quadric matrices for surface properties [6, 9], such an extension should be straightforward.

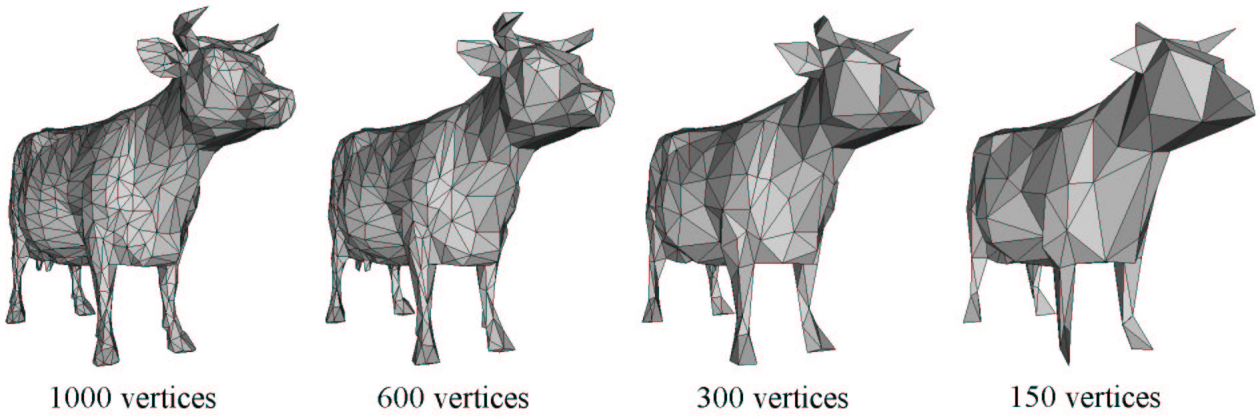

Figure 8: Example of LOD (using [6]), where the cow model has been optimized to different numbers of vertices. The loss of detail is much greater for small numbers of vertices/faces. This is why planar grouping is valuable — it provides faster transformation when no more vertices/faces can be removed without unacceptable quality loss.

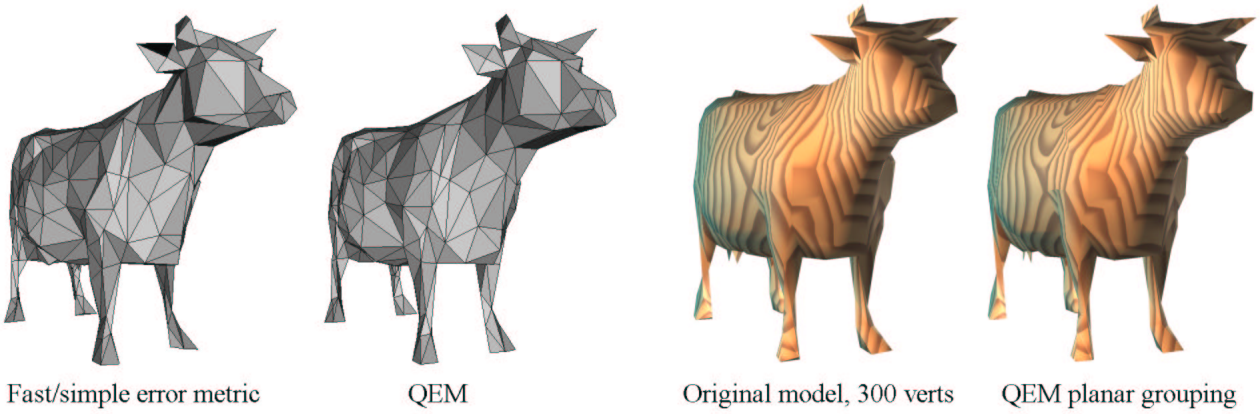

Figure 9: Quality comparison using 300-vertex cow model (compare to Fig. 8). A large ε was used. The original error metric is visibly lower quality than QEM (consider the front-left leg, nose, and right shoulder). On the right are the original and QEM planar grouped models rendered with a 3D procedural wood texture (to show any volume/shape changes). Note that the planar grouped models can be transformed in less than one-half the time required for the original LOD model.

