Ellipsoid decomposition of 3D-models

Stephan Bischoff Leif Kobbelt

Computer Graphics Group
RWTH Aachen, Germany

Abstract

In this paper we present a simple technique to approximate the volume enclosed by a given triangle mesh with a set of overlapping ellipsoids. This type of geometry representation allows us to approximately reconstruct 3D-shapes from a very small amount of information being transmitted. The two central questions that we address are: how can we compute optimal fitting ellipsoids that lie in the interior of a given triangle mesh and how do we select the most significant (least redundant) subset from a huge number of candidate ellipsoids. Our major motivation for computing ellipsoid decompositions is the robust transmission of geometric objects where the receiver can reconstruct the 3D-shape even if part of the data gets lost during transmission.

1 Introduction

Today the most common representation for 3D objects are polygon meshes. They are flexible enough to approximate arbitrary shapes and the complexity of their processing increases linearly with the number of polygons but is independent from the structural or topological complexity of the object. There is a huge body of literature about various techniques for the generation, modification, storage, transmission, and display of polygonal models [6, 21, 22, 20, 23].

Although polygons, i.e. piecewise linear surfaces have approximation properties that are sufficient for most graphics applications it turns out that realistic models still require up to several millions of triangles. This is why hierarchical models and multiresolution representations have been introduced into computer graphics [9, 11, 20]. The basic observation here is that a coarse approximation of a surface can already be obtained with a surprisingly small number of triangles [10, 18, 19]. If we use more triangles, the approximation gets better and better but the *efficiency*, i.e., the quality gain per polygon is quickly decreasing. Progressive meshes exploit this effect by storing very coarse base meshes plus a sequence of refinement operations [7, 12, 15]. Any prefix the transmitted sequence this sequence allows the receiver to reconstruct an approximation of the original mesh and the representation is progressive in the sense that the most important shape features show up very early in the sequence while the big number of less important details follow later in the sequence.

One drawback of polygon meshes — even in their hierarchical or progressive representation — are the strict topological consistency requirements. If we lose one percent of the mesh data then it might become impossible to reconstruct any part of the surface from the remaining 99 percent. This is mostly due to the necessary cross references between faces and vertices and can be solved only by introducing a high degree of redundancy into the representation. This observation motivates the investigation of robust transmission schemes which allow the receiver to reconstruct most of the geometric information even if a certain percentage of the data is lost during transmission.

The key to robust transmission is to divide the geometry into independent chunks of information (e.g. single points) which allow the receiver to reconstruct the manifold neighborhood relation without relying on indexing or explicit inter-vertex reference information. In order to avoid complex algorithms for general surface reconstruction from point clouds on the receiver's side, earlier approaches to robust transmission assumed that at least the global surface topology can be transmitted losslessly [5]. Once this coarse shape information is known, the receiver can insert additional points into the mesh based on spatial proximity (cf. Fig. 1 and Fig. 2).

In this paper we want to extend this earlier approach such that the coarse shape information can be transmitted robustly as well. The general approach is to approximate the volume enclosed by the given polygon mesh with a set of overlapping ellipsoids. These ellipsoids represent independent pieces of geometric information and each ellipsoid defines its own part of the geometry. Redundancy is introduced by the fact that the ellipsoids overlap each other. Hence the overall shape and especially the surface topology will not change

Figure 1: In order to avoid that the robust transmission becomes as complicated as the general surface reconstruction problem from point clouds, the sender (who knows the original mesh) provides additional information to the receiver which simplifies the reconstruction (*hinted reconstruction*). In this case the sender provides a coarse base mesh to convey the global surface topology. All other mesh vertices are sent without any connectivity information. The receiver can approximately reconstruct the orignal shape by inserting the vertices into the mesh based on spatial proximity.

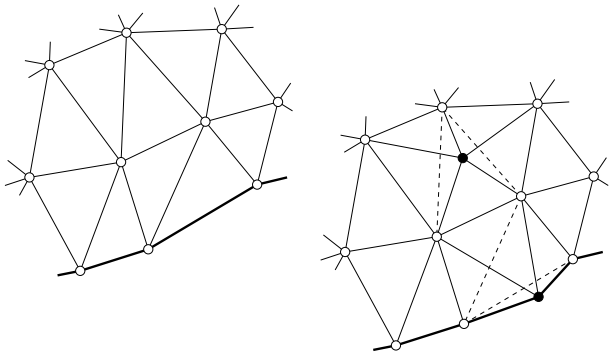

Figure 2: If in a robust transmission scheme vertices are received without any connectivity information, they are simply inserted into the nearest triangle. In order to preserve the mesh quality at all stages, every vertex insertion is followed by edge-flipping operations in order improve the valence balance or the aspect ratio of the triangles.

if only a few ellipsoids are missing. The redundancy does not significantly increase the storage requirements since an ellipsoid can be defined with only 9 scalar values. This corresponds to the storage requirements of one vertex in a shared-vertex triangle mesh representation[1].

