Eurographics Symposium on Geometry Processing (2007)
Alexander Belyaev, Michael Garland (Editors)

Fast Normal Vector Compression with Bounded Error

E. J. Griffith, M. Koutek and F. H. Post

Delft University of Technology, Netherlands

Abstract

We present two methods for lossy compression of normal vectors through quantization using "base" polyhedra. The first revisits subdivision-based quantization. The second uses fixed-precision barycentric coordinates. For both, we provide fast (de)compression algorithms and a rigorous upper bound on compression error. We discuss the effects of base polyhedra on the error bound and suggest polyhedra derived from spherical coverings. Finally, we present compression and decompression results, and we compare our methods to others from the literature.

Categories and Subject Descriptors (according to ACM CCS): I.3.0 [Computer Graphics]: General I.3.6 [Computer Graphics]: Methodology and Techniques E.4 [Data]: Coding and Information Theory

1. Introduction

Since Deering [Dee95] introduced geometry compression in 1995, it has been a popular research topic. Much work in the field has focused on mesh simplification, connectivity compression, and vertex position compression, but vertex attribute and normal compression have received less attention. Most work dealing with normal compression lacks rigorous analysis, and many techniques require reasonable amounts of computational resources for decompression.

We find compression necessary when visualizing time-dependent data due to the amount of data that must be read from disk during interactive visualization. However, little processing time is available for decompression because visualization often involves a large amount of interactive data processing, e.g. for volume rendering or particle tracing. Furthermore, decompression techniques requiring contextual knowledge are undesirable since they hinder GPU-based decompression and operating on subsets of the data.

Most existing normal compression techniques offer only image-based analysis of compression error, if an analysis is provided. However, image artifacts introduced by errors in normal directions are more visible in some areas than others, e.g. specular highlights and reflections. Thus, image quality assessment is scene, and often viewer, dependent, and it makes quantitative method comparisons difficult.

Here, we focus on compressing normal vectors, with the goals of bounded error and fast (de)compression. Oliveira and Buxton [OB06] expanded on index-based normal compression using subdivided "base" polyhedra and measured the resulting error. We extend and formalize these ideas, and we provide a method using barycentric coordinates when higher precision is necessary. We provide fast compression and decompression algorithms for both methods, where GPU-based decompression is possible. Furthermore, we are able to analytically derive upper bounds on the error for both methods. Using 16 bits per compressed normal, we are able to achieve an upper bound on the angular error of less than 0.57°, which presents almost no visual difference when used for rendering (Figure 1). Based on our error analysis, we suggest the use of new base polyhedra derived from spherical coverings [SHS97].

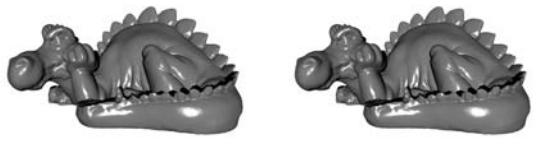
Figure 1: *Ray-traced images of the smoothed Phlegmatic Dragon with (left) and without (right) compressed normals.*

The remainder of this paper is organized as follows. We discuss related work in Section 2. In Section 3, we give an overview of our methods, and we cover the mathematical underpinnings. We describe and analyze the methods in Section 4. In Section 5, we present the results from our work. We conclude and discuss future work in Section 6.

2. Related Work

A major goal in geometry compression is overcoming transmission bottlenecks. Some work targets the RAM/GPU bottleneck (e.g. [Dee95, Cho97]). Other work focuses on network transmission (e.g. [TR98, TGHL98, TG98]). Peng et al. [PKK05] and Gotsman et al. [GGK02] give overviews of several techniques. A recent example from Purnomo et al. [PBCK05] quantizes all vertex data based on an image quality metric. Here we primarily list work specifically describing methods for quantizing normal vectors.

One alternative to quantization is entropy encoding. [GGK02] and [PKK05] list several techniques. Entropy encoding, however, requires contextual knowledge, which makes it less desirable when independent normal decompression is important, such as GPU-based implementations or working with data subsets. We note, though, that through careful quantization, it is possible to combine quantization with entropy encoding, such as in [Dee95, AKH06, IS02]. Thus, our techniques could be combined with entropy encoding at the expense of decompression speed.

Most normal quantization methods exploit the face symmetry of face-transitive polyhedra to generate a "uniform" distribution of points on the unit sphere. Deering [Dee95] uses warped spherical coordinates within the faces of a disdyakis dodecahedron. Ahn et al. [AKH06] generate regularly spaced points on the unit cube. The MPEG-4 BInary Format for Scenes (BIFS) [iso05], also used in QSplat [RL00], generates non-linearly warped sets of points on the unit cube. MPEG-4 3D Mesh Compression (3DMC) [iso04] uses a method described by Taubin et al. [THLR98] that uses representative points from the triangles of a recursively subdivided unit octahedron. Botsch et al. [BWK02] also subdivide the unit octahedron, but they project the result onto the unit sphere and use the face normal normals as the representative points. Oliveira and Buxton [OB06] further expanded on this idea by considering each of the Platonic solids as "base" polyhedra.

Several alternatives to the polyhedral methods exist. One approach is to use fixed precision spherical coordinates. Isenburg and Snoeyink [IS02] extract and quantize the smallest two components of each normal vector. Another possibility is to quantize normals using the points generated by the HEALPix method (Górski et al. [GHB*05]), which divides the unit sphere into regions of equal area.

One element lacking from these methods, however, is a rigorous error analysis of the quantization. Deering [Dee95] stated that compression errors should be at most 0.01 radians (0.573°), to prevent visible artifacts, but he presented no analysis to verify that his method satisfied this criterion. Aside from image-based metrics, the only error analysis we are aware of since then is from Oliveira and Buxton [OB06]. They measured the errors resulting from compressing normals from various models using quantized normals derived from subdividing the Platonic solids. We are able to provide a rigorous upper bound on the error resulting from quantization based on our methods and based on optimal quantization. Using these error bounds, we have experimented with even more base polyhedra, and we have improved, often significantly, on the error bounds from methods in the literature. Based on our comparisons with other methods, we found that our subdivision method is the only method currently able to satisfy Deering's criterion at 16-bit precision.

