DOI: 10.1111/j.1467-8659.2010.01737.x

Eurographics Symposium on Rendering 2010
Jason Lawrence and Marc Stamminger
(Guest Editors)

Volume 29 (2010), Number 4

On Floating-Point Normal Vectors

Quirin Meyer[1], Jochen Süßmuth[1], Gerd Sußner[2], Marc Stamminger[1] and Günther Greiner[1]

[1]Computer Graphics Group Erlangen, [2]Realtime Technology AG

Abstract

In this paper we analyze normal vector representations. We derive the error of the most widely used representation, namely 3D floating-point normal vectors. Based on this analysis, we show that, in theory, the discretization error inherent to single precision floating-point normals can be achieved by $2^{50.2}$ uniformly distributed normals, addressable by 51 bits. We review common sphere parameterizations and show that octahedron normal vectors perform best: they are fast and stable to compute, have a controllable error, and require only 1 bit more than the theoretical optimal discretization with the same error.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.6]: Methodology and Techniques—Graphics data structures and data types;

1. Introduction

In Computer Graphics, *3D normal vectors* are omnipresent; in particular they are used for shading and lighting. However, storing a unit normal as three floating-point numbers exhibits a high redundancy for two reasons. First, all unit normals form the unit sphere, which is a 2-manifold, and thus two parameters theoretically suffice for uniquely identifying each unit normal. Second, the dynamic range of floating-point numbers results in a non-uniform distribution of normals across the sphere.

Our work is motivated by high-quality rendering of automotive CAD data. In this setting, models easily contain millions of vertices with associated normals. The highly tessellated models are very prone to artifacts due to normal discretization (Fig. 3). It is therefore desirable to store normals in a more compact form while causing as little precision loss as possible.

A common approach is to distribute N normals across the sphere as uniformly as possible using an optimization process, and to store the resulting normals in a *look-up table (LUT)*. A normal is represented by the index of the closest normal in the LUT, which has to be searched. This is only feasible for a rather small number of normals, e.g. $N = 2^{16}$, and even with simple Blinn-Phong lighting, strong visual artifacts may be observed (Fig. 3c).

We address the problem differently. Ultimately, a normal needs to be transformed into a component-wise representation. Therefore, we first examine the angular precision of this representation. Based on this analysis, we suggest to use *octahedron-normal vectors (ONVs)*, which have a controllable error, thus possibly being as precise as the component-wise floating-point representation. ONVs are fast and simple to compute and the component-wise representation can be reconstituted stably. In brief, our main results are as follows:

- A uniform sampling of a sphere with the accuracy of single precision floating-point normals requires 51 bits.
- We analyze various common sphere discretizations, and
- show that ONVs are the best choice. They require only one bit more than the theoretical optimum discretization and can be computed very efficiently.

2. Previous Work

Although normals play a central role in Computer Graphics, it has not yet been theoretically assessed how many normals are required for producing artifact-free renderings. Deering [Dee95] deduces from empirical tests that 2^{17} normals are sufficient for rendering and proposes to encode normal vectors by dividing the sphere into 48 spherical triangles. A normal is then represented using 18 bits, where 6 bits are used to address the spherical triangle containing the normal

and 2×6 bits to encode the spherical coordinates within one such triangle. Oliveira and Buxton [OB06] compute a set of representative normals by recursively subdividing a Platonic solid. Following Deerings argumentation, they store the generated normals in a 16-bit LUT. Griffith et al. [GKP07] generalize the aforementioned method for arbitrary convex base polyhedrons and suggest a method based on a barycentric parameterization of the base polyhedrons' faces. Górski and coworkers [GHB*05] propose to unfold a rhombic dodecahedron into the plane using the HEALPix projection for distributing $12N^2$ points as uniformly as possible on a sphere, such that the points lie on $4N-1$ circles of latitude.