The paper is organized as follows: in the next section we describe the main steps of our robust transmission procedure. Then we explain the details of our optimal ellipsoid fitting technique in Section 3. Finally we discuss results and future application scenarios in Section 4.

[1]Every vertex requires 3 coordinate values, each triangle 3 vertex indices. Since there are twice as many triangles as vertices, we have 9 values per vertex.

2 Robust geometry transmission

For robust transmission we have to guarantee that whenever a part of the geometry information gets lost, the remaining portion of the data can be used to reconstruct at least a good approximation of the object, i.e., the approximation quality *slowly* degrades with an increasing percentage of lost data. Our basic idea is to decompose the geometric shape into a set of ellipsoids representing the coarse shape of the object plus a set of independent sample points which represent the fine detail.

The ellipsoids are generated such that they completely fill out the interior of the object, the set of sample points is simply the set of original mesh vertices.

The robust transmission procedure works as follows: First the ellipsoids are transmitted. Since they are given implicitly by a symmetric 4×4 matrix Q, normalized such that $x^T Qx > 0$ for exterior and $x^T Qx < 0$ for interior points x, it is easy to collect their shape information in a spatial distance field. For this we define a sufficiently fine spatial grid of scalar values which is initialized to $+1$ everywhere. When a new ellipsoid Q is received, we compute its bounding box and evaluate the implicit function $x^T Qx$ at all the grid-points within that box. The grid values are updated if the new function value is smaller than the current grid value.

When all ellipsoids are received, we extract the zero-surface from the spatial scalar field by applying the marching cubes algorithm [3, 4]. Obviously the method still works even if some of the ellipsoids are dropped. The extracted surface will not change its topology as long as the remaining ellipsoids are still overlapping. Moreover, if the mutual overlap is strong enough, the loss of a small portion of the ellipsoids will not affect the reconstructed shape significantly.

Once we have generated an initial mesh, we start inserting the sample points which are the original mesh vertices. The

details of this procedure are described in [5]. All we need is a good initial surface approximation and then mesh reconstruction based on vertex insertion and edge flipping works robustly. To increase the quality of the final result, we have to eventually eliminate those mesh vertices which have been generated by the marching cubes algorithm since they are not part of the original mesh geometry.

3 Ellipsoid decomposition

The problem of covering the volume in the interior of a given polygon mesh is not well-defined since, obviously, there are many different possible ellipsoid decompositions. Our algorithm is designed to find one candidate among this multitude of decompositions that is locally optimal with respect to shape, orientation and distribution of the ellipsoids. A local optimum in this context means that we check a finite number of configurations and search for the best one by some greedy optimization scheme.

3.1 Ellipsoid fitting

Fitting ellipsoids to a given volume is not easy—even much simpler problems like e.g. finding the smallest enclosing sphere are mathematically involved [17]. In a first attempt to fit ellipsoids to the volume bounded by a polygon mesh, one might pick a random seed point in the interior, determine three orthogonal principal axes and then grow the ellipsoid along these axes until it touches the mesh. However, there are several critical drawbacks in this straightforward procedure.

First, random distribution of seed samples in the interior does not take the global shape of the 3D object into account. For example if we have a large convex shape with some long and thin feature peeking out (cf. Fig 3) then the random distribution will pick seed points with a very high probability in the convex region of the object. Seed samples in the thin feature a very unlikely since the relative volume is too small. Nevertheless this small feature carries a significant visual and geometrical information and hence it is more important than an accurate approximation of the convex part. Consequently it makes more sense to place the random seed points on the surface of the polygon mesh and to grow the ellipsoids into the interior.

The second problem is that it is almost impossible to determine a good choice for the principal directions at a point in the interior. For cylindric shapes, e.g., we would like to align one principal direction to the cylinder axis. This, however is difficult to tell for a point lying in the interior of the mesh. We could use a variant of the medial axis transform and local distance field information to optimize the principal axis orientation but estimating the medial axis of a polygon mesh is computationally complex and prone to numerical instabilities.

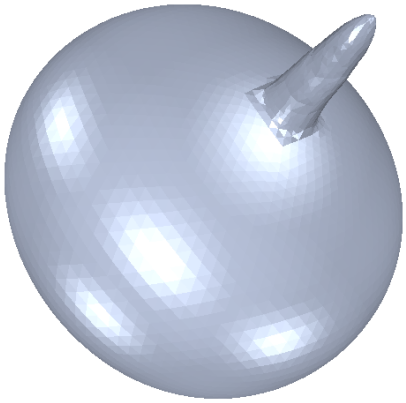

Figure 3: A large convex object with a small feature. Volumetric random samples hit the feature with a very small probability while random surface samples are more likely to hit.

Again, starting the ellipsoid growth on the boundary simplifies the situation: We can start growing an ellipsoid from a random point $\mathbf{p}$ on the boundary until it touches another surface point $\mathbf{q}$. Then growth is continued into the direction(s) orthogonal to the axis $\overline{\mathbf{pq}}$ until a third point $\mathbf{r}$ is found. Finally we grow in the orthogonal direction to the plane $\overline{\mathbf{pqr}}$.

