Antialiasing of Environment-Maps

Andreas G. Schilling

ISSN 0946-3852
WSI-97-14

Wilhelm-Schickard-Institut für Informatik
Graphisch-Interaktive Systeme
Auf der Morgenstelle 10/C9
D-72076 Tübingen
Tel.: +49 7071 29-75462
Fax: +49 7071 29-5466

email: schilling@uni-tuebingen.de
URL: http://www.gris.uni-tuebingen.de/~andreas

Abstract

Environment-maps, like texture maps or any other maps consisting of discretely stored data have to be properly filtered, if they are being resampled in the process of rendering an image. For environment maps, this is especially important, as the sampling rate is subject to extreme changes due to the curvature of the reflecting surfaces. However, for the same reason, the antialiasing is especially difficult to perform, as the sampling rate has to be determined for each pixel. We introduce a method to perform this calculation and determine the parameters for anisotropic filtering of the environment map. The principles that are used to perform this antialiasing can be applied for antialiasing reflected textures in general e.g. in ray tracing.

Problem

Environment mapping is an excellent tool to enhance the quality of computer generated images. Although approximations are used (theoretically for each point of an object, a different environment map should be used), the pictures rendered with environment mapping often achieve a visual quality similar to ray-traced pictures. Unfortunately, antialiasing of environment maps is difficult. The sampling rate on the environment map can change by orders of magnitudes within the same object due to changes of the curvature of the object. Traditionally, the filtering, which often is performed with mip-maps, has to be adjusted manually to get the desired effect. If no anisotropic antialiasing is possible, artifacts are unavoidable. A simple example is the environment mapping on a cylinder, which has a curvature only in one direction. In this direction we get a very low sampling rate, which requires massive filtering. In the other direction, parallel to the axis of the cylinder, the sampling rate is higher and less filtering would be required.
If environment mapping is combined with bump mapping, even more realistic images are possible. On the other hand, the antialiasing is becoming more difficult, as a bumpy surface reflects incoming rays into a larger sector of space which corresponds to a larger area of the environment map.

Solution

The intersection of the reflection of the pixelwide viewing beam and the environment map will be called footprint in this paper (Fig. 1). The size and shape of this footprint depends on three factors:

- the angular range of viewing rays covered by one pixel (Fig. 2)
- the curvature of the reflecting surface (Fig. 3), and
- the scattering of the viewing rays caused by the bump map on the reflecting surface (Fig. 4).

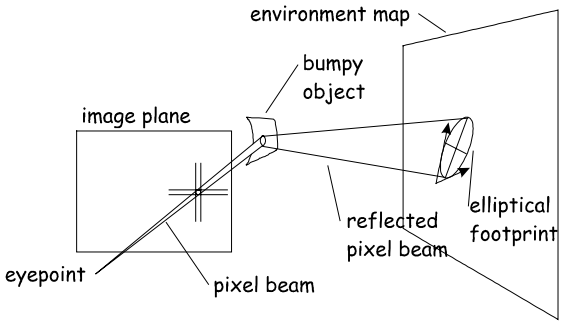

Fig. 1: *Environment mapping*

We will calculate and combine these contributions using linear approximations. The result will be a matrix, that describes the footprint with an ellipse. The main axes of this ellipse are then used to perform anisotropic filtering of the environment map, using for example the footprint assembly method [3].
The contributions of the first two factors can be combined, as they are related in a fixed way. One of the contributions can compensate the other one or their results can add up. An example for the compensation would be a parabolic mirror, viewed from its focal point: the change in the direction of the viewing ray from pixel to pixel is compensated by the change of the normal direction, so that the reflected rays are parallel. We account for that by calculating the combined contribution of the first two factors. For the third factor, however, we have to assume that it is independent from the first two ones, as we know nothing about the location of the different normal directions within the area of the bump map covered by the pixel.