3. Analysis of Floating-Point Normal Vectors

In Computer Graphics, the main application for normals is lighting. Lambertian reflectance is contained in almost any lighting model. Using Lambertian reflectance, the error Δ_C of the computed pixel intensity when using a quantized normal $\vec{q}$ instead of the exact normal $\vec{n}$ is

$$\Delta_C = |\langle \vec{n}, \vec{l} \rangle \cdot c_d - \langle \vec{q}, \vec{l} \rangle \cdot c_d|.$$

As the value of the diffuse color c_d is at most 1 and $\|\vec{l}\| = 1$, we bound the error using the Cauchy-Schwarz inequality to

$$\Delta_C \leq \|\vec{n} - \vec{q}\|.$$

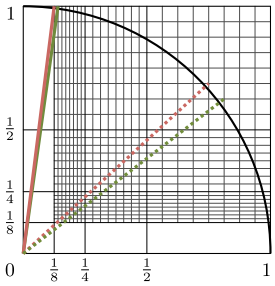
If the display color depth is d, we require $\Delta_C < 2^{-d}$ so that the intensity error is negligible. Using Δ_C as the radius of a disk, we can estimate how many such disks, assuming a perfect packing, can be distributed over the unit sphere, i.e.:

$$N = \frac{4\pi}{2^{-2d}\pi} = 2^{2(d+1)}. \tag{1}$$

Thus, for $d = 8$ bits per color channel, $N = 2^{18}$ equally spaced normals are sufficient for Lambertian reflectance. However, when using lighting models with higher variation, artifacts arise as shown in Fig. 3. As a remedy, one could increase the number of normals but then, by altering BRDF parameters, we can easily construct an example that shows quantization errors, again.

Thus, we can only answer the question of how many normals are required, when considering the specific problem for which they are used. If such estimations cannot be derived, one ends up storing the components explicitly as three floating-point numbers. Since this is also the format all other representations of normal vectors have to be converted to at some point, the component representation can be considered the gold standard. Any other representation — in order to be lossless — should be as accurate.

In the following, we derive the angular accuracy of the component-wise representation of normals using floating-point numbers. A normalized IEEE-754 floating-point number [IEE08] consists of a sign bit s, a signed integer e for the exponent, and a bit string of length m with elements a_i for the mantissa. It maps to the real value $(-1)^s \cdot 2^e \cdot \left(1 + \sum_{i=0}^{m-1} 2^{-i-1} \cdot a_i\right)$. This representation results in varying sampling densities, i.e. in the interval $[2^i, 2^{i+1})$ the sampling rate is twice as high as in the interval $[2^{i+1}, 2^{i+2})$. As the components of a normal are not limited to a single interval, *floating-point normal vectors (FPNVs)* are highly non-uniformly distributed. However, the higher resolution contained in some areas is lost when normals are transformed into areas of lower resolution: In the inset figure, the solid red and green vectors are located in a region of high floating-point resolution. By rotation with the same angle they get mapped onto the dashed vectors into a region with lower precision, coarsening the angular precision. This is due to the *finite storage* of floating-point numbers and not due to the *finite precision* of floating-point operations. As the extra resolution is destroyed by an operation as simple as a camera transformation, even if it were computed at infinite precision, it is not exploitable. Thus, the error measurement for the angular precision of FPNVs is *the largest angular distance* between a real-valued direction vector and its nearest floating-point vector in the lattice of floating-point numbers.



Normal vectors are computed from direction vectors by normalizing them. As soon as the direction vector is computed, discretization errors occur. Thus, we analyze the angular error of floating-point direction vectors. When converting a real-valued vector into a *floating-point vector*, it is rounded to the closest vertex of the containing *cell* of the 3D floating-point lattice. For floating-point vectors, the maximum error caused by quantization is the distance from the center of the cell to one of its vertices. However, for *floating-point direction vectors*, we are interested in the *angular quantization error*

$$\Delta_S = \arccos \frac{\langle \vec{v}, \vec{c} \rangle}{\|\vec{v}\| \, \|\vec{c}\|}, \tag{2}$$