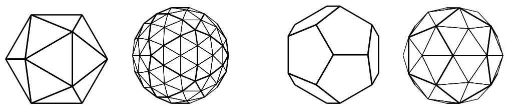
Figure 2: *Left: An icosahedron "refined" by subdividing each face and projecting the new vertices onto the unit sphere. Right: A dodecahedron and a dodecahedron triangulated by introducing a vertex at the center of each face.*

3. Overview and Underpinnings

We propose two methods for compressing normal vectors through quantization: a subdivision method and a barycentric method. Both methods refine convex polyhedra with vertices on the unit sphere by splitting each face into a set of similar triangles and projecting the new vertices onto the unit sphere (Figure 2, left). Quantization is is based on replacing normal vectors with the "closest" vertex from a refined polyhedron. Any non-triangular polyhedral faces are first triangulated by introducing a vertex at the face centroid, connecting each of the face's vertices to this new vertex, and projecting the new vertex onto the unit sphere (Figure 2, right). For the method details, see Sections 4.2 and 4.3.

3.1. Definitions and Notation

Here, we consider *unit normal vectors*, or *normals*, and points on the unit sphere to be interchangeable. Boldface will be used to indicate when point p on the unit sphere is being treated as normal vector $\mathbf{p}$. In general, we will refer to a normal as $\mathbf{n}$ and the quantization of $\mathbf{n}$ as $\mathbf{n}^\star$.

A *polyhedron*, $P = (F,V)$, has faces, $F = \{f_1,\ldots,f_\ell\}$, and vertices, $V = \{v_1,\ldots,v_m\}$. All vertices lie on the unit sphere, and each face, f_i, has a face normal $\mathbf{n}_{f_i}$. $F(P)$ are the faces of polyhedron P, and $V(P)$ are the vertices. A *triangulated polyhedron* has only triangular faces.

The shortest distance between two points, p and q, on the unit sphere is the angular distance between them:

$$\mathrm{dist}(p,q) = \arccos(\mathbf{p}\cdot\mathbf{q}). \tag{1}$$

The error, ε, between a normal, $\mathbf{n}$, and its quantization, $\mathbf{n}^\star$, is the angular distance between them:

$$\varepsilon = \mathrm{dist}(n,n^\star) = \arccos(\mathbf{n}\cdot\mathbf{n}^\star). \tag{2}$$

3.2. Normal Quantization

A normal is quantized in two steps. First, given normal, $\mathbf{n}$, and triangular faces, F, where $\mathbf{n}$ intersects at least one $f \in F$, one such $f \in F$ must be selected, as in Algorithm 1. Second, the closest vertex from that face to $\mathbf{n}$ is selected, as in Algorithm 2. This closest vertex is the quantized normal, $\mathbf{n}^\star$. Thus, given a triangulated polyhedron, P, $\mathbf{n}$ is quantized:

$$\mathbf{n}^\star = \text{QUANTIZE}(\text{FIND\_FACE}(F(P), \mathbf{n}), \mathbf{n}). \quad (3)$$

Algorithm 1 FIND_FACE($F, \mathbf{n}$)

```
$F \leftarrow f_1, \ldots, f_n$ {triangular faces}, $\mathbf{n}$ ←normal vector
$d \leftarrow -1$, $best \leftarrow 0$
for all $f_i \in F$ do
    $v_a, v_b, v_c \leftarrow$ vertices of $f_i$, counterclockwise
    $\mathbf{a} \leftarrow (\mathbf{v}_c \times \mathbf{v}_b)$, $\mathbf{b} \leftarrow (\mathbf{v}_a \times \mathbf{v}_c)$, $\mathbf{c} \leftarrow (\mathbf{v}_b \times \mathbf{v}_a)$
    if $(\mathbf{a} \cdot \mathbf{n}) > 0$ and $(\mathbf{b} \cdot \mathbf{n}) > 0$ and $(\mathbf{c} \cdot \mathbf{n}) > 0$ then
        return $f_i$ {$\mathbf{n}$ intersects $f_i$}
    else if $(\mathbf{n}_{f_i} \cdot \mathbf{n}) > d$ then
        $best \leftarrow i$, $d \leftarrow (\mathbf{n}_{f_i} \cdot \mathbf{n})$ {$\mathbf{n}$ is close to face normal $\mathbf{n}_{f_i}$}
return $f_{best}$
```

Algorithm 2 QUANTIZE($f, \mathbf{n}$)

```
$f \leftarrow$ triangular face, $\mathbf{n}$ ←normal vector
$v_a, v_b, v_c \leftarrow$ vertices of $f$
if $(\mathbf{v}_a \cdot \mathbf{n}) > (\mathbf{v}_b \cdot \mathbf{n})$ and $(\mathbf{v}_a \cdot \mathbf{n}) > (\mathbf{v}_c \cdot \mathbf{n})$ then
    return $\mathbf{v}_a$ {$\mathbf{n}$ is closest to $\mathbf{v}_a$}
else if $(\mathbf{v}_b \cdot \mathbf{n}) > (\mathbf{v}_c \cdot \mathbf{n})$ then
    return $\mathbf{v}_b$ {$\mathbf{n}$ is closest to $\mathbf{v}_b$}
else
    return $\mathbf{v}_c$ {$\mathbf{n}$ is closest to $\mathbf{v}_c$}
```

3.3. Error Bound

We will now prove an upper bound on the error for quantizing a given normal. We will show that, if Equation 3 is used for quantization, then, given a convex, triangulated polyhedron, P, and a normal, $\mathbf{n}$, and its quantization, $\mathbf{n}^\star$:

$$\text{dist}(n, n^\star) \leq \max(\{\text{dist}(v_i, n_{f_j}) | v_i \in f_j, f_j \in F(P)\}).$$

That is, the maximum error between any normal and its quantization is at most the maximum distance between a face's normal and its vertices. Thus, the error bound can be analytically determined by examining each polyhedral face.

First, we observe that every face of P represents a planar triangle and the spherical triangle that is its projection onto the unit sphere (Figure 3 left). Hence, the faces of P also represent a set of spherical triangles covering the unit sphere.

Next, we note that the vertices, v_a, v_b, and v_c, of triangular face, f_i, lie on the unit sphere. These three points define a circle on the unit sphere, which is the circumcircle of f_i (Figure 3 middle) and its spherical triangle. This circumcircle also defines a spherical cap, which lies "above" the plane defined by f_i (Figure 3 right). The line from the origin through the circle's center is normal to the plane in which the circle lies, which is the plane defined by f_i. Thus, the normal vector of f_i, $\mathbf{n}_{f_i}$ intersects f_i at its circumcenter. $\mathbf{n}_{f_i}$, also intersects the center of the spherical cap defined by the circle. Therefore, $\mathbf{n}_{f_i}$ is also the circumcenter for the spherical triangle and so $\text{dist}(n_{f_i}, v_a) = \text{dist}(n_{f_i}, v_b) = \text{dist}(n_{f_i}, v_c)$.