3.2 Ellipsoid growth strategy

The ellipsoid growth starting from a surface point $\mathbf{p}$ is now described in more detail. We begin with the estimation of a normal vector $\mathbf{n}_p$ at $\mathbf{p}$. This is easy no matter if $\mathbf{p}$ lies in the interior of a triangular face, on an edge, or coincides with a vertex of the mesh. We define a sphere $Q_1(r)$ with radius r whose center is $\mathbf{c}_r = \mathbf{p} + r\mathbf{n}_p$ such that it touches the point $\mathbf{p}$. Then we increase the value of r until it touches a second surface point $\mathbf{q}$.

To determine the optimal r we first compute for each vertex $\mathbf{x}$ the radius $r(\mathbf{x})$ of the sphere defined by the interpolation conditions $\mathbf{x}$, $\mathbf{p}$ and the normal $\mathbf{n}_p$

$$r(\mathbf{x}) = \frac{||\mathbf{x}-\mathbf{p}||^2}{2(\mathbf{x}-\mathbf{p})^T\mathbf{n}_p}$$

Then we set

$$r_{opt} = \min\{r(\mathbf{x})|(\mathbf{x}-\mathbf{p})^T\mathbf{n}_p > 0\}$$

After we found the largest inscribed sphere $Q_1(r)$ with the two contact points $\mathbf{p}$ and $\mathbf{q}$ we continue growing in orthogonal direction to the axis $\overline{\mathbf{pq}}$. For this we re-write the quadric

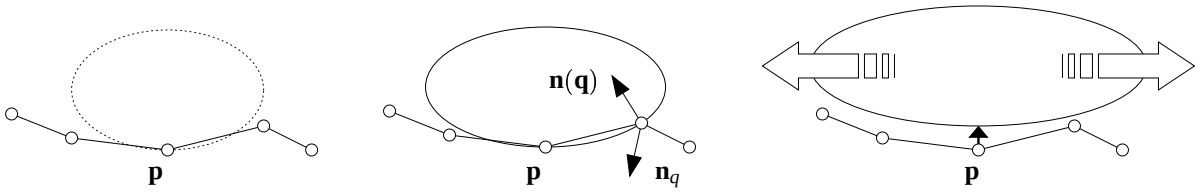

Figure 4: Left: Ellipsoid growth is impossible if triangles instead of vertices are testet for intersection. Middle: Ellipsoid normal $\mathbf{n}(\mathbf{q})$ and mesh normal $\mathbf{n}_q$ do not have to agree. Right: A small offset enables a significant ellipsoid growth.

$Q_1(r)$ as a 4×4 matrix:

$$\begin{pmatrix} 1 & 0 & 0 & -c_r[x] \\ 0 & 1 & 0 & -c_r[y] \\ 0 & 0 & 1 & -c_r[z] \\ -c_r[x] & -c_r[y] & -c_r[z] & c_r[x]^2+c_r[y]^2+c_r[z]^2-r^2 \end{pmatrix}$$

such that $[\mathbf{x};1]^T Q_1 [\mathbf{x};1] = 0$ implicitly defines the sphere.

Based on the normal vectors $\mathbf{n}_p = \mathbf{c}-\mathbf{p}$ and $\mathbf{n}_q := \mathbf{c}-\mathbf{q}$ we define another quadric Q_2 which is the product of the two tangent planes at $\mathbf{p}$ and $\mathbf{q}$ respectively.

$$Q_2: \qquad (\mathbf{x}-\mathbf{p})^T \mathbf{n}_p \cdot (\mathbf{q}-\mathbf{x})^T \mathbf{n}_q \;=\; 0$$

Again, we re-write Q_2 as a 4×4 matrix $Q_2 = (N+N^T)/2$ where N is given as

$$\begin{pmatrix} -\mathbf{n}_p[x]\,\mathbf{n}_q[x] & -\mathbf{n}_p[x]\,\mathbf{n}_q[y] & -\mathbf{n}_p[x]\,\mathbf{n}_q[z] & \mathbf{n}_p[x]\,(\mathbf{n}_q^T\mathbf{q}) \\ -\mathbf{n}_p[y]\,\mathbf{n}_q[x] & -\mathbf{n}_p[y]\,\mathbf{n}_q[y] & -\mathbf{n}_p[y]\,\mathbf{n}_q[z] & \mathbf{n}_p[y]\,(\mathbf{n}_q^T\mathbf{q}) \\ -\mathbf{n}_p[z]\,\mathbf{n}_q[x] & -\mathbf{n}_p[z]\,\mathbf{n}_q[y] & -\mathbf{n}_p[z]\,\mathbf{n}_q[z] & \mathbf{n}_p[z]\,(\mathbf{n}_q^T\mathbf{q}) \\ \mathbf{n}_q[x]\,(\mathbf{n}_p^T\mathbf{p}) & \mathbf{n}_q[y]\,(\mathbf{n}_p^T\mathbf{p}) & \mathbf{n}_q[z]\,(\mathbf{n}_p^T\mathbf{p}) & -(\mathbf{n}_p^T\mathbf{p})\,(\mathbf{n}_q^T\mathbf{q}) \end{pmatrix}$$