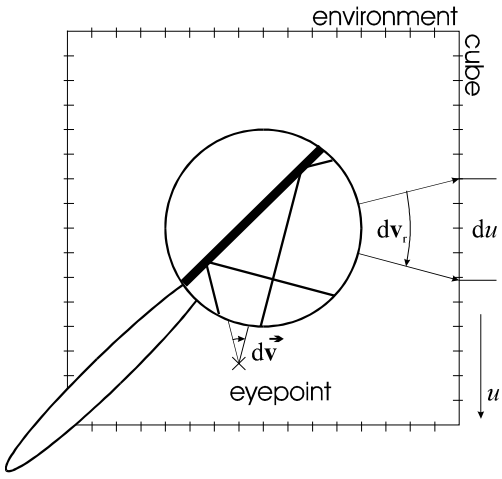

Fig. 2: *First contribution, caused by d***v**. *Reflecting object is plain.*

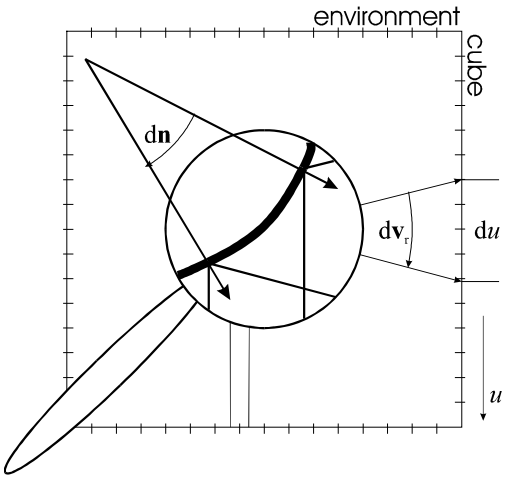

Fig. 3: *Second contribution, caused by d***n***. Viewing rays are parallel.*

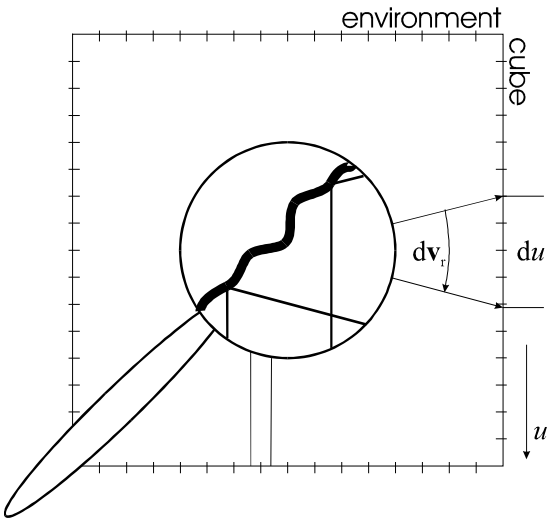

Fig. 4: *Third contribution caused by roughness of bump map. Viewing rays are parallel and underlying surface is plain.*

The calculation of the three contributions

The calculation of the first two contributions is performed by calculating the effect on the u - v -coordinates in the environment map, if the viewing ray is moved by one pixel in *x* or *y* direction. This effect is expressed by the derivatives of the environment map coordinates from the screen coordinates, which are grouped into the following matrix:

$$\mathbf{D} = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix} \qquad (1)$$

The points of a unit circle are transformed by this matrix into an ellipse; if this ellipse is centered around the u - v -coordinates of the center ray of the pixel, it describes the footprint in the environment map caused by a circular pixel. The quadratic form for the description of this ellipse is

$$\begin{pmatrix} u & v \end{pmatrix} \mathbf{K}_{1+2}^{-1} \begin{pmatrix} u \\ v \end{pmatrix} = 1 \qquad (2)$$

with

$$\mathbf{K}_{1+2} = \mathbf{D}^T \mathbf{D} \qquad (3)$$

The index „1+2“ denotes the relation to effects 1 and 2 (Fig. 2 and Fig. 3). The first task to be solved is to calculate the derivatives contained in the above matrix. We will perform this calculation for $\frac{\partial u}{\partial x}$; the other derivatives are evaluated analogously.
For the derivatives, we first look at the calculation of the reflected ray and the u - v -coordinates for the environment map.
First some definitions[1]:
The viewing ray has the direction $\mathbf{v}$ (which should not be mistaken for the environment map coordinate v). The viewing ray is reflected at the surface with surface normal $\mathbf{n}_{mod} = \mathbf{n} + b_1\mathbf{e}_1 + b_2\mathbf{e}_2$. The reflected ray can then be expressed as

$$\mathbf{v}_r = 2\mathbf{n}_{mod}\langle \mathbf{v} | \mathbf{n}_{mod} \rangle - \mathbf{v}|\mathbf{n}_{mod}|^2 \qquad (4)$$