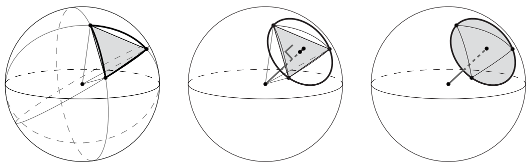

Figure 3: *Left: Spherical triangle and underlying planar triangle. Middle: The face normal of the planar triangle intersects triangle's circumcenter and the spherical triangle's circumcenter. Both triangles share the same circumcircle. Right: The circumcircle defines a spherical cap.*

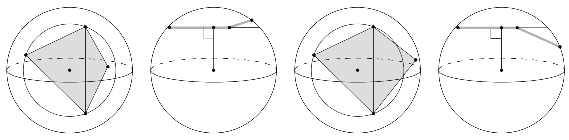

Figure 4: *Left: Front and top view of a vertex from one face lying on the spherical cap defined by another face. Right: Front and top view a vertex from one face lying outside the spherical cap defined by another face.*

To proceed, we introduce the following lemma.

Lemma 1 If triangulated polyhedron P is convex, then $\forall f_i \in F(P)$, $\text{dist}(n_{f_i}, v_j) \leq \text{dist}(n_{f_i}, v_k)$ with $v_j \in f_i$, $v_k \notin f_i$, and $v_k, v_j \in V(P)$.

Briefly, the lemma states that, for a convex, triangulated polyhedron, no vertex from one face may lie on the interior of the spherical cap defined by another face. See Figure 4.

Proof Given the triangulated polyhedron, P, select face, $f_i \in F(P)$. Suppose that there is some vertex, $v_k \in V(P)$, with $v_k \notin f_i$, that is closer to n_{f_i} than the vertices of f_i. Therefore, v_k lies on the interior of the spherical cap defined by f_i and is "above" the plane defined by f_i. Thus, there must exist a line segment between v_k and some vertex of f_i such that some part of the line segment passes "above" f_i. Since P is convex, nothing inside P can be "above" f_i so some part of the line segment must pass through the exterior of P. However, no line segment connecting two vertices of a convex polyhedron may pass through its exterior. Therefore, there can be no such vertex v_k if P is convex. $\square$

One result of this lemma is that the spherical triangulation defined by convex, triangulated polyhedron P is, in fact, a Delaunay triangulation of the unit sphere since no vertex from one face lies on the interior of the circumcircle (spherical cap) of another face (see, for example, [GO04]). Hence, the vertices of P are the "sites" of a spherical Voronoi diagram covering the unit sphere, and the face normals are the Voronoi vertices. This leads to an "ideal" error bound and an error bound specific to our normal compression methods.

The ideal error bound deals with normals quantized by replacing them with the closest vertices from P, and the more specific error bound deals with normals quantized using Equation 3. These bounds are equivalent, and we will use the ideal bound to compare our methods to others from the literature. We will now use Lemma 1 to first prove the ideal error bound and then to prove the specific error bound.

Theorem 1 Given a convex, triangulated polyhedron, P, a normal, $\mathbf{n}$, and the closest vertex, $v^\star \in V(P)$, to $\mathbf{n}$, then:

$$\mathrm{dist}(n, v^\star) \le \max(\{\mathrm{dist}(v_i, n_{f_j}) | v_i \in f_j, f_j \in F(P)\}). \quad (4)$$

Essentially, this theorem states that, if a normal is quantized by replacing it with the closest vertex from P, then the quantization error will be less than the maximum distance between a triangular face vertex and the normal for that face.

Proof From Lemma 1, P defines a spherical Voronoi diagram covering the unit sphere, where $V(P)$ are the sites and the face normals of P are the Voronoi vertices. Clearly, all Voronoi cells from a spherical Voronoi diagram are bounded, and, for bounded Voronoi cells, the farthest points on a cell from the cell's site are Voronoi vertices of the cell. The Voronoi vertices for the cell with vertex, $v \in V(P)$, as its site are the face normals, n_{f_k} from all faces, $f_k \in F(P)$, with $v \in f_k$. Thus, for any point, p, in that cell, we have: $\mathrm{dist}(p, v) \le \max(\{\mathrm{dist}(n_{f_k}, v) | v \in f_k\})$. Since, a given point, n, on the unit sphere, must lie on some Voronoi cell with some vertex, $v^\star \in V(P)$, as its site, then we know that: $\mathrm{dist}(n, v^\star) \le \max(\{\mathrm{dist}(v_i, n_{f_j}) | v_i \in f_j, f_j \in F(P)\})$. □

We now prove the bound on quantization with Equation 3.

Theorem 2 Given a convex, triangulated polyhedron, P, a normal, $\mathbf{n}$, and its quantization, $\mathbf{n}^\star$, from by Equation 3, then:

$$\mathrm{dist}(n, n^\star) \le \max(\{\mathrm{dist}(v_i, n_{f_j}) | v_i \in f_j, f_j \in F(P)\}). \quad (5)$$

Here, the idea is that quantizing normals by replacing them with the closest vertex from the triangular face they intersect gives the same error bound as Theorem 1, where normals are replaced by the closest vertex from P.

Proof From Lemma 1, P defines a spherical Voronoi diagram covering the unit sphere, where $V(P)$ are the sites and the face normals of P are the Voronoi vertices. We also know that each face, $f_i \in F(P)$, has n_{f_i} as the center of the circle/spherical cap circumscribing f_i. As the circumcenter, n_{f_i} is coincident with the intersection of the perpendicular bisectors of the sides of the spherical triangle defined by f_i. These bisectors define three regions in the spherical triangle, where all points from each region are closest to one triangle vertex and are at most the radius of the spherical cap away from the that vertex. See Figure 5 for examples of the planar case. The radius of the spherical cap is precisely the distance between n_{f_i} and any vertex of f_i. Thus, the maximum distance that any point on the unit sphere can be from the closest vertex of the spherical triangle that contains it, is the distance between the normal of that face and any of its vertices. Since P is triangulated and convex and all normals are quantized using Equation 3, the inequality in Theorem 2 holds. □

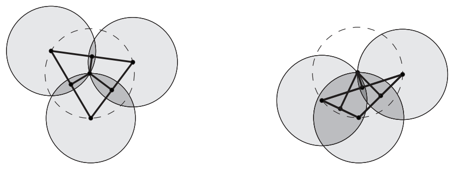
Figure 5: *Acute (left) and obtuse (right) triangles, with the perpendicular bisectors shown and copies of the circumscribing circle placed at the vertices.*

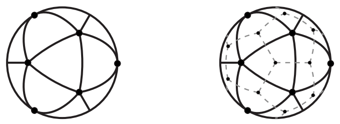
Figure 6: *Left: Spherical triangles defined by an icosahedron. Right: The same spherical triangles with the spherical Voronoi diagram illustrated.*

3.4. Quantization Optimality

Here, we will prove that, given a convex, triangulated polyhedron, P, Equation 3 will optimally quantize normal vector, $\mathbf{n}$, if P has only acute triangles. By optimal quantization, we mean that $n^\star$ will be the closest vertex from P to n.