Since both quadrics Q_1 and Q_2 are touching the two points $\mathbf{p}$ and $\mathbf{q}$ with the corresponding normal vectors $\mathbf{n}_p$ and $\mathbf{n}_q$, the same is true for all linear combinations [1, 2]

$$Q_1 + \alpha Q_2 \tag{1}$$

Hence we can use the coefficient α to continue growing our ellipsoid. With increasing value α the ellipsoid will stretch within the wedge defined by the two tangent planes at $\mathbf{p}$ and $\mathbf{q}$ (cf. Fig 5). As long as α stays below some threshold α_0 the resulting quadric will still be of the ellipsoid type (and not a paraboloid or hyperboloid). We can check the type of the quadric during the algorithm by computing the eigenvalues of the upper left 3×3 sub-matrix: If all three eigenvalues are positive, the quadric is of ellipsoid type. To compute the optimal α we observe that an arbitrary point $\mathbf{x}$ lies within the ellipsoid defined by $Q_1+\alpha Q_2$ if and only if

$$\mathbf{x}^T(Q_1+\alpha Q_2)\mathbf{x} = \mathbf{x}^T Q_1 \mathbf{x} + \alpha \mathbf{x}^T Q_2 \mathbf{x} < 0$$

If α is positive, $\mathbf{x}^T Q_2 \mathbf{x} < 0$ is a necessary condition for $\mathbf{x}$ to lie within $Q_1+\alpha Q_2$ and hence we find

$$\alpha > -\frac{\mathbf{x}^T Q_1 \mathbf{x}}{\mathbf{x}^T Q_2 \mathbf{x}} =: \alpha(\mathbf{x}) \tag{2}$$

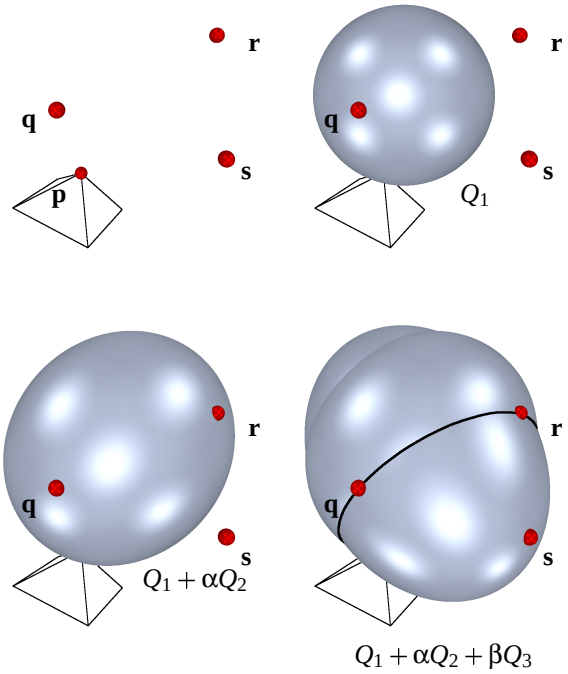

Figure 5: Ellipsoid growth strategy: Starting out at a single vertex $\mathbf{p}$ (upper left, on top of pyramid) the ellipsoid growth algorithm searches in normal direction for the biggest empty sphere Q_1 until it touches the vertex $\mathbf{q}$ (upper right). Next the sphere is extended to an ellipsoid by equation (1) until it touches a third point $\mathbf{r}$ (lower left). The three contact points $\mathbf{p},\mathbf{q},\mathbf{r}$ determine an ellipse E on the ellipsoid (lower right, black line) which serves as profile curve of the cylinder Q_3. Finally the ellipsoid is elongated along the cylinder by equation (3) until a fourth point $\mathbf{s}$ is touched (lower right).

The optimal α is easily computed as

$$\alpha_{opt} = \min\{\alpha(\mathbf{x}) | \mathbf{x}^T Q_2 \mathbf{x} < 0\}$$

By growing α we eventually find a third touching point $\mathbf{r}$. In cylindrical parts of the mesh model the three points $\mathbf{p}$, $\mathbf{q}$, and $\mathbf{r}$ have a high probability of lying in a plane which is approximately perpendicular to the cylinder axis (cf. Fig 5). Hence we can find a cigar-shaped fitting ellipsoid by growing into the direction orthogonal to the plane spanned by $\mathbf{p}$, $\mathbf{q}$, and $\mathbf{r}$.