Neither $\mathbf{v}$ nor $\mathbf{n}_{mod}$ needs to be normalized. We get: $|\mathbf{v}_r| = |\mathbf{v}||\mathbf{n}_{mod}|^2$. For the calculation of the environment map coordinates we use a cubical arrangement of the environment maps and assume that the coordinates are only dependent on the direction of the reflected ray. The coordinates u and v are thus calculated by dividing the two smaller components of $\mathbf{v}_r$ by its largest component. Thus the absolute value of $\mathbf{v}_r$ cancels out.

---

[1] We will use the following notation: Identity matrix: $\mathbf{I} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$,
dot product:
$\mathbf{a}^T\mathbf{b} = \langle \mathbf{a} | \mathbf{b} \rangle = a_x b_x + a_y b_y + a_z b_z$,
outer product:

$$\mathbf{a}\mathbf{b}^T = |\mathbf{a}\rangle\langle\mathbf{b}| = \begin{pmatrix} a_x b_x & a_x b_y & a_x b_z \\ a_y b_x & a_y b_y & a_y b_z \\ a_z b_x & a_z b_y & a_z b_z \end{pmatrix}$$

For greater flexibility, we can rotate the environment map cube by using $\mathbf{v}_r' = \mathbf{A}\mathbf{v}_r$ instead of $\mathbf{v}_r$, where $\mathbf{A}$ is a simple 3x3 rotation matrix. In the following we will omit $\mathbf{A}$ for the sake of simplicity.

We can now write:

$$\frac{\partial u}{\partial x} = \frac{\partial u}{\partial \mathbf{v}_r}\left(\frac{\partial \mathbf{v}_r}{\partial \mathbf{v}}\frac{\partial \mathbf{v}}{\partial x} + \frac{\partial \mathbf{v}_r}{\partial \mathbf{n}_{mod}}\frac{\partial \mathbf{n}_{mod}}{\partial x}\right) \quad (5)$$

If the x-component $v_{r,x}$ of $\mathbf{v}_r = \begin{pmatrix} v_{r,x} \\ v_{r,y} \\ v_{r,z} \end{pmatrix}$ is the one with the largest absolute value (we will assume that for the following; other cases are treated analogously) we get:

$$\frac{\partial u}{\partial \mathbf{v}_r} = \frac{1}{v_{r,x}}\begin{pmatrix} -u & 1 & 0 \end{pmatrix} \quad (6)$$

In eq. (5), the first summand in the brackets denotes the contribution shown in Fig. 2. As we can write for the (unnormalized) viewing vector:

$$\mathbf{v} = -\begin{pmatrix} x \\ y \\ z_0 \end{pmatrix} \quad (7)$$

with z_0 being the focal distance, $\frac{\partial \mathbf{v}}{\partial x}$ is the same constant vector ($\frac{\partial \mathbf{v}}{\partial x} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$) for all pixels of the screen.

For $\frac{\partial \mathbf{v}_r}{\partial \mathbf{v}}$ we get:

$$\frac{\partial \mathbf{v}_r}{\partial \mathbf{v}} = 2\left|\mathbf{n}\right\rangle\left\langle\mathbf{n}\right| - \left|\mathbf{n}\right|^2\mathbf{I} \quad (8)$$

The second addend in the brackets in denotes the contribution shown in Fig. 3.

For $\frac{\partial \mathbf{n}_{mod}}{\partial x}$ we can use the value of $\frac{\partial \mathbf{n}}{\partial x}$, as the roughness of the bump map is considered separately[2]. $\frac{\partial \mathbf{n}}{\partial x}$ is the change of the (unnormalized) surface normal, when we proceed by one pixel to the right. If $\mathbf{n}$ is interpolated across a polygon, this is a constant vector. This is true even if the interpolation is performed with perspective correction, as we need no normalized $\mathbf{n}$ and can omit the perspective division.