where $\vec{c}$ is the center and $\vec{v}$ is a vertex of the cell. Let a *block* be the set of cells which have the same resolution. We will show that the angular quantization error has its maximum in a "lower-left cell" of a diagonal block and $\vec{v}$ is one of the vertices $[2^i + \varepsilon, 2^i, 2^i]^T$, $[2^i, 2^i + \varepsilon, 2^i]^T$, or $[2^i, 2^i, 2^i + \varepsilon]^T$, where ε is the size of the cell. We consider the 2D case and without loss of generality we can assume that the cell is located in a block above the diagonal. We move this cell to a "lower-left cell" of a diagonal block in two steps (Fig. 1):

1) We translate the cell within the block of constant resolution vertically to the lower bound of the block.
2) Then we move it horizontally to the right, until we reach the diagonal, hereby scaling the width linearly from the initial width to the width of the final cell.

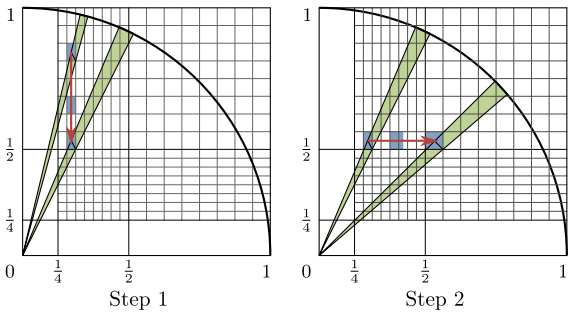


Figure 1: The discretization error of a point in the center of a cell increases when moving the cell towards the diagonal.

Both motions increase the angle between the cell center $\vec{c}(t)$ and a cell vertex $\vec{v}(t)$. This can be seen by analyzing the rational function $\Delta_S(t) = \frac{\langle \vec{v}(t), \vec{c}(t) \rangle^2}{\|\vec{v}(t)\|^2 \|\vec{c}(t)\|^2}$. In fact, a detailed analysis reveals that this function has a negative derivative. Thus, it is decreasing, and since $\sqrt{\cdot}$ is increasing and arccos is decreasing we conclude that the angle between $\vec{c}(t)$ and $\vec{v}(t)$, which is given by $\arccos\sqrt{\Delta_S(t)}$, is increasing. The argumentation directly carries over to 3D.

This observation shows that for the standard floating-point representation the maximum discretization error $\Delta_S^{\max}$ for the angle is given by (2) with $\vec{c} = (1/2+\varepsilon/2, 1/2+\varepsilon/2, 1/2+\varepsilon/2)$ and $\vec{v} = (1/2+\varepsilon, 1/2, 1/2)$:

$$\Delta_S^{\max} = \arccos\sqrt{1 - \frac{8\varepsilon^2}{9+12\varepsilon+12\varepsilon^2}} = \frac{2}{3}\sqrt{2}\varepsilon + O(\varepsilon^3).$$

Thus, for FPNVs with m-bit mantissae we have

$$\Delta_S^{\max} = \frac{\sqrt{2}}{3}2^{-m} + O(2^{-3m}). \tag{3}$$

Using this formula we can give a lower bound for the minimum number of normals that are necessary to sample the unit sphere with the same resolution as the standard FPNV representation. By dividing the area of the unit sphere by the area of a disk with radius $\Delta_S^{\max}$, we obtain $\frac{4\pi}{(\Delta_S^{\max})^2\pi} = \frac{9}{2}\varepsilon^2 = 18\cdot 2^{2m}$. Hence, one needs

$$\left\lceil \log_2(18\cdot 2^{2m}) \right\rceil = 2m+5 \tag{4}$$

bits to encode normals having the same accuracy as three floating-point numbers with mantissa length of m. Consequently, $\lceil 50.2 \rceil = 51$ bits are needed for a normal discretization to be as accurate as single precision FPNVs.