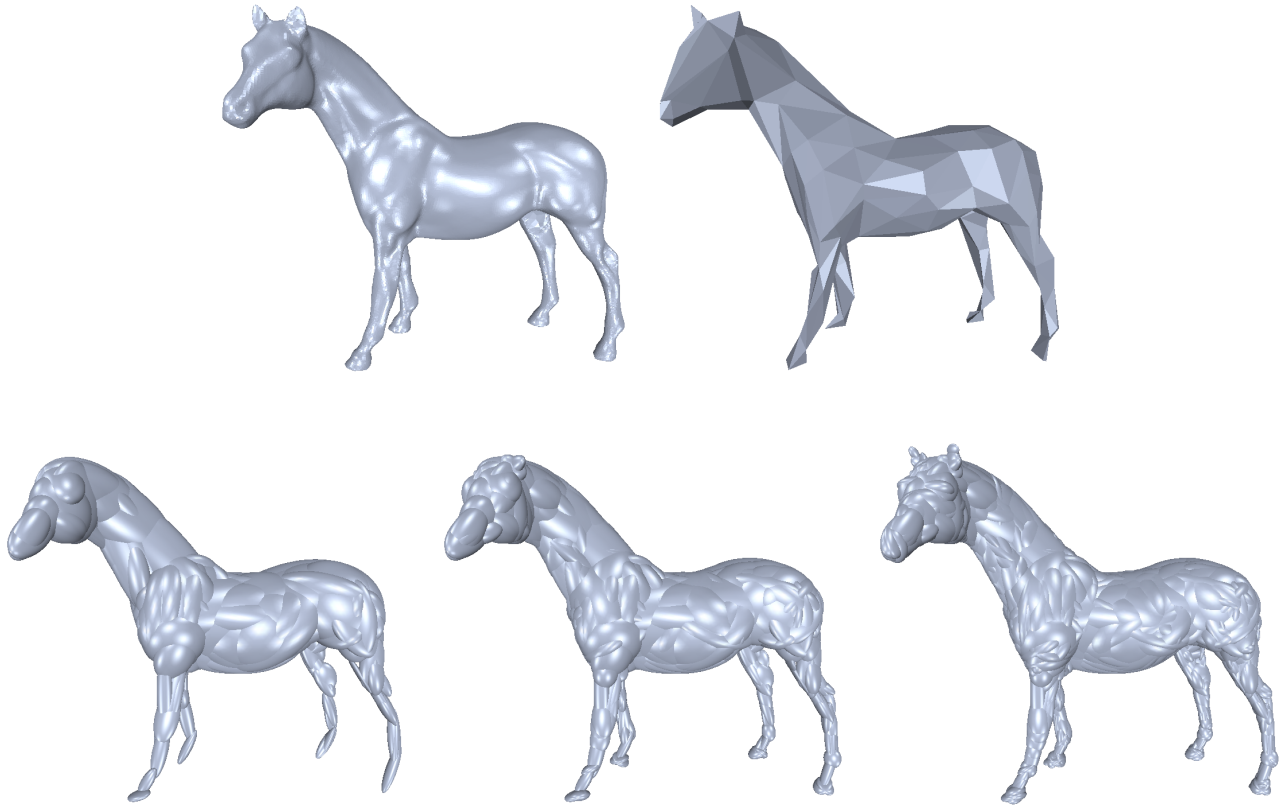

Figure 6: Upper left: Original horse model consisting of about 49000 vertices. Upper right: Horse model reduced to 200 vertices by Garland and Heckbert's error-quadric decimation algorithm [10]. Lower row, left to right: Ellipsoid approximations consisting of 100, 200 and 400 ellipsoids (3600, 7200 and 14400 bytes) respectively. Note the much improved detail comparing the 200 ellipsoid model to the memory-wise equivalent 200 vertices reduced model (7200 bytes).

The simplest solution would be to define another quadric Q_3 and to grow the current ellipsoid by finding the maximum β for which

$$Q_1 + \alpha Q_2 + \beta Q_3 \tag{3}$$

lies completely in the interior volume of the mesh and is still of the ellipsoid type. However, as it turns out, we cannot use the same technique as above since now we have *three* touching points and tangent planes and these interpolation constraints uniquely define a single quadric such that there is no more degree of freedom.

We therefore use another heuristic to define the third quadric Q_3: Let the plane that contains the three touching points $\mathbf{p}$, $\mathbf{q}$, and $\mathbf{r}$ be given in parameteric form by

$$\mathbf{x}(u,v) \;=\; u(\mathbf{q}-\mathbf{p}) + v(\mathbf{r}-\mathbf{p}) + \mathbf{p}. \tag{4}$$

The intersection of this plane with the quadric $\bar{Q} := Q_1 + \alpha Q_2$ is an ellipse E in the (u,v)-parameter space. E is given by a 3×3 matrix

$$E \;=\; \begin{pmatrix} \mathbf{q}-\mathbf{p} & \mathbf{r}-\mathbf{p} & \mathbf{p} \\ 0 & 0 & 1 \end{pmatrix}^T \bar{Q} \begin{pmatrix} \mathbf{q}-\mathbf{p} & \mathbf{r}-\mathbf{p} & \mathbf{p} \\ 0 & 0 & 1 \end{pmatrix}.$$

For this ellipse we can find the (u,v)-coordinates of the center by solving

$$E \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \;=\; 0$$

and we find the (u,v)-coordinates for the two principal axes and corresponding radii by computing the eigenvectors and eigenvalues of the upper left 2×2-submatrix of E. Finally we transform the center, axes, and radii from the (u,v) parameter space into world coordinates by using (4).