Theorem 3 Given convex, triangulated polyhedron P, with all faces acute triangles, normal, $\mathbf{n}$, and its quantization, $\mathbf{n}^\star$, from by Equation 3, then:

$$\mathrm{dist}(n, n^\star) = \min(\{\mathrm{dist}(n, v_i) | v_i \in V(P)\}). \quad (6)$$

Proof From Lemma 1, P defines a spherical Voronoi diagram covering the unit sphere, where $V(P)$ are the sites and the face normals of P are the Voronoi vertices. We also know that, since the faces of P are acute triangles, the circumcenters of the faces lie within the faces themselves. Therefore, the circumcenters of the spherical triangles also lie within the spherical triangles. Thus, each spherical triangle contains one Voronoi vertex, and the Voronoi edges perpendicularly bisect the sides of the spherical triangles (Figure 6). Each spherical triangle lies in three Voronoi cells, the Voronoi sites of which are the spherical triangle's vertices. Therefore, no point in spherical triangle, $\triangle ABC$, is closer to a vertex from another spherical triangle than it is to one of the vertices of $\triangle ABC$. Thus, quantizing $\mathbf{n}$ using Equation 3 will result in the smallest possible distance between $\mathbf{n}^\star$ and $\mathbf{n}$. □

One property of this theorem is that Equation 3 is relatively robust to numerical error. The Voronoi cell for each vertex occupies a portion of all spherical triangles containing that vertex. Thus, even if a normal is near the edge of a spherical triangle and Algorithm 1 selects the incorrect face, Algorithm 2 will likely select the correct vertex. The cases where Algorithm 2 could select the incorrect polyhedron vertex are those when the normal is near a Voronoi edge. Since Voronoi edges are equidistant from Voronoi sites, i.e. the polyhedron vertices, then, in those situations, selecting the wrong vertex has little effect on the quantization error.

It is important to note that this theorem does not hold for convex, triangulated polyhedra containing obtuse triangles. In such polyhedra, some triangles will contain multiple Voronoi vertices, and, therefore, some regions of those triangles will be closer to a vertices from adjacent triangles. However, from Theorems 1 and 2, we know that, while some normals may be non-optimally quantized in such polyhedra, the upper bound on the error remains the same. In order to preserve the optimality of the quantization, though, we should use polyhedra containing only acute triangles.

As an aside, existing subdivision methods [OB06, BWK02, iso04] are relatively susceptible to error. In these methods, a normal is quantized by replacing it with a representative point from the face it intersects. However, there is no guarantee that a normal is closer to the representative point from that face than it is to those of neighboring faces, leading to possible non-optimal quantizations. This exacerbates potential errors from incorrect face selection.

3.5. Euler Characteristic

The Euler Characteristic for a polyhedron is defined as:

$$X = V - E + F$$

where X is the Euler Characteristic and V, E and F are respectively the numbers of vertices, edges, and faces of the polyhedron. $X = 2$ for the simply connected polyhedra we work with, and, since we only work with triangulated polyhedra, we have this useful relation:

$$V = \frac{F}{2} + 2 \tag{7}$$

4. Normal Compression

In this section, we provide the details over our two proposed normal compression methods: the subdivision method and the barycentric method. See Figure 7.

4.1. Bit precision and efficiency

Bit precision is the number of bits used to represent a compressed normal. In our approaches, all compressed normals from a given set are represented with the same bit precision.

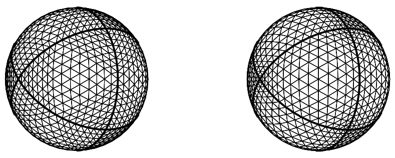

Figure 7: *Octahedron refined using the subdivision method (left) and the barycentric method (right).*

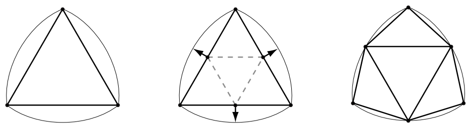

Figure 8: *Triangular face (left) subdivided by introducing vertices at edge midpoints (middle), which are projected onto the unit sphere (right).*

For a given bit precision, b, there are 2^b unique bit strings. Ideally, this would also mean quantization methods would generate 2^b unique normals. However, this is often not the case. In general, for a given bit precision and quantization scheme, a polyhedron generating more unique normals will have a lower error bound than one generating fewer unique normals (Table 1). Therefore, it is desirable to seek out combinations generating more unique normals.

4.2. Subdivision Method

The subdivision method generates a set of quantized normals by recursively subdividing the faces of a triangulated polyhedron. At each subdivision level, each polyhedral face is subdivided by introducing new vertices at the midpoints of face edges and projecting the new vertices onto the unit sphere. See Figure 8. For a triangulated polyhedron, P_0, polyhedra, $P_1, \ldots, P_n$, represent different levels of subdivision.

Before a set of normals can be compressed, the base polyhedron, P_0, is first refined to a certain subdivision level, P_ℓ, and every vertex from P_ℓ is assigned a unique ID number. This defines the normal table. Once the table is constructed, normals are quantized using Algorithm 3. See Figure 9. If the base polyhedron, P_0, and all refined polyhedron, P_i, are convex, then the guarantee from Theorem 3 will hold, and the compression process will always find the closest polyhedron vertex for each normal.

Given a base polyhedron, P_0, and a level of subdivision, s, the number of unique quantized normals generated by the subdivision method is $|V(P_s)|$. Since P_s has $(4^s)|F(P_0)|$ faces, we can use Equation 7 to compute this:

$$|V(P_s)| = \frac{(4^s)|F(P_0)|}{2} + 2. \tag{8}$$

However, the normal table can contain at most 2^b entries at bit precision, b. Therefore, we must calculate the maximum

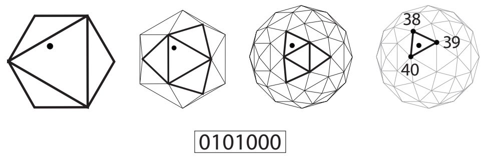

Figure 9: *A normal quantized by recursively finding the face it intersects and then selecting the closest vertex from the final face. The quantized normal is a normal table index.*

