Design of an On–Chip Reflectance Map

Jeroen Terwisscha van Scheltinga, Jaap Smit, and Marco Bosma

University of Twente
Department of Electrical Engineering EF9250
PO Box 217, 7500 AE Enschede, The Netherlands
e–mail: jaap@nt.el.utwente.nl

Abstract

A reflectance map design is described which uses a minimal amount of memory for the table, in order to be applicable as an on–chip shader. The shader is designed for use with the volumetric super resolution hardware, which performs shading at supersampled locations. However, the design may be used as well to support surface visualization applications. Despite the small table size, the image quality obtained is excellent, even on smooth surfaces.

1 Introduction

Shading of surfaces is performed using a light model. The light model of Phong [1] is simple and therefore often used. It uses the surface normal, the direction of the light source(s), the direction of the viewer and characteristics of the surface to calculate the intensity at a sample position.

The surface normal must be normalized to unit length. This involves operations as square and add, square root and three divisions [2], [3]. Thereafter, the different components of the Phong light model need to be calculated and added together. Especially the specular component is expensive to compute. If more than one light source is used, calculating the reflection components has to be repeated for every light source and added to the result. Therefore, calculating the light intensity at a sample position is a costly operation, especially when the specular component of the Phong light model or multiple light sources are used.

Because of these expensive operations, often a look–up table is used, containing precalculated intensities for a set of predefined surface normal directions. It can easily be adjusted to handle different parameters, light models and even multiple light sources without affecting the rendering time. Parallel view and light vectors are assumed however, to make the reflection independent of the place. Unless the parameters and the light and view directions do not change, the look–up table has to be recomputed every frame, which is not a problem since the host computer can easily compute a small table in real time.

To find the color intensity at a sample position, some sort of encoding of the surface normal is needed to find the index in the reflectance map. In volume rendering, the surface normal can be found by calculating the normalized gradient vector of the grey–level data [4].

The simplest encoding method uses a 3D look–up table [5]. The most significant bits from each gradient component are directly used to index the table. At least five bits per component are needed to give acceptable results, leading to a minimal table size of 32 KByte.

Sophisticated encoding schemes map the surface normal onto one or more 2D planes [6], [7].

If a unit length surface normal is used, only the sign bit of one of the components is needed together with the other two components [6]. The surface normal is thus mapped onto two planes. Six bits per component then give a table size of 8 KByte.

All surface normal components can be normalized on one of the components [7]. The other two (normalized) components are converted into angles by using two atan tables of 64 KByte. These angles are used to index the reflectance map. The reflectance map has a size of $180^2 \approx 32$ KByte entries.

The reflection vector can be used as an index into a cube environment map [8]. The reflection vector is computed from non–normalized surface normal and eye vectors, and therefore is itself also unnormalized. Computing the reflection vector requires twelve multiplications. The environment map is stored in a texture map and creates perspective–correct mirror–like reflections or specular highlights.

Strong highlights caused by specular shading can be a problem when using a small look–up table. Because only a few table entries are available for the fast changing intensity information near the highlight, intensity contouring will occur if the table size is too small.

2 A reflectance map encoded on a cube

All surface normals are located on the surface of a sphere with radius one. A single entry in the reflectance map contains the intensity of a small surface area of this sphere. Ideally, the encoding would map each entry to a surface area of identical size, to ensure an uniform partitioning of all possible surface normal directions.

2.1 Overview

As a sphere is hard to map in a table, another shape is needed. As a simple approximation of a sphere a cube can be used. A cube consists of six planes, which can be mapped in a table with ease.

In general, the gradient vector can have any length, even zero, in which case it cannot be normalized. To map the gradient vector on a cube, each component is normalized on the maximum component. After normalization, the maximum component is always equal to one, hence it can be represented with its sign bit only. For each of the three maximum components possible, a separate map is needed. The map corresponding with the maximum component is used, while the two remaining scaled vector components are the indices used to address this map.