The above construction gives us a planar ellipse E in 3-space around the center $\mathbf{a}_0$ with two principal axes $\mathbf{a}_1$ and $\mathbf{a}_2$ and the corresponding radii r_1 and r_2. We define Q_3 to be the elliptic cylinder that contains E and has $\mathbf{a}_3 = \mathbf{a}_1 \times \mathbf{a}_2$ as its main axis. This cylinder contains all points $\mathbf{x}$ for which the following equation holds:

$$\frac{[(\mathbf{x}-\mathbf{a}_0)^T\,\mathbf{a}_1]^2}{r_1^2} + \frac{[(\mathbf{x}-\mathbf{a}_0)^T\,\mathbf{a}_2]^2}{r_2^2} \;=\; 1.$$

We can easily find a 4×4 matrix representation of Q_3 by factoring out the above equation. Once we have this representation, we can proceed the ellipsoid growing process with

respect to the β parameter in (3) analogously to the determination of the α parameter.

As we will demonstrate in Section 4, this ellipsoid growing procedure induces a good alignment of the principal axes to the main axes of the enclosed volume.

Remark When implementing the above ellipsoid fitting procedure there are some difficulties which arise from the fact that polygon meshes do not have a continuous normal field and hence a sphere or ellipsoid can touch the mesh at a surface point $\mathbf{q}$ while the surface normal $\mathbf{n}(\mathbf{q})$ and ellipsoid normal $\mathbf{n}_q$ do not necessarily agree. This can lead to a locking situation where the ellipsoid cannot grow anymore although some local ε modification would enable substantial growth (Figure 4).

To take this into account we modifed the above construction slightly in our implementation. Instead of using the exact touching points $\mathbf{p}$, $\mathbf{q}$, and $\mathbf{r}$ we use the points $\mathbf{p}'$, $\mathbf{q}'$, and $\mathbf{r}'$ which are obtained by shifting the original points by some small offset ε in normal direction into the interior of the mesh. This provides some additional flexibility for the determination of the coefficients α and β.

3.3 Ellipsoid selection

We apply the above construction to generate an extreme over-representation of the interior volume of a given polygon mesh. Since the computation of the growing ellipsoids is quite fast we can, e.g., compute a fitting ellipsoid for a moderately complex polygon mesh by placing a seed point on every vertex. Even though these ellipsoids will be strongly overlapping we do not increase the total memory requirements because, as we discussed in the introduction, a collection of n ellipsoids requires the same amount of storage as a triangle mesh with n vertices.

To reduce the redundancy of the ellipsoid decomposition we select the most significant ones from the set of candidates generated in the initialization phase. For this we use a rather simple greedy optimization. We start with the largest ellipsoid, i.e., with the ellipsoid that has the largest volume. Then in every step we include that ellipsoid which adds the most to the total volume of the overlapping ellipsoids. If there are several ellipsoids contributing approximately the same additional volume, we select the one that has the smallest minimum radius.

The only computationally expensive task in the ellipsoid selection process is the computation of the volume contribution of the individual ellipsoids. In every selection step we have the set of earlier included ellipsoids $E_1, \ldots, E_k$ and for every new candidate we have to estimate the difference volume

$$V = E_{\text{new}} \setminus (E_1 \cup \ldots \cup E_k).$$

We solved this problem by computing an approximation over a spatial grid of binary voxels. Whenever an ellipsoid is added to the selection set, we rasterize it and flag all voxels the fall inside it as *covered*. Then when we test a new candidate, we also rasterize the ellipsoid and count the number of uncovered voxels to obtain an estimate of the volume increase V. This greedy candidate selection scheme is easily implemented by a heap data structure. All ellipsoids are sorted in the heap according to their volume contribution. On popping an ellipsoid from the top of the heap we re-calculate the volume contribution of all the intersecting ellipsoids and rebuild the heap. This technique can further be improved by just flagging the volume contribution of intersecting ellipsoids as invalid. Each time an element is moved to the top of the heap, we check whether its volume contribution is still valid. If not, we re-calculate the volume and rebuild the heap. Using this "lazy evaluation" technique keeps redundant rasterization steps at a minimum und substantially speeds up the selection algorithm.

4 Results and future work

In this section we demonstrate the effectiveness of our approach by applying it to some of the standard polygon mesh datasets in the graphics commmunity.