Algorithm 3 COMPRESS_SUBDIVISION($P, \mathbf{n}, \ell$)

```
$P \leftarrow$ triangulated polyhedron {vertices have unique ID}
$\mathbf{n} \leftarrow$normal vector
$\ell \leftarrow$ maximum level of subdivision
$f \leftarrow$ FIND_FACE($F(P), \mathbf{n}$)
for $i = 1 \ldots \ell$ do
    $f_a, f_b, f_c, f_d \leftarrow$ subdivided faces of $f$
    $f \leftarrow$ FIND_FACE($\{f_a, f_b, f_c, f_d\}, \mathbf{n}$)
$v \leftarrow$ QUANTIZE($f, \mathbf{n}$)
return $v$.id
```

subdivision level, ℓ, for P_0 such that $|V(P_\ell)| \leq 2^b$ using the following equation derived from Equation 8:

$$\ell = \lfloor \log_4(2^{b+1} - 4) - log_4|F(P_0)| \rfloor. \tag{9}$$

Given M normal vectors and base polyhedron P_0 refined to subdivision level ℓ, decompression and compression complexities are as follows. Decompression only requires a normal table look-up, and thus is $O(M)$. The normal table can be constructed in $O((4^\ell)|F(P_0)|)$ time. For large values of M and small values of ℓ and $|F(P_O)|$, this time is negligible. Compression of the normals takes $O(M(|F(P_0)| + 4\ell))$ time.

The subdivision method offers two chief advantages. First, decompression is trivial, and thus incurs almost no computational overhead. Second, with careful polyhedron selection, it is able generate almost the maximum number of unique normals, which generally results in a lower error bound. The normal table must be kept in memory, however, which is impractical for sufficiently large normal tables.

4.3. Barycentric Method

The barycentric method is based on refining faces of a base triangulated polyhedron by computing fixed-precision barycentric coordinates. This divides the face into a set of similar triangles, and the newly introduced vertices are then projected onto the unit sphere. See Figure 10. Unlike the subdivision method, the barycentric method is not recursive, and it does not generate any intermediate polyhedra between the base polyhedron, P, and the refined polyhedron, P_r. In fact, P_r is never explicitly generated. Instead, faces from P_r are computed as necessary.

The compressed normals are represented as bit strings

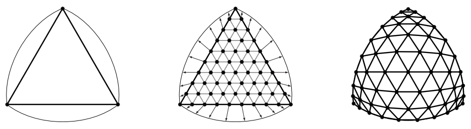

Figure 10: *Triangular face (left) refined by introducing vertices at barycentric coordinates of fixed-precision (middle), which are projected onto the unit sphere (right).*

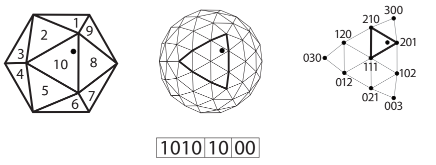

Figure 11: *A normal quantized by finding the face it intersects and computing barycentric coordinates. The quantized normal is a face index and two fixed-precision coordinates.*

consisting of three integers (Figure 11). The first identifies the face from the base polyhedron, which the quantized normal intersects. The next two integers represent the u and v barycentric coordinates. Since the barycentric coordinates u, v and w sum to 1, there is no need to explicitly store w. The compression algorithm is presented in Algorithm 4. If both the base polyhedron, P, and the refined polyhedron, P_r, are convex, then the guarantee from Theorem 3 will hold, and the compression process will always find the closest polyhedron vertex for each normal vector.

The bit precision for the barycentric method is divided between the face identifier and the (equal precision) integer u and v coordinates. A base polyhedron, P, with $|F(P)|$ faces, requires a minimum bit precision of $\log_2|F(P)| + 2$. If u and v are each c bits long, then the number of unique vertices in P_r can be computed with this equation:

$$|V(P_r)| = \frac{(2^c - 1)^2 |F(P)|}{2} + 2 \tag{10}$$

Given M normal vectors and a base polyhedron P, decompression and compression complexities are as follows. Decompression requires a face look-up, computing a weighted sum of the vertices and normalizing the result. This requires constant time, and thus decompression is $O(M)$. Compression of the normals takes $O(M|F(P_0)|)$ time since the face from the base polyhedron containing the normal must be found before the barycentric coordinates can be computed.

The primary advantage of the barycentric method is its fast decompression algorithm, which does not require a large normal table. In comparison with the subdivision method, the barycentric method has two disadvantages. For an equivalent bit precision, the barycentric method will generate fewer unique normals. This is not a significant disadvantage though, since, for higher bit precisions, it is impractical to

Algorithm 4 COMPRESS_BARYCENTRIC($P, \mathbf{n}, p$)

```
$P \leftarrow$ triangulated polyhedron {faces have unique ID}
$\mathbf{n} \leftarrow$normal vector
$p \leftarrow$ barycentric coordinate bit precision
$f \leftarrow$ FIND_FACE($F(P), \mathbf{n}$)
$v_a, v_b, v_c \leftarrow$ vertices of $f$
$\hat{n} \leftarrow$ intersection point between $\mathbf{n}$ and $f$
$d \leftarrow 2^p - 1$
$u, v, w \leftarrow$ barycentric coordinates of $\hat{n}$ in $f$
**if** $d - (\lfloor u*d \rfloor + \lfloor v*d \rfloor + \lfloor w*d \rfloor) = 2$ **then**
    $u_x \leftarrow \frac{\lceil u*d \rceil}{d}, v_x \leftarrow \frac{\lceil v*d \rceil}{d}, w_x \leftarrow \frac{\lfloor w*d \rfloor}{d}$
    $u_y \leftarrow \frac{\lceil u*d \rceil}{d}, v_y \leftarrow \frac{\lfloor v*d \rfloor}{d}, w_y \leftarrow \frac{\lceil w*d \rceil}{d}$
    $u_z \leftarrow \frac{\lfloor u*d \rfloor}{d}, v_z \leftarrow \frac{\lceil v*d \rceil}{d}, w_z \leftarrow \frac{\lceil w*d \rceil}{d}$
**else if** $d - (\lfloor u*d \rfloor + \lfloor v*d \rfloor + \lfloor w*d \rfloor) = 1$ **then**
    $u_x \leftarrow \frac{\lfloor u*d \rfloor}{d}, v_x \leftarrow \frac{\lfloor v*d \rfloor}{d}, w_x \leftarrow \frac{\lceil w*d \rceil}{d}$
    $u_y \leftarrow \frac{\lfloor u*d \rfloor}{d}, v_y \leftarrow \frac{\lceil v*d \rceil}{d}, w_y \leftarrow \frac{\lfloor w*d \rfloor}{d}$
    $u_z \leftarrow \frac{\lceil u*d \rceil}{d}, v_z \leftarrow \frac{\lfloor v*d \rfloor}{d}, w_z \leftarrow \frac{\lfloor w*d \rfloor}{d}$
**else**
    **return** $f$.id, $u*d$, $v*d$
$\mathbf{x} \leftarrow u_x\mathbf{v}_a + v_x\mathbf{v}_b + w_x\mathbf{v}_c$
$\mathbf{y} \leftarrow u_y\mathbf{v}_a + v_y\mathbf{v}_b + w_y\mathbf{v}_c$
$\mathbf{z} \leftarrow u_z\mathbf{v}_a + v_z\mathbf{v}_b + w_z\mathbf{v}_c$
$\mathbf{n}^\star \leftarrow$ QUANTIZE($\{\frac{\mathbf{x}}{|\mathbf{x}|}, \frac{\mathbf{y}}{|\mathbf{y}|}, \frac{\mathbf{z}}{|\mathbf{z}|}\}, \mathbf{n}$)
**if** $\mathbf{n}^\star = \mathbf{x}$ **then**
    **return** $f$.id, $u_x*d$, $v_x*d$
**else if** $\mathbf{n}^\star = \mathbf{y}$ **then**
    **return** $f$.id, $u_y*d$, $v_y*d$
**else**
    **return** $f$.id, $u_z*d$, $v_z*d$
```