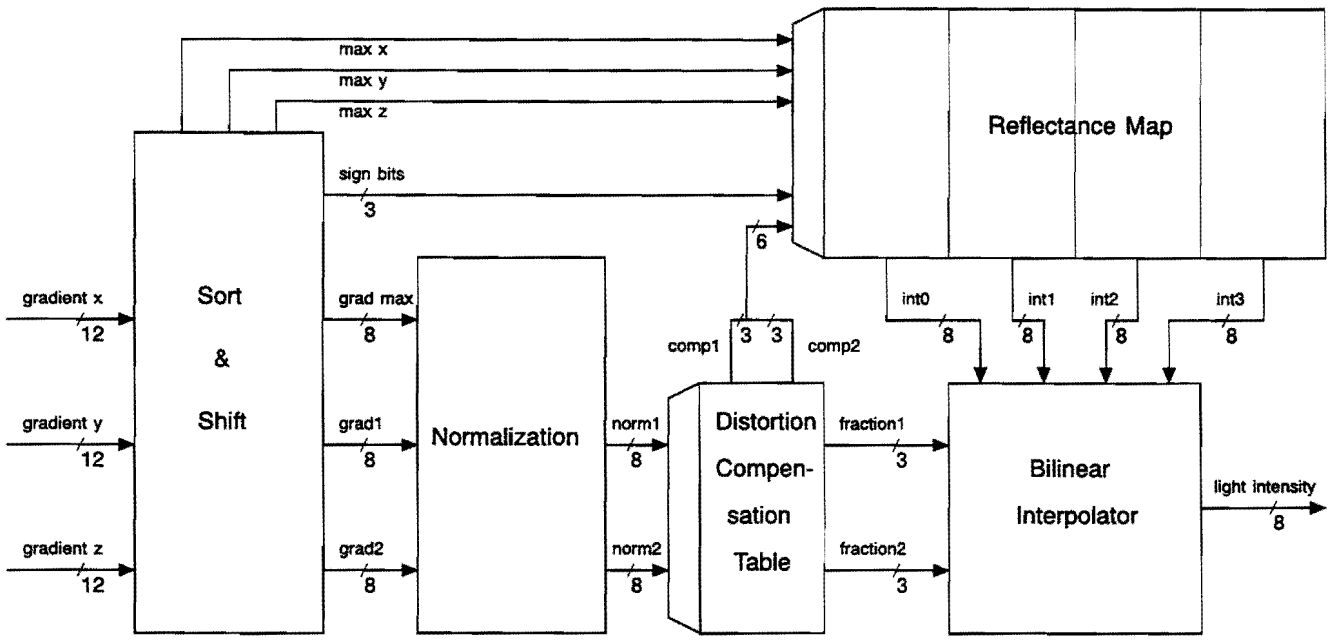

Figure 1. *Overview reflectance map design.*

In Figure 1, an overview of the table look–up process is given. The three gradient components are sorted to find the maximum. If leading zero bits are present in all components they are shifted out to retain the highest possible precision. The two smallest components are normalized on the maximum component. These normalized values give the indices for the reflectance map by using a distortion compensation table. After fetching four entries from the reflectance map, an interpolator provides the light intensity at the sample position. The light intensity can be used directly as a grey value, or it is multiplied with the red, green and blue values of the sample point to give the sample color.

Figure 2. *Example of the image quality obtained using a 1.5 KByte reflectance map.*

An example image generated with a reflectance map of 1.5 KByte is given in Figure 2. It is practically indistinguishable from a version rendered using floating point arithmetic. Test images of a sphere are used to illustrate the image quality, as a sphere contains slowly varying surface normal directions and clearly shows the individual entries of the reflectance map if the map size is too small.

The parts of the design are now discussed in more detail.

2.2 Sort and shift

The gradient vector is represented in X, Y and Z components in 12–bits precision. The three sign bits are taken apart, as they are directly used as index bits in the shading table. Only the length of each component is used during normalization.

The three components are sorted by computing X–Y, X–Z and Y–Z. A simple logical operation on the sign bits of the three results gives the maximum component. The number of bits is reduced to eight to simplify the computations which follow. Often the gradient vector is small, so leading zero bits are shifted out first, to maintain the precision of the results.

The maximum component is multiplexed to the max output, the other two components are not sorted but are multiplexed in a specific order instead.

2.3 Normalization

To normalize the gradient components on the maximum component, they have to be divided by the maximum component. A direct implementation of the above requires two dividers. Another possibility is to compute the reciprocal of the maximum first, followed by two multiplications. This requires one divider and two multipliers. However, instead of the divider, a small ROM table can be used to provide the reciprocal of the maximum.

Normally, the scaled components contain only fraction bits. But if a component is equal to the maximum, the scaled component will be one, so an extra bit is needed. Instead of this extra bit, all fraction bits could be set to get a similar result.

2.4 Distortion compensation

Although the encoding on a cube gives a good partitioning of all surface normal directions, some distortion due to the mapping of the sphere on a cube will result. The size of the spherical surface normal area mapped by a single table entry decreases as the normal direction is further away from one of the axes. This is illustrated in Figure 3. The distance d2 is smaller than the distance d1, while the distance dx is the same for all positions in the table. A test image of a sphere illustrating this distortion effect is shown in Figure 4, using only two bits per component to emphasize the effect.