4. Common Sphere Parameterizations

When trying to satisfy the error of single precision FPNVs with LUT methods, a table size of about 24 Petabytes is required. Subdivision methods would alleviate this problem, however, at high computational costs, as more than 20 subdivision steps are needed to achieve the same precision.

Another possible approach is to use sphere parameterizations: with a parameterization we compute the three components of a normal from two parameters. For example, a 3D point on a unit sphere may be retrieved from two angles using *spherical coordinates (SC)*. Similar to the considerations of Sec. 3, it can be shown that the maximum error of this parameterization is

$$\Delta_{SC}^{\max} = \arccos\left(\cos^2\left(\pi\cdot\frac{\varepsilon}{2}\right)\right) = \frac{\sqrt{2}}{2}\pi\varepsilon + O(\varepsilon^3). \tag{5}$$

However, spherical coordinates require trigonometric functions which are expensive to compute. Another straightforward approach, typically used for tangent space normal maps [ATI05], is *parallel projection (PP)*: Given x and y, the z component of a point $[x,y,z]^T$ on the unit sphere can be computed by $|z| = \sqrt{1-x^2-y^2}$. An additional sign bit has to be stored to address both hemispheres. One disadvantage is, that not all values of $[x,y] \in [-1,1]^2$ map to a point on the sphere. More importantly, computing z becomes very instable for points close to the equator, as $\lim_{x^2+y^2\to 1}\|\nabla s(x,y)\| = \infty$, where the maximum error is:

$$\Delta_{PP}^{\max} = \arccos\left(1 - \frac{\sqrt{2}}{2}\varepsilon\right) = \sqrt[4]{2}\sqrt{\varepsilon} + O(\varepsilon^2). \tag{6}$$

To alleviate this effect, the sphere can be divided into sextants by reconstructing the largest component of $[x,y,z]$ from the two smaller ones. We refer to this approach as *sextant parallel projection (SPP)*. This comes at additional branching cost during reconstruction and three extra bits for identifying the sextant. This results in a maximum error of

$$\begin{aligned}\Delta_{SPP}^{\max} &= \arccos\left(\frac{2}{3} - \frac{\sqrt{3}}{3}\varepsilon + \frac{1}{6}\sqrt{4+\sqrt{192}\varepsilon - 6\varepsilon^2}\right) \\ &= \frac{\sqrt{2}}{2}\sqrt{3}\varepsilon + O(\varepsilon^2).\end{aligned} \tag{7}$$

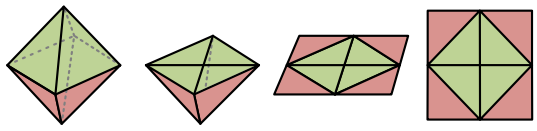
Another option is to use a *cube map (CM)*, i.e. projecting the normal onto the cube $[-1,1]^3$ and omitting the coordinate, whose absolute value equals 1. The maximum error for this parameterization is:

$$\Delta_{CM}^{\max} = \arccos\left(\frac{2}{\sqrt{2\varepsilon^2+4}}\right) = \frac{\sqrt{2}}{2}\varepsilon + O(\varepsilon^3). \tag{8}$$

Like the SPP parameterization, the disadvantage is that we need three bits to encode six sextants, hence two states remain unused.

5. Octahedron Normal Vectors

The aforementioned parameterizations are either expensive to compute, numerically instable, or do not fully utilize the given bit budget. We propose to use *octahedron-normal vectors (ONVs)* instead [PH03, ED08]. They can be obtained



(a) (b) (c) (d)

Figure 2: Unwrapping an octahedron. The apex of the green pyramid (a) is projected to the bottom plane of the pyramid (b). The faces of the red pyramid are folded up onto the same plane (c), yielding a 2D unwrapping of the octahedron (d).

by projecting an FPNV onto the octahedron by normalizing it using the 1-norm. Then the octahedron is unwrapped to a square according to Fig. 2 and the parameters $[u,v]$ in the plane are stored in fixed-point format. Given the parameters $[u,v]$ of an encoded normal, a normal $\vec{n}' = [x,y,z]^T$ on the octahedron can be reconstituted using the following equations:

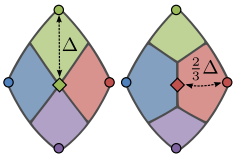
$$\begin{aligned} z &= 1-|u|-|v| \\ [x,y]^T &= \begin{cases} [u,v]^T & \text{if } z \geq 0 \\ [\sigma(v)-v, \sigma(u)-u]^T & \text{if } z < 0 \end{cases} \end{aligned} \tag{9}$$

where $\sigma(t) = 1$ for $t \geq 0$ and $\sigma(t) = -1$ otherwise. The original normal $\vec{n}$ is obtained by normalizing $\vec{n}'$ using the 2-norm. This computation is fast, as it involves only a few basic operations and it is numerically stable, with only fixed-point computations being used. Also note that in contrast to parallel-projection parameterizations, every element of the domain $[-1,1]^2$ maps to a valid normal.

The maximum angular error $\Delta^{\max}_{ONV_{domain}}$ for ONVs can be derived as the angle between $[1/3, 1/3, 1/3]^T$ and $[1/3 + \varepsilon/2, 1/3 + \varepsilon/2, 1/3 - \varepsilon]^T$, where ε is the sample spacing.

$$\Delta^{\max}_{ONV_{domain}} = \arccos\sqrt{\frac{2}{2+9\varepsilon^2}} = \frac{\sqrt{2}}{2}3\varepsilon + O(\varepsilon^3) \tag{10}$$

We can reduce this error even further: the Voronoi cells of the points in the parameter domain, when projected onto the sphere (inset figure left), do not correspond to the Voronoi cells of the ONVs projected onto the sphere (inset figure right). Hence, rounding in the parameter domain does not generally guarantee that a normal is quantized to the closest ONVs with respect to the angular distance. Thus, instead of rounding in the parameter domain, we round in the range. First we compute the parameter domain cell in which the original normal maps to by inverting (9). Then we choose from its four corner vertices that one, whose normal has the smallest angular distance. One can show that this reduces the error to $2/3 \cdot \Delta^{\max}_{ONV_{domain}}$, i.e. the theoretical error $\Delta^{\max}_{ONV_{range}}$ for ONVs is



$$\Delta^{\max}_{ONV_{range}} = \sqrt{2}\varepsilon + O(\varepsilon^3). \tag{11}$$

From Eqs. (3) and (11) follows that we need 52 bits for ONVs to be as precise FPNVs at 96 bits. Detailed derivations of the error bounds in Sec. 3 – 5 are presented in [MSS*10].

6. Results and Discussion

In Sec. 4 we derived theoretical bounds for the maximum error of various common sphere parameterizations depending on the domain's sample spacing ε. The maximum error for naïve parallel projection is $O(\sqrt{\varepsilon})$ and thus PP is much worse than other parameterizations with an error of $O(\varepsilon)$. Yet, for normal maps, this parameterization makes sense, since in a normal map, normals predominantly point into z direction. The formulae (5, 7, 8, 10, 11) reveal that the errors (up to third order) for SC, SPP, CM, and both ONVs are all linear in ε, with CM having the smallest factor. This suggests that CM are the best choice. However, CM have to index six faces by a three bit mask. Thus, given a bit budget of b bits, only $b-3$ bits may be used effectively to sample the domain, which results in a larger sample distance ε.

In Tab. 1, we give the sampling distances $\varepsilon(b)$ and error bounds $\Delta^{\max}(b)$ for a given bit budget b for FPNVs and the aforementioned parameterizations. Note that CM perform better than ONVs when quantized in the domain. However, when quantizing in the range, ONVs outperform the others. Quantizing in the range only reduces the *average* error of CM, the *maximum* error $\Delta^{\max}_{CM}$ remains unchanged. Thus, from a theoretical point of view, ONVs are the best choice.