However, this may not be necessary for planar and linear grouping to produce good results. The tessellation of polygonal meshes (before or after LOD) is usually based in part on how fine of detail the surface properties represent. Therefore, where surface properties represent fine detail, the faces will be small. By using the ellipsoid size tests presented in Section 5.2, the ellipsoids in fine-detail regions will naturally be small, allowing only small motion in the vertices. However, a surface property discontinuity may not require fine tessellation, and therefore could be problematic.

Of course, QEM is not usable for planar or linear grouping of a set of points not associated with geometry. For example, texture coordinates, a cloud of unorganized points, etc. Under such a case, the original simple/fast error metric should be used (since it is geometry independent).

It is important to note that planar/linear grouping can only be used for transforming a rigid set of points. This includes rigid vertices, rigid texture coordinates, texture coordinate generation from rigid vertices, etc.

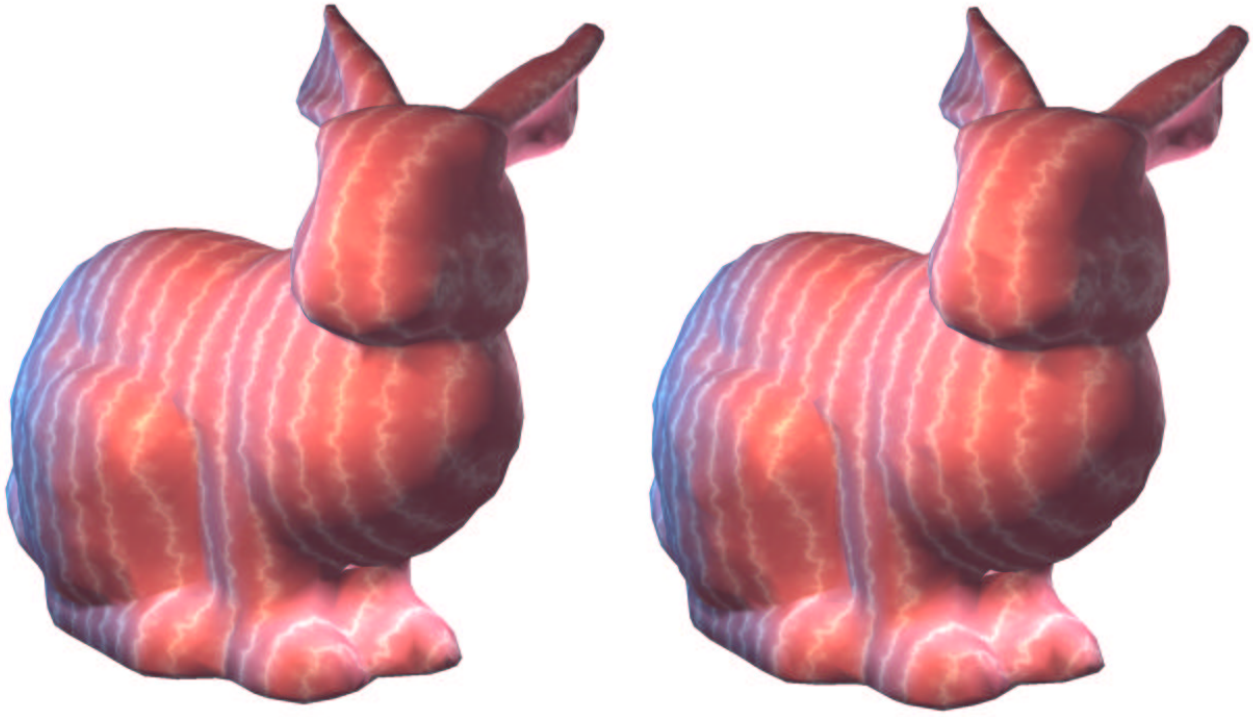

Figure 10: 1,500 vertex bunny model rendered with a 3D procedural marble texture. The original model is on the left, and the QEM planar grouped model is on the right, with $\varepsilon = 0.5\%$. They are virtually identical. In this example, planar grouping reduces transformation time to less than one-third the time required for the original LOD model.