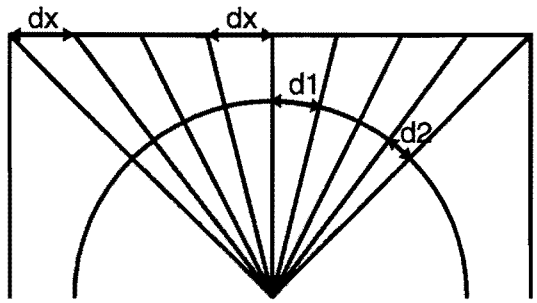

Figure 3. *Distortion effect caused by mapping a cube on a sphere.*

The rectangular area of a single entry is distorted, and is the result of mapping the cube onto a spherical surface. As a result, some surface normal directions have a better angular resolution than others. A distortion compensation table coded using an atan function can partially compensate for this effect. This table only has to handle cases from 0° to 45° and can be placed in ROM. As it must be able to handle two reads in parallel, two tables or two read ports are needed. In Figure 5, distortion compensation is applied using the same settings as in Figure 4. The effect of the compensation is clearly visible near the highlight. The areas near the highlight are smaller than the corresponding areas of the image without compensation.

2.5 Table look–up

Table look–up is performed by using the sign bits of the three normal components and the two, distortion compensated, values of the normalized components. The maximum component is used to select the corresponding map. The map contains 8–bit intensity values.

The angle mapped by a single entry, using three bits per component from the distortion compensation table, is 45° / 7 = 6.5°. Every extra bit per component bisects this angle.

2.6 Interpolator

Finally, an interpolation stage computes the intensity at the sample position. Because the intensity values of neighboring entries in the reflectance map change slowly, an interpolator can give a significant improvement in image quality. Therefore, the distortion compensation table provides three extra (lower order) bits per component to use in the interpolator. The four closest entries to the sample position are read in parallel from the reflectance map. By dividing the reflectance map in odd and even parts for both components, this parallel reading is always possible.

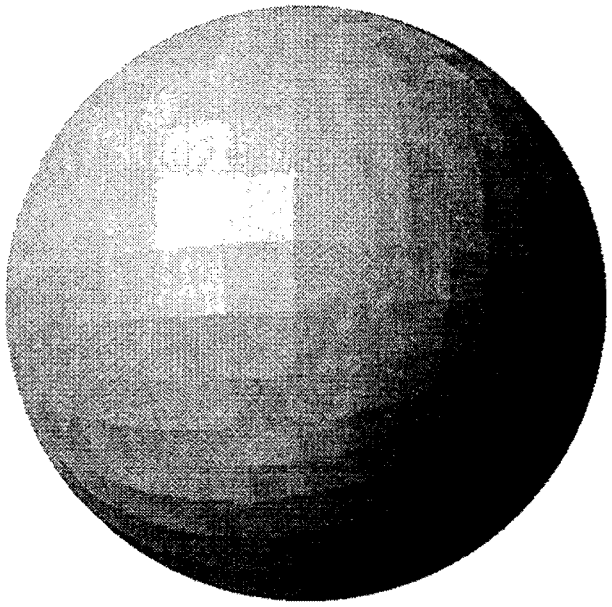

Figure 4. *Sphere without distortion compensation.*

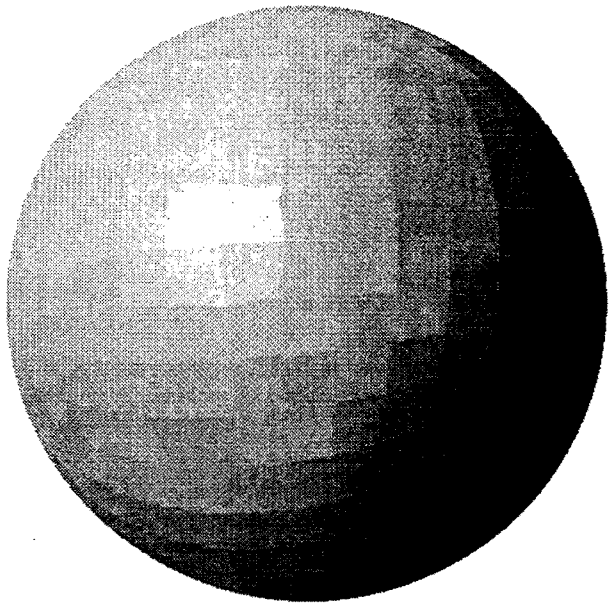

Figure 5. *Sphere with distortion compensation.*

The four intensity values are bi–linearly interpolated using the three extra bits. Figure 6 shows the result using three bits per component from the distortion compensation table. The same image is also generated by using floating point arithmetic to perform the shading, and compared with the reflectance map image. The resulting 50 times enlarged error image is shown in Figure 7. It is clear that the largest errors are made near the highlight.

Near a highlight, the intensity values change fast. The interpolator cannot create correct values if the highlight is too strong. To prevent this, more entries in the reflectance map are needed or strong highlights must be avoided.

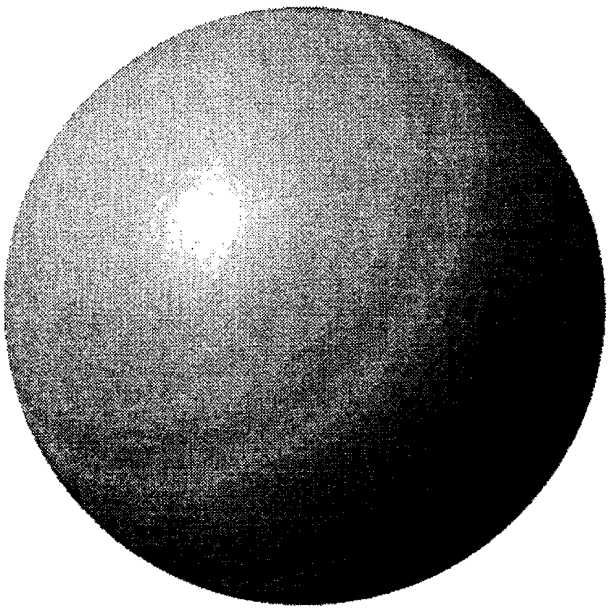

Figure 6. *Sphere with distortion compensation and interpolation.*

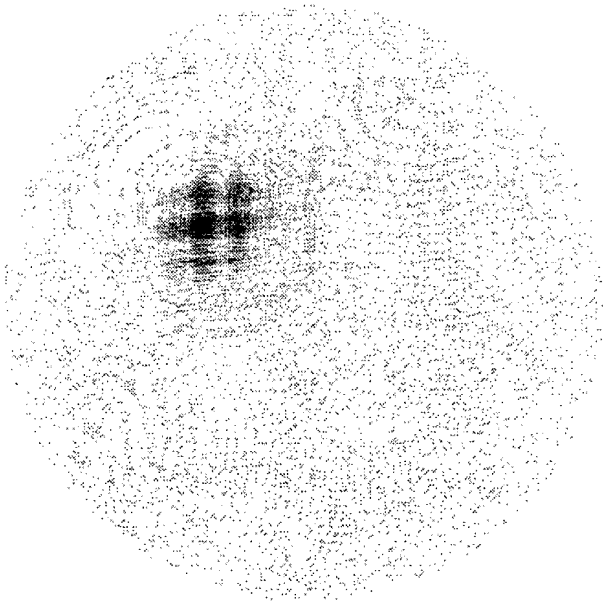

Figure 7. *A 50 times enlarged error image of the image in Figure 6.*

The entries in the reflectance map are organized differently in behalf of the interpolator. Figure 8 shows one side of the cube (without distortion compensation) using two bits per component, with the circles representing entries of the map. Each square represents a surface area with equal intensity. At the place of a circle the light intensity is calculated. Figure 8a depicts the situation without interpolation, showing uniformly partitioned entries. In Figure 8b, the situation is depicted with an interpolation using two bits per component. Only some squares have a corresponding entry and the squares are smaller because of the interpolation. The interpolation needs entries at the borders of the plane, which causes some entries to almost overlap others. This effectively reduces the range of the components by one.

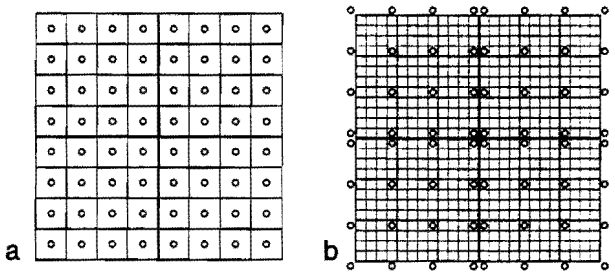

Figure 8. *Partitioning of map entries over one side of the cube.* **a:** *without interpolation.* **b:** *with interpolation.*

For the interpolation the closest lower (base) entry and the three neighboring higher entries are used. Therefore, outer entries must never become a base entry. This is ensured by placing them outside the plane. The distortion compensation table can be used to compensate for the unusual range of values needed.