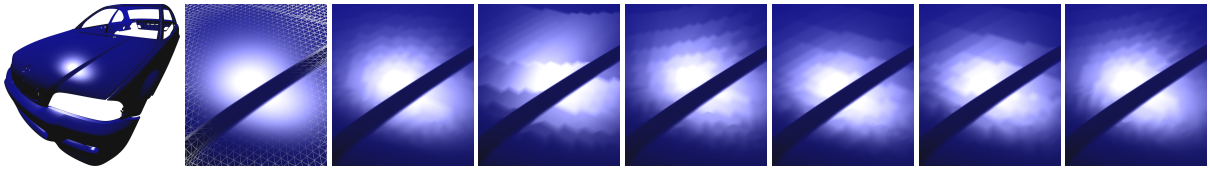
In the third row of Tab. 1 we compare the maximum quantization error for a fixed bit budget of 48 bits. For comparison, Griffith's "Spherical Covering 2" [GKP07] uses a complicated subdivision method requiring 21 subdivisions or a Petabyte large LUT to obtain an error of $1.38 \cdot 10^{-7}$ when using 48 bits. Note that ONVs come close to this number at only a fraction of that cost. Moreover, the error of FPNVs using 96 bits ($5.62 \cdot 10^{-8}$) is only a factor of 3 better than $\text{ONV}_{\text{range}}$ ($1.69 \cdot 10^{-7}$) using only half the storage.

Fig. 3 shows a qualitative comparison of the tested parameterizations at a bit budget of 16 bit. We use a finely tessellated model of a car body. The tessellation is depicted in Figure 3b. Figures 3c and 3h show that the visual quality of ONVs is indistinguishable from the one achieved by Griffith.

Looking at the number of bits required to store normals with a given maximum angular error, ONVs are very close to the theoretical optimum. Comparing the error Δ_{OPT} of a perfectly distributed sphere packing of $N = 2^{b_{opt}}$ normals with the error of ONVs

$$\Delta_{OPT} = 2^{1-\frac{b_{opt}}{2}} \quad = \quad 2\sqrt{2}\cdot 2^{-\frac{b_{ONV}}{2}} = \Delta_{ONV_{range}}$$

and solving for b_{ONV} reveals that we always need only one bit more than the theoretical optimum.



(a) Total (b) Ground truth (c) Griffith (d) PP (e) SPP (f) SC (g) CM (h) ONV

Figure 3: Effect of various sphere parameterizations on a finely tessellated model of car body using Blinn-Phong per-pixel lighting. The entire model is shown in (a) and a close-in a up of the highlight in (b) using single precision FPNVs. The remaining images use normals quantized to 16 bits, under different parameterizations.

Method	FPNV	PP	SPP	SC	CM	$\text{ONV}_{\text{domain}}$	$\text{ONV}_{\text{range}}$
$\varepsilon(b)$	$2\cdot 2^{-\frac{b}{3}}$	$2\cdot 2^{-\frac{b-1}{2}}$	$\sqrt{2}\cdot 2^{-\frac{b-3}{2}}$	$1\cdot 2^{-\frac{b-1}{2}}$	$2\cdot 2^{-\frac{b-3}{2}}$	$2\cdot 2^{-\frac{b}{2}}$	$2\cdot 2^{-\frac{b}{2}}$
$\Delta^{\max}(b)$	$\frac{\sqrt{32}}{3}\cdot 2^{\frac{b}{6}}\cdot 2^{-\frac{b}{2}}$	$2^{1+\frac{b}{4}}\cdot 2^{-\frac{b}{2}}$	$2\sqrt{6}\cdot 2^{-\frac{b}{2}}$	$2\pi\cdot 2^{-\frac{b}{2}}$	$4\cdot 2^{-\frac{b}{2}}$	$3\sqrt{2}\cdot 2^{-\frac{b}{2}}$	$2\sqrt{2}\cdot 2^{-\frac{b}{2}}$
$\Delta^{\max}(48)$	$4.60\cdot 10^{-4}$	$4.88\cdot 10^{-4}$	$2.92\cdot 10^{-7}$	$3.74\cdot 10^{-7}$	$2.38\cdot 10^{-7}$	$2.53\cdot 10^{-7}$	$1.69\cdot 10^{-7}$