4.1 Ellipsoid decomposition

Figure 6 shows the results of applying our algorithm to the well-known horse model. The original mesh contains about 49000 vertices. In order to avoid ellipsoid growth being restricted by the immediate neighboring vertices (1-ring), we smoothed the mesh in a preprocessing step. Then we randomly selected 2500 seed vertices and applied our ellipsoid growth algorithm ($\varepsilon = 0.1$) (see remark in section 3.2). The computation of the ellipsoids took only a few seconds. Finally we reorded the ellipsoids according to their volume contributions using a $100 \times 100 \times 100$ voxel grid as described in the previous section. This last step took about half an hour, but note that these are off-line preprocessing timings.

4.2 Robust transmission

Figure 7 depicts a typical reconstruction sequence. First the server transmits the base geometry and topology by sending a number of ellipsoids. The client may already render these ellipsoids as a first approximation of the final model. Then the client extracts a triangle mesh from the ellipsoid approximation, by e.g. a marching cubes algorithm [3] or by some kind of shrink-wrapping approach [8]. Because of the presmoothing and the offset ε, the resulting mesh will slightly shrink compared to the original mesh. Hence we have to adapt the reconstruction scheme in the following way: Each progressively received vertex is first projected onto the current mesh and then inserted into the closest triangle via a 1-to-3 split operation. A local edge flipping heuristic restores a generalized Delauny-criterion to preserve the tesselation

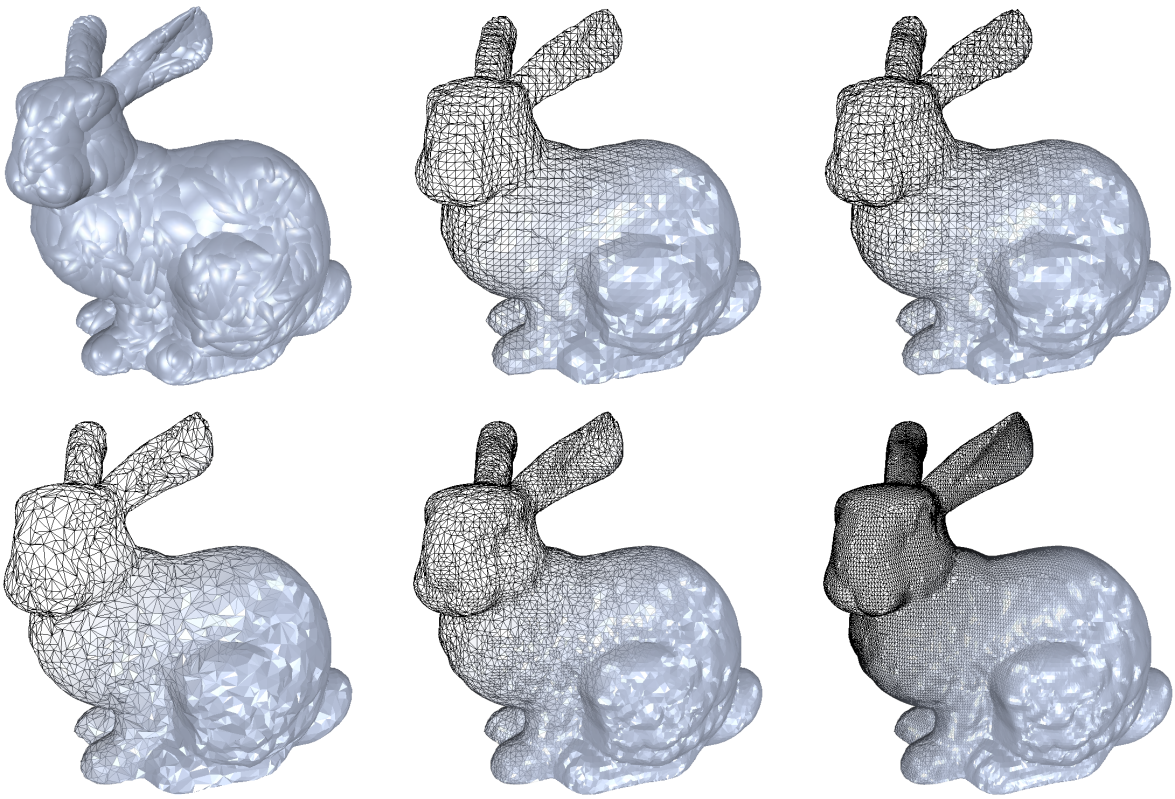

Figure 7: The sequence above depicts a typical reconstruction sequence: From the ellipsoid model of the bunny (upper left, 300 ellipsoids) the reconstruction algorithm extracts a marching cubes mesh (upper middle, 10000 vertices). The first chunk of received vertices is projected onto the mesh and then inserted into the mesh (upper right, 5000 vertices inserted). After that the points of the marching cubes mesh are removed by a decimation algorithm. The remaining vertices are shifted back to their original positions (lower left, 5000 vertices). Finally the remaining vertices are inserted into the mesh (lower middle, 10000 vertices inserted, lower right 30000 vertices inserted).

quality of the mesh. A detailed description of this scheme and its efficient implementation using space partitioning can be found in [5].