keep a normal table in memory. Secondly, the distribution of quantized normals generated by the barycentric method tends to distort more in the center of the base polyhedron faces, whereas the subdivision method produces more homogeneous distributions. For smaller base polyhedron faces, this tends to mean that the subdivision method generates more even distributions, which usually result in lower maximum errors. See Figure 7.

4.4. Base Polyhedron Selection

Up to this point, we have not discussed the selection of a base polyhedron for these methods, but this selection has a significant impact on the error bound. While we are able to analytically determine the upper bound on the error for a given polyhedron, different base polyhedra will result in different refined polyhedra, which will have different upper bounds on the quantization error. We examined a variety of base polyhedra to determine which offered the lowest upper bound. We looked at the Platonic solids, used by Oliveira and Buxton [OB06], Archimedean solids, Catalan solids, and polyhedra generated by computing the convex hulls of spherical coverings from Sloane et al. [SHS97]. To compute the convex hulls, we used the QHull software [BDH96]. For all the polyhedra we considered, we triangulated any non-triangular faces (Figure 2 right), and we projected all vertices onto the unit sphere.

We defined five criteria for a polyhedron to be considered suitable for use with our methods.

1. The polyhedron must be convex.
2. The faces of the polyhedron must all be acute triangles.
3. The polyhedron must have 256 or fewer faces.
4. The polyhedron must remain convex when refined.
5. Faces of refined polyhedra must also be acute triangles.

In our experience, the fourth and fifth criteria are met if the angles of the spherical triangles defined by the base polyhedron faces are all less than or equal to 90°. We have been able to verify this numerically, but we have not yet been able to provide a formal proof.

From our set of polyhedra, these criteria ruled out several of the Catalan solids and, notably, the tetrahedron, which fails to satisfy criterion 4. We then tested the remaining polyhedra to see which yielded the lowest maximum compression error at different bit precisions for both methods. Results from the polyhedra in Figure 12 are in Table 1. We found that carefully chosen spherical coverings from Sloane et al. yield a lower maximum error in than the Platonic, Archimedean, or Catalan solids in each of the four cases.

Furthermore, we found that the better spherical coverings for the barycentric method performed at least as well as some of the poorer polyhedra from the subdivision method for equivalent bit precisions. The best spherical covering from the barycentric method with 24-bit precision reduces the maximum error by an order of magnitude over the best spherical covering for the subdivision method at 16-bit precision.

5. Results

In this section, we present results related to our compression and decompression algorithms. First, we present details about the performance of our compression and decompression methods. Secondly, we compare our method to various methods from the literature.

5.1. Performance

We tested the performance of our methods on six well-known models. For models lacking normals, normals were generated using the PLY tools provided by the Stanford 3D Scanning Repository[†]. Degenerate normals were ignored. We recorded the running time for both compression and decompression for the subdivision and barycentric models using a 3.0 GHz Pentium 4 machine. In our timings, we only

[†] http://graphics.stanford.edu/data/3Dscanrep/

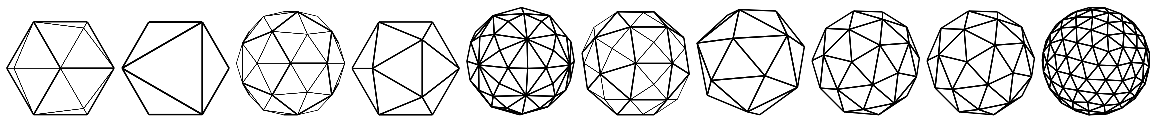

Figure 12: *Ten (triangulated) base polyhedra. From left to right: Cube, Octahedron, Dodecahedron, Icosahedron, Disdyakis Triacontahedron, Rhombicuboctahedron, Spherical Coverings 1 through 4.*

			Subdivision				Barycentric			
			12-Bit Precision		16-Bit Precision		16-Bit Precision		24-Bit Precision	
Polyhedron	F	V	$\|N\|$	ε_{max}	$\|N\|$	ε_{max}	$\|N\|$	ε_{max}	$\|N\|$	ε_{max}
Cube*	24	14	3,074	2.54°	49,154	0.636°	11,534	1.35°	3,133,454	0.0821°
Octahedron	8	6	1,026	5.05°	16,386	1.27°	15,878	1.29°	4,186,118	0.0792°
Dodecahedron*	60	32	1,922	2.99°	30,722	0.747°	28,832	0.773°	7,833,632	0.0469°
Icosahedron	20	12	2,562	2.73°	40,962	0.684°	9,612	1.41°	2,611,212	0.0857°
Disdyakis Triacontahedron	120	62	**3,842**	2.38°	**61,442**	0.595°	13,502	1.29°	3,901,502	0.0761°
Rhombicuboctahedron*	80	42	2,562	3.24°	40,962	0.811°	9,002	1.73°	2,601,002	0.204°
Spherical Covering 1	30	17	**3,842**	**2.24°**	**61,442**	**0.561°**	14,417	1.16°	3,916,817	0.0703°
Spherical Covering 2	126	65	**4,034**	**2.18°**	**64,514**	**0.546°**	14,177	1.10°	4,096,577	0.0649°
Spherical Covering 3	64	34	2,050	2.94°	32,770	0.735°	**30,754**	**0.759°**	**8,355,874**	**0.0461°**
Spherical Covering 4	256	130	2,050	2.93°	32,770	0.733°	**28,802**	**0.764°**	**8,323,202**	**0.0450°**

Table 1: *Base polyhedra (Figure 12) with face (F) and and vertex (V) counts. Unique normals, $|N|$, and error upper bound, ε_{max}, are listed for each at 12 and 16-bit precisions (subdivision method) and at 16 and 24-bit precisions (barycentric method). Polyhedra marked with an asterisk were triangulated for use with our methods. The best polyhedra in each column are bolded.*

timed the performance of compression and decompression on data resident in main memory, and we did not include the time necessary to load the required data from disk. For each method, we recorded both the maximum error found between a normal vector and the compressed normal during quantization and the average of all the errors. For the subdivision method, we used Spherical Covering 1 (Figure 12, Table 1) as the base polyhedron, and, for the barycentric method, we used Spherical Covering 3. We chose these coverings over 2 and 4 for performance reasons. The error bounds are only slightly higher, and, since the compression times for these schemes are linear in the number of faces in the base polyhedron, the compression times are lower with these base polyhedra.