Table 1: The maximum error for various parameterizations for a given bit budget b. The resulting maximum sample spacing ε in the parameter domains and the maximum angular error $\Delta^{\max}$ are shown in the first and second row, respectively. The maximum error for 48 bits per normal is listed in third row.

ONVs are perfectly suited for per-vertex normal compression on graphics hardware, as they do not require complicated branching or expensive arithmetic functions. On our hardware, only 17 assembly instructions are required for decompression. In engineering applications, large, finely tessellated models are very common and memory space may run short. Yet, accurate normals are required for lighting and reflectance computations. Using ONVs at 48 bit offers a simple approach to reduce the space required for normal vectors by 50%, while still having random access on the normals at very little costs and precision loss. Note that such settings are memory bandwidth limited rather than compute-bound. In our applications, running on current hardware, we can even observe a speed-up of approximately 5%, due to the reduced memory transfers. Moreover, integration into existing shader-programs is trivial, as only the access to the normals has to be encapsulated by a function.

7. Conclusion and Future Work

In this paper we presented the angular error inherent to FPNVs. Based on these observations, we analyzed the most popular sphere parameterizations. ONVs turned out to be the best choice from a practical and theoretical point of view: they are simple to implement and require only a few instructions. Given a fixed bit budget, their error is at most twice the error of an optimal (theoretical) bin packing of a sphere. Others methods, e.g. LUT and subdivision methods, exhibit even less redundancy, however, at very high costs for either reconstituting or storing the normals. This effort could save at most one bit per vertex normal over our ONVs. At 3 times the error inherent to FPNVs, ONVs save 50% memory.

As our method is orthogonal to generic compression methods, we plan to explore the combination of ONVs with standard compression techniques. We will also investigate the suitability and quality of ONVs for tangent-space normal maps, especially using texture compression [ATI05].

References

[ATI05] ATI: *Radeon X800: 3Dc White Paper*. Tech. rep., 2005.

[Dee95] DEERING M. F.: Geometry Compression. In *Proc. of SIGGRAPH '95* (1995), pp. 13–20.

[ED08] ENGELHARDT T., DACHSBACHER C.: Octahedron Environment Maps. In *Proc. of VMV '08* (2008), pp. 383–388.

[GHB*05] GÓRSKI K. M., HIVON E., BANDAY A. J., WANDELT B. D., HANSEN F. K., REINECKE M., , BARTELMANN M.: HEALPix: A Framework for High-Resolution Discretization and Fast Analysis of Data Distributed on the Sphere. *The Astrophysical Journal 622*, 2 (2005), 759–771.

[GKP07] GRIFFITH E. J., KOUTEK M., POST F. H.: Fast Normal Vector Compression with Bounded Error. In *Proc. of SGP '07* (2007), pp. 263–272.

[IEE08] IEEE: IEEE Standard for Floating-Point Arithmetic. *IEEE Std 754-2008* (2008), 1–58.

[MSS*10] MEYER Q., SÜSSMUTH J., SUSSNER G., STAMMINGER M., GREINER G.: *Quantization Errors of Popular Normal-Vector Representations*. Tech. rep., Inf. 9, Univ. of Erlangen, 2010. In Preparation, Available at www9.cs.fau.de.

[OB06] OLIVEIRA J. F., BUXTON B. F.: PNORMS: Platonic Derived Normals for Error Bound Compression. In *Proc. of VRST '06* (2006), pp. 324–333.

[PH03] PRAUN E., HOPPE H.: Spherical Parametrization and Remeshing. In *Proc. of SIGGRAPH '03* (2003), pp. 340–349.