For $\frac{\partial \mathbf{v}_r}{\partial \mathbf{n}}$ we get:

$$\frac{\partial \mathbf{v}_r}{\partial \mathbf{n}_{mod}} = 2\left\langle\mathbf{n}_{mod}\middle|\mathbf{v}\right\rangle\mathbf{I} + 2\left|\mathbf{n}_{mod}\right\rangle\left\langle\mathbf{v}\right| - 2\left|\mathbf{v}\right\rangle\left\langle\mathbf{n}_{mod}\right| \quad (9)$$

With eq. (1-9) we can express $\mathbf{K}_{1+2}$ of eq. (3), which describes the footprint of the pixelbeam in the environment map caused by the aperture of the pixel beam and the curvature of the object. The factor that is not yet considered is the bump map, which can

- change the curvature of the surface, and
- scatter the pixel beam in multiple directions by small bumps.

To make things easier, we consider both effects independent of the underlying surface by using the roughness pyramids described in Appendix A. The roughness matrix $\mathbf{B}$ gives us a mapping from the unit circle to an ellipse that describes the distribution of perturbation vectors for the perturbation of the surface normals within one texel-area of the chosen resolution of the bump

[2] This could be changed at the expense of higher effort by considering the curvature of the *filtered* Bump-Map already in this stage and adding it to $\frac{\partial \mathbf{n}}{\partial x}$. Of course, in this case, this curvature may not be contained in the roughness pyramid. In the ideal case, $\frac{\partial \mathbf{n}_{mod}}{\partial x}$ and $\frac{\partial \mathbf{n}_{\mathrm{mod}}}{\partial y}$ would be determined such, that for the regarded area A with centerpoint (x_0, y_0) the Integral

$$\int_A \left| \frac{\mathbf{n}_{\mathrm{mod,filtered}}(x_0,y_0) + (x-x_0)\frac{\partial \mathbf{n}_{\mathrm{mod}}}{\partial x} + (y-y_0)\frac{\partial \mathbf{n}_{\mathrm{mod}}}{\partial y}}{\left|\mathbf{n}_{\mathrm{mod,filtered}}(x_0,y_0) + (x-x_0)\frac{\partial \mathbf{n}_{\mathrm{mod}}}{\partial x} + (y-y_0)\frac{\partial \mathbf{n}_{\mathrm{mod}}}{\partial y}\right|} - \frac{\mathbf{n}_{\mathrm{mod,unfiltered}}(x,y)}{\left|\mathbf{n}_{\mathrm{mod,unfiltered}}(x,y)\right|} \right|^2 dxdy$$

is minimized.

map. We are, however not primarily interested in the perturbation of the normals, but in the resulting perturbation of the reflected viewing ray $\mathbf{v}_r$ or, more precisely, in the perturbation of the original environment map coordinates that result from the perturbation of $\mathbf{v}_r$. Using a linear approximation, the ellipse of perturbation vectors to the normal vector can be transformed into an ellipse in the u - v -space of environment map coordinates, that results of the normal vector perturbation. The needed transformation can be written as:

$$\mathbf{M} = \begin{pmatrix} \frac{\partial u}{\partial \mathbf{v}_r} \\ \frac{\partial v}{\partial \mathbf{v}_r} \end{pmatrix} \frac{\partial \mathbf{v}_r}{\partial \mathbf{n}_{mod}}, \qquad (10)$$

which is a 3x2 Matrix. We have already calculated the needed expressions in eq. (6) and (9). Applying the transformation to $\mathbf{D}$ renders a 2x2 Matrix:

$$\mathbf{B}' = \mathbf{M}\begin{pmatrix} \mathbf{e}_1 & \mathbf{e}_2 \end{pmatrix}\mathbf{B}, \qquad (11)$$

which describes the reflection ellipse in the u - v - space of environment map coordinates.