Table 2 lists the performance of our subdivision and barycentric methods on six well-known models. In all cases, the maximum recorded error remained below the analytically derived error upper bound, and the average recorded error was slightly more than half of the maximum error. The compression and decompression times for the barycentric method follow a clear linear trend that increases with the number of normal vectors. The compression times for the subdivision method shows a non-linear trend due to the overhead of explicitly constructing the refined polyhedron from the base polyhedron. For larger numbers of normal vectors, though, the subdivision method proves to be faster than the barycentric method.

The memory requirements for compression and decompression are not listed in the table, but they are quite low. For compression with the subdivision method, the normal table of about 65,536 12-byte normals must be generated and kept in memory. This table is at most 768 kilobytes. For decompression with the subdivision method, this table must also be kept in memory. However, all compressed normals share the same normal table so only one copy of this table need be kept in memory. For the barycentric method, the base polyhedron must be kept in memory, which means that each vertex must be stored as well as a list of faces, which index into the vertices. For Spherical Covering 3, this is totals 600 bytes for 34 unique 12 byte vertices and 64 faces consisting of three one byte indices.

5.2. Method Comparison

Due to the lack of error analysis from existing methods, it is difficult to compare our method to those from the literature and other spherical point distributions. Here, we attempt a comparison of our method with those of Oliveria and Buxton [OB06] (PNORMS), the MPEG-4 3D Mesh Coding [iso04, THLR98] (3DMC), Botsch et al. [BWK02] (Octahedron), Deering [Dee95] (Deering), the MPEG-4 BInary Format for Scenes [iso05, RL00] (BIFS), Isenburg and Snoeyink [IS02] (Projection), and Ahn et al. [AKH06] (Cube). We also include a comparison with the point distributions generated by fixed precision spherical coordinates and by the HEALPix method [GHB*05]. For the PNORMS method, we used an icosahedron as a base polyhedron, and, for our method, we used the subdivision method on Spherical Covering 1 (Figure 12, Table 1). Figure 13 illustrates

		16-Bit Precision Subdivision Spherical Covering 1				24-Bit Precision Barycentric Spherical Covering 3			
Model	Normals	t_{com}	t_{dec}	$\max(\varepsilon)$	$\mathrm{mean}(\varepsilon)$	t_{com}	t_{dec}	$\max(\varepsilon)$	$\mathrm{mean}(\varepsilon)$
Stanford Bunny	35,947	$0.798s$	$0.0002s$	$0.548°$	$0.312°$	$0.180s$	$0.008s$	$0.0458°$	$0.0267°$
Armadillo	172,974	$1.60s$	$0.0012s$	$0.557°$	$0.312°$	$0.92s$	$0.042s$	$0.458°$	$0.0266°$
Happy Buddha	543,652	$3.43s$	$0s.0036$	$0.556°$	$0.312°$	$2.80s$	$0.131s$	$0.0459°$	$0.0267°$
Phlegmatic Dragon	703,018	$3.53s$	$0.0040s$	$0.554°$	$0.310°$	$3.56s$	$0.169s$	$0.0448°$	$0.0267°$
David ($2mm$)	6,924,951	$30.3s$	$0.042s$	$0.560°$	$0.311°$	$34.0s$	$1.64s$	$0.0460°$	$0.0267°$
Lucy	14,027,872	$60.8s$	$0.086s$	$0.556°$	$0.311°$	$70.0s$	$3.36s$	$0.0460°$	$0.0267°$

Table 2: *Compression, t_{com}, and decompression, t_{dec}, times for the normals from various known models. Average,* $\mathrm{mean}(\varepsilon)$*, and maximum,* $\max(\varepsilon)$*, error recorded during compression are listed. Normals were compressed in both methods using polyhedra derived from spherical coverings (See Section 4.4). In all cases,* $\max(\varepsilon)$ *remained below ε_{max} (Table 1).*

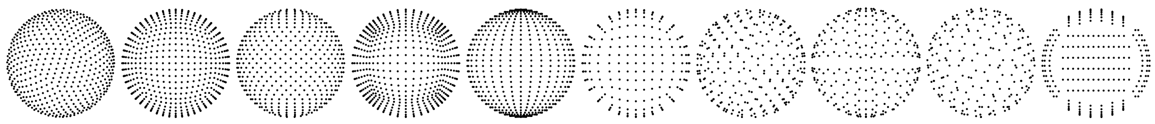

Figure 13: *Ten sets of quantized normals generated at 10 bits of precision. From left to right (corresponding with Table 3): our method, BIFS, HEALPix, Cube, Spherical Coordinates, Deering, Octahedron, 3DMC, PNORMS, and Projection.*

the spherical point distribution generated by each method at 10-bit precision.

In our test, we used each method to generate the set of all unique quantized normals at 16-bit precision. Using QHull [BDH96], we then converted this point set into a triangulated, convex polyhedron. Taking advantage of Theorem 1, we are able to use this polyhedron to analytically determine an upper bound on the error for quantizing normals based on that set of points. Note that this error bound assumes that the actual normal quantization process will not exceed the error bound from optimal quantization. The actual error upper bound is likely to be higher since the methods offer no such guarantees.

Table 3 contains the results of our comparisons. In general, the methods that had a low error bound generated near the maximum number of unique normals and produced relatively "uniform" distributions on the unit sphere. The error bounds from our subdivision method were consistently lower than those of the other methods (Tables 1 and 2). Interestingly, the method proposed by Deering, the Octahedron method, the 3DMC method and even the PNORMS method all have a higher error bound than using fixed precision spherical coordinates. Our barycentric method also has a lower bound than fixed precision spherical coordinates, but its error bound is higher than that of BIFS and HEALPix.