3 Implementation

The reflectance map design described, is simulated using the Vivid software developed at the University of Twente. Vivid is a test environment for volume rendering. It allows interactive change of table parameters, to be able to calculate and view image differences immediately.

A good image quality is obtained using only three bits per component from the distortion compensation table for indexing the reflectance map. This gives a 9–bit index (three sign bits plus two times three component bits), so the reflectance map size is 512 bytes. Three such tables are needed, resulting in a total size of 1.5 KByte. The distortion compensation table contains 6–bit entries, of which three bits are used as an index and three bits in the interpolator. The total amount of entries in the distortion compensation table is 256 and is placed in ROM.

A special case arises when the maximum component is zero. Then no surface normal exists as all components of the gradient vector are zero. This situation can be checked for during normalization, after which appropriate action will be taken.

In Figure 9 an engine block is rendered using this reflectance map design. There is practically no difference with the image generated with floating point arithmetic, as can be seen in Figure 10, which is a 50 times enlarged error image. Only at the highlights the errors become slightly visible.

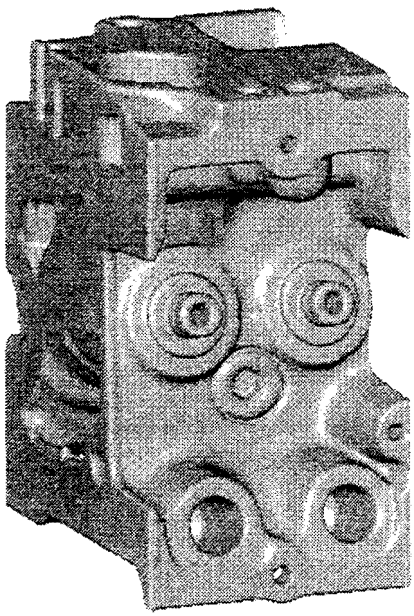

Figure 9. *Image of an engine block generated with a 1.5 Kbyte reflectance map.*

Figure 10. *A 50 times enlarged error image of the image in Figure 9.*

4 Conclusions

We presented the design of an on–chip reflectance map. Despite the small table size, it generates high quality images.

A cubic encoding of the surface normal ensures a good partitioning of all possible directions over the reflectance map entries. Therefore, the map size can be small and the intermediate calculations can be done using low precision without obtaining incorrect results.

The interpolator provides a higher angular resolution without increasing the size of the reflectance map, by taking advantage of the correlation between the entries. This prevents the appearance of intensity contouring, but is less suitable when specular highlights gets stronger.

Leaving out the distortion compensation table and the interpolator will result in lower quality images, but can simplify the design, while the difference is hardly visible if no large smooth surfaces are present.

References

[1] B.T. Phong, "Illumination for Computer Generated Pictures," *Communications of the ACM*, 18(6), June 1975, pp. 311 – 317.

[2] M. Margala, N.G. Durdle, S. Juskiw, V.J. Raso, and D.L. Hill, "A 33 MHz 16–bit gradient Calculator for Real–Time Volume Imaging," *Proc. of the 9th Eurographics Workshop on Graphics Hardware*, September 1994, ISSN 1017–4656, pp. 80–85.

[3] G. Knittel, "A VLSI–Design for fast Vector Normalization," *Proc. of the 8th Eurographics Hardware Workshop*, Barcelona, September 1993, pp. 1–14.

[4] M. Bosma, J. Smit, and J. Terwisscha van Scheltinga, "Super Resolution Volume Rendering Hardware," *Proc. of the 10th Eurographics Workshop on Graphics Hardware*, August 1995.

[5] M.C. Dogget and G.R. Hellestrand, "A Hardware Architecture for Video Rate Smooth Shading of Volume Data," *Proc. of the 9th Eurographics Workshop on Graphics Hardware*, September 1994, ISSN 1017–4656, pp. 95–102.

[6] T.S. Yoo, U. Neumann, H. Fuchs, S.M. Pizer, T. Cullip, J. Rhoades, and R. Whitaker, "Achieving Direct Volume Visualization with Interactive Semantic Region Selection," *Proc. Visualization 91*, IEEE CS Press, Los Alamitos, California, 1991, pp. 58–67.

[7] D. Jackèl and H. Rüsseler, "A Real Time Rendering System with Normal Vector Shading," *Proc. of the 9th Eurographics Workshop on Graphics Hardware*, September 1994, ISSN 1017–4656, pp. 48–57.

[8] D. Voorhies and J. Foran, "Reflection Vector Shading Hardware," *Proc. of SIGGRAPH '94*, July 1994, pp. 163–166.