$$\mathbf{K}_3 = \mathbf{B}'\mathbf{B}'^T \qquad (12)$$

is the roughness covariance matrix transformed into the environment map and represents the missing contribution to the footprint resulting from the bump map. As the contributions are assumed to be independent of each other, the covariance matrix can simply be added to the contribution described by $\mathbf{K}_{1+2}$:

$$\mathbf{K} = \mathbf{K}_{1+2} + \mathbf{K}_3 \qquad (13)$$

The approximated footprint of the pixel-beam in the environment map coordinates u - v is so described by the quadratic form

$$\mathbf{x}^T\mathbf{K}^{-1}\mathbf{x} = 1, \qquad (14)$$

where $\mathbf{K}$ contains the three factors that contribute to the size and shape of the pixel beam: the original angle of the pixel beam, the curvature of the object before bump mapping, and the roughness of the bump map. In order to employ anisotropic filtering for antialiasing the environment map, we have to find the main axes of the ellipse, which requires that the eigenvectors $\mathbf{r}_1$ and $\mathbf{r}_2$ and the eigenvalues λ_1^2 and λ_2^2 of $\mathbf{K}$ are determined:

$$\mathbf{K} = \begin{pmatrix} a & b \\ b & c \end{pmatrix} = \mathbf{R}\begin{pmatrix} \lambda_1^2 & 0 \\ 0 & \lambda_2^2 \end{pmatrix}\mathbf{R}^{-1}, \qquad (15)$$

with

$$\mathbf{R} = \begin{pmatrix} \mathbf{r}_1 & \mathbf{r}_2 \end{pmatrix}$$

and

$$\mathbf{R}^{-1} = \begin{pmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \end{pmatrix}$$

We use the following equations:

$$\lambda_1^2 = \frac{1}{2}\left(a + c + \sqrt{(c-a)^2 + 4b^2}\right), \qquad (16)$$

and

$$\lambda_2^2 = \frac{1}{2}\left(a + c - \sqrt{(c-a)^2 + 4b^2}\right) \qquad (17)$$

The eigenvectors are determined via table-lookup with the following equations:

$$\mathbf{r}_1 = \begin{pmatrix} \cos(\alpha) \\ -\sin(\alpha) \end{pmatrix} \qquad (18)$$

$$\mathbf{r}_2 = \begin{pmatrix} \sin(\alpha) \\ \cos(\alpha) \end{pmatrix} \qquad (19)$$

with

$$\tan(2\alpha) = \frac{2b}{c-a}, \qquad (20)$$

where α is restricted to be between - 45° and 45°. We address a table with this value, that contains about ten different values for $\sin(\alpha)$ and for $\cos(\alpha)$, distributed evenly over the angular range. The association of the eigenvalues and eigenvectors does, however, not relate to the above indices. But it can be found easily: $\mathbf{r}_1$ belongs to the larger eigenvalue, if $c - a > 0$ (the term $c - a$ is already used three times in eq. (16),(17) and (20) above).
With $\lambda_1 > \lambda_2$ and $\mathbf{r}_1$ being the eigenvector that belongs to the eigenvalue λ_1^2, the parameters for footprint assembly [3] are: $\lambda_1\mathbf{r}_1$ as stepping direction and -length, and λ_2 for the calculation of the mipmap level.

Further work

The presented algorithms perform position independent environment mapping. They can easily be extended to account for the object position, which means, that the environment cube is not any more considered to have infinite size. This would be especially advantageous, if nested environment cubes with partly transparent environment maps would be used.
The first order curvature should not be included in the roughness pyramid but should be added directly to the curvature of the underlying surface (see footnote 2 on page 4.
The antialiasing algorithms are suitable for an efficient hardware implementation, which would allow the generation of properly filtered,

environment mapped images of bump mapped objects in real time.

Conclusion

We have presented a method for antialiasing environment maps. The combination of bump and roughness maps with environment mapping provides the means to produce realistic effects, that have been missing up to now, often enough even in ray-traced images, e.g. small scrapes in glossy surfaces, rough surfaces seen from the distance, waves on a water-surface, seen from enough distance, etc.. Photo-realistic images of comparable quality can not be generated comparably fast and efficient; up to now, the only alternative is ray-tracing with costly antialiasing like massive supersampling. The principles, shown in this paper can of course also be applied to antialias textures in ray-tracing.

Example Images

Appendix A: The representation of bump maps.

In this appendix, we describe the representation of the bump and roughness masks we