Method	Unique Normals	ε_{max}
Our Subdivision Method	61,442	0.561°
BIFS [iso05]	64, 896	0.612°
HEALPix [GHB*05]	49,154	0.682°
Cube [AKH06]	64,896	0.779°
Spherical Coordinates	65,026	0.787°
Deering [Dee95]	24,578	1.26°
Octahedron [BWK02]	32,768	1.26°
3DMC [iso04]	32,258	1.27°
PNORMS [OB06]	27,200	1.36°
Projection [IS02]	41,712	1.38°

Table 3: *Number of unique normals generated and ε_{max} for each method (Figure 13) at 16-bit precision.*

6. Conclusions and Future Work

We presented two methods for lossy normal vector compression through quantization based on refining base polyhedra. The first revises the existing subdivision methods, using a table of normals comprised of the refined polyhedra vertices. The second method quantizes normals by computing fixed-precision barycentric coordinates within base polyhedron faces. We provided fast compression and decompression algorithms with low memory requirements for both methods, and we tested their performance on various known models.

We used the property that our quantized normals are vertices of refined polyhedra to introduce three results. First, we showed an analytical upper bound on error for a normal vector optimally quantized using a convex, triangulated polyhedron. Second, we showed that this error bound also holds for quantization using our methods. Third, we showed that, if all the faces of the base polyhedron and refined polyhedra are acute, then our methods will optimally quantize normals.

We performed several comparisons with the error bounds we derived. First, we analyzed our methods using several base polyhedra. We found that base polyhedra derived from spherical coverings from Sloane et al. [SHS97] gave the lowest error bound. We also found that the subdivision method gives a lower error bound than the barycentric method at the same bit precision. Thus, when a normal table can reasonably be kept in memory, the subdivision method is preferrable. Next, we were able to compute the error upper bound, assuming optimal quantization, for a variety of methods from the literature at 16 bit precision. We showed that our subdivision method had the lowest error bound out of the methods we tested, and that several existing methods had higher error bounds than using fixed precision spherical coordinates. Further, our subdivision method was the only method, at 16-bit precision, to satisfy Deering's criterion that the error be at most 0.01 radians [Dee95].

In the future, there are a variety of objectives we would like to meet. We would like to refine the compression process so that base polyhedra with more faces can be used. We plan to implement GPU versions of the decompression algoriths. We also plan to extend this work for use on arbitrary vectors. Lastly, we would like to be able to further formalize the error bounds so that we analyze base polyhedra more quickly.

Acknowledgements

We would like to thank the Netherlands Organization for Scientific Research (NWO) for providing project funding.

The Bunny, Happy Buddha, Armadillo and Lucy models are from the Stanford 3D Scanning Repository. The David model is from the Digital Michaelangelo project. The Phlegmatic Dragon is from the Academy of Sciences of the Czech Republic and the Czech Technical University in Prague.

References

[AKH06] AHN J.-H., KIM C.-S., HO Y.-S.: Predictive compression of geometry, color and normal data of 3-D mesh models. *IEEE Trans. on Circuits and Systems for Video Technology 16*, 2 (2006), 291–299.

[BDH96] BARBER C. B., DOBKIN D. P., HUHDANPAA H.: The quickhull algorithm for convex hulls. *ACM Trans. on Mathematical Software 22*, 4 (1996), 469–483.

[BWK02] BOTSCH M., WIRATANAYA A., KOBBELT L.: Efficient high quality rendering of point sampled geometry. In *Proc. EG Rendering Workshop* (2002), pp. 53–64.

[Cho97] CHOW M. M.: Optimized geometry compression for real-time rendering. In *Proc. Visualization* (1997), pp. 347–354, 559.

[Dee95] DEERING M.: Geometry compression. In *Proc. of ACM SIGGRAPH* (1995), pp. 13–20.

[GGK02] GOTSMAN C., GUMHOLD S., KOBBELT L.: Simplification and compression of 3D meshes. In *Tutorials on Multiresolution in Geometric Modelling* (2002), pp. 319–362.

[GHB*05] GÓRSKI K., HIVON E., BANDAY A., WANDELT B., HANSEN F., REINECKE M., BARTELMANN M.: HEALPIX: A framework for high resolution discretization, and fast analysis of data distributed on the sphere. *Astrophysical Journal 622*, 2 (2005), 759–771.

[GO04] GOODMAN J., O'ROURKE J.: *Handbook of Discrete and Computational Geometry*. Chapman & Hall/CRC, 2004.

[IS02] ISENBURG M., SNOEYINK J.: Coding with ASCII: compact, yet text-based 3D content. In *Proc. 3D Data Processing* (2002), pp. 609–616.

[iso04] *ISO/IEC 14496-2:2004 Information technology — Coding of audio-visual objects — Part 2: Visual*. International Organization for Standardization, Geneva, Switzerland, 2004.

[iso05] *ISO/IEC 14496-11:2005 Information technology — Coding of audio-visual objects — Part 11: Scene description and application engine*. International Organization for Standardization, Geneva, Switzerland, 2005.

[OB06] OLIVEIRA J. F., BUXTON B. F.: PNORMS: Platonic derived normals for error bound compression. In *Proc. ACM VRST* (2006), pp. 324–333.

[PBCK05] PURNOMO B., BILODEAU J., COHEN J. D., KUMAR S.: Hardware-compatible vertex compression using quantization and simplification. In *Proc. Graphics Hardware* (2005), pp. 53–61.

[PKK05] PENG J., KIM C. S., KUO C. C. J.: Technologies for 3D mesh compression: A survey. *Journal of Visual Communication and Image Representation 16*, 6 (2005), 688–733.

[RL00] RUSINKIEWICZ S., LEVOY M.: QSplat: A multiresolution point rendering system for large meshes. In *Proc. ACM SIGGRAPH* (2000), pp. 343–352.

[SHS97] SLOANE N., HARDIN R., SMITH W.: Spherical coverings. Published electronically at http://www.research.att.com/∼njas/coverings, 1997.

[TG98] TOUMA C., GOTSMAN C.: Triangle mesh compression. In *Proc. Graphics Interface* (1998), pp. 26–34.

[TGHL98] TAUBIN G., GUÉZIEC A., HORN W., LAZARUS F.: Progressive forest split compression. In *Proc. ACM SIGGRAPH* (1998), pp. 123–132.

[THLR98] TAUBIN G., HORN W., LAZARUS F., ROSSIGNAC J.: Geometry coding and VRML. *Proceedings of the IEEE 96*, 6 (1998), 1228–1243.

[TR98] TAUBIN G., ROSSIGNAC J.: Geometric compression through topological surgery. *ACM Trans. on Graphics 17*, 2 (1998), 84–115.