7. Conclusion and future work

Linear grouping is a powerful technique. There are no other algorithms we are aware of that provide a similar service (optimizing vertex transformation without removing vertices). This has valuable application as a post-LOD tool. In other words, once a mesh has been optimized to minimum acceptable quality through LOD, linear grouping can be applied to achieve even faster rendering.

In this paper, we have introduced planar grouping, a new technique similar to linear grouping, but with superior performance (both in grouping and rendering). We have also introduced a better error metric based on QEM, and applied it to both planar and linear grouping. We have shown planar grouping to be very effective, and that it usually causes no visible change in the geometry for moderate vertex elimination ratios (91% to 95%). For low to moderate vertex elimination, the fast/simple error metric provides results as good as QEM. However, for aggressive planar and linear grouping, QEM provides better results.

There are many opportunities for further research. For example, there are many ways that QEM can be applied to planar and linear grouping. We chose an approach that is mathematically simple, but admittedly not optimal. There may be a better approach that is also reasonably mathematically and computationally simple.

Another area for further work is to apply planar and/or linear grouping to more of the rendering pipeline. For example, could planar grouping be used for normal vector transformation? Further, could planar grouping be useful for geometry compression? We expect that there are many further useful applications of planar/linear grouping.

Another interesting extension to planar/linear grouping would be support for non-rigid data sets. For example, could we rapidly compute how the location of vertices may change within a planar/linear grouping? If so, the vertices could simply be updated and then rapidly transformed.

Acknowledgements

The second author's research was funded in part by European Research Training Network MINGLE (HPRN-1999-00117). The bunny model is courtesy Stanford Graphics Lab. The cow model is courtesy of Avalon and Viewpoint Digital.

References

[1] J. Arvo, D. Kirk: *A Survey of Ray Tracing Acceleration Techniques*. In A. Glassner (ed.): *An Introduction to Ray Tracing*. Academic Press Limited, London 1989.

[2] J. Blinn: *The Algebraic Properties of Second-Order Surfaces*. In J. Bloomenthal (ed.): *Introduction to Implicit Surfaces*.Morgan Kaufmann Publishers Inc, San Fransisco 1997.

[3] J. Dinerstein, L. Egbert, N. Flann: *Linear Grouping — A Method for Optimizing 3D Vertex Transformation and Clipping*. Journal of Graphics Tools **6**(1), 1–6 (2001).

[4] J. Dinerstein, P. Egbert: *Improved Linear Grouping*. Proc. 5th International Conference of Computer Graphics and Artificial Intelligence, 137–142 (2002).

[5] M. Garland, P.S. Heckbert: *Surface Simplification Using Quadric Error Metrics*. SIGGRAPH, 209–216 (1997).

[6] M. Garland, P.S. Heckbert: *Surface Simplification with Color and Texture Using Quadric Error Metrics*. IEEE Visualization, 263–269 (1998).

[7] E. Haines: *Essential Ray Tracing Algorithms*. In A. Glassner (ed.): *An Introduction to Ray Tracing*. Academic Press Limited, London 1989.

[8] H. Hoppe: *Smooth View-Dependent Level-of-Detail Control and its Application to Terrain Rendering*. IEEE Visualization, 35–42 (1998).

[9] H. Hoppe: *New Quadric Metric for Simplifying Meshes with Appearance Attributes*. IEEE Visualization, 59–66 (1999).

[10] P. Lindstrom, G. Turk: *Fast and Memory Efficient Polygonal Simplification*. IEEE Visualization, 279–286 (1998).

[11] J. O'Rourke: *Computational Geometry in C*. 2nd ed., Cambridge University Press, Cambridge/UK 1998.

[12] R. Ronfard, J. Rossignac: *Full-Range Approximation of Triangulated Polyhedra*. Eurographics, 67–76 (1996).

Received May 8, 2003; final form November 19, 2003