Since the vertices that were created by the marching cubes step are not part of the original geometry and they have to be removed as soon as a sufficient number of other vertices are received to maintain the coarse mesh geometry. The removal is done by a simple mesh decimation scheme, the remaining vertices are then shifted back to their original positions. Now the rest of the vertices can be inserted as before. Since no inter-vertex/face references are necessary, this scheme is robust in the sense that a certain percentage of ellipsoid/vertex loss is tolerated by the reconstruction algorithm.

4.3 Future work

We plan a more detailed investigation of various selection criteria to determine the optimal ellipsoid decomposition. The selection has to guarantee sufficient overlap between the ellipsoids but on the other hand it has to lead to a significant redundancy reduction.

A possible future application could be the automatic shape recognition. Deriving some statistical measures from the set of ellipsoids, like the average and standard deviation of ellipsoid radii and their ratio could be an identifyer that is invariant under various shape modifications. This will be a topic of future research.

5 Acknowledgement

This work was supported by the Deutsche Forschungsgemeinschaft under grant KO2064/1-1, “Distributed Processing and Delivery of Digital Documents”.

References

[1] W.Boehm, H.Prautzsch. *Geometric Concepts for Geometric Design.* AK Peters, 1994

[2] H.S.M.Coxeter. *Introduction to Geometry.* Wiley & Sons, 1961

[3] W.E.Lorensen, H.E.Cline. *Marching Cubes: a high resolution 3D surface reconstruction algorithm.* SIGGRAPH 87 Proceedings, 163–169

[4] L.Kobbelt, M.Botsch, U.Schwanecke, H.-P.Seidel. *Feature Sensitive Surface Extraction from Volume Data.* SIGGRAPH 01 Proceedings, 57–66

[5] S.Bischoff, L.Kobbelt, *Towards Robust Broadcasting of Geometry Data,* to appear in Computers & Graphics

[6] H.Hoppe, T.DeRose, T.Duchamp, J.McDonald, W.Stuetzle. *Surface Reconstruction from Unorganized Points.* SIGGRAPH 92 Proceedings, 71–78

[7] H.Hoppe. *Progressive Meshes.* SIGGRAPH 96 Proceedings, 99–108

[8] L.Kobbelt, J.Vorsatz, U.Labsik, H.-P.Seidel. *A Shrink Wrapping Approach to Remeshing Polygonal Surfaces.* Computer Graphics Forum, 18(3):199 – 130, 1999

[9] Lee, Sweldens, Schröder, Cowsar, Dobkin. *MAPS: Multiresolution Adaptive Parameterization of Surfaces.* SIGGRAPH 98 proceedings, 95–104

[10] M.Garland, P.S.Heckbert. *Surface Simplification Using Quadric Error Metrics.* SIGGRAPH 97 proceedings

[11] L.Kobbelt, S.Campagna, J.Vorsatz, H.-P.Seidel. *Interactive Multi-Resolution Modeling on Arbitrary Meshes.* SIGGRAPH 98 proceedings, 105–114

[12] P.Alliez, M.Desbrun.*Progressive encoding for lossless transmission of 3D meshes.* SIGGRAPH 01 proceedings

[13] G.Turk. *Re–tiling polygonal surfaces.* Computer Graphics SIGGRAPH 92 Proceedings, 55–64

[14] D.Zorin et al.*Subdivision for Modeling and Animation.* SIGGRAPH 00 Course Notes

[15] G.Taubin, A.Gueziec, W.Horn, F.Lazarus. *Progressive Forest Split Compression.* SIGGRAPH 98 Proceedings, 123–132

[16] J.C.Carr, T.J.Mitchell, R.K.Beatson, J.B.Cherrie, W.R. Fright, B.C.McCallum, T.R.Evans. *Reconstruction and Representation of 3D Objects With Radial Basis Functions.* SIGGRAPH 01 Proceedings

[17] E.Welzl. *Smallest enclosing disks.* New Results and New Trends in Computer Science, Springer, 1991, 359–370

[18] P.Lindstroem, G.Turk. *Fast and memory efficient polygonal simplification.* IEEE Visualization 98 Conference Proceedings, 279–286, 1998

[19] A.Ciampalini, P.Cignoni, C.Montani, R.Scopigno. *Multiresolution decimation based on global error.* The Visual Computer, 13(5):228-246, 1997

[20] D.Zorin, P.Schröder, W.Sweldens. *Interactive multiresolution mesh editing.* SIGGRAPH 97 Proceedings, 256-268

[21] L.Kobbelt et al. *Geometric Modeling Based on Polyognal Meshes.* Eurographics 2000 Tutorial

[22] D.Zorin et al. *Subdivision for Modeling and Animation*, SIGGRAPH 00 Course Notes

[23] C.Touma, C.Gotsman. *Triangle mesh compression.* Proceedings of Graphics Interface 98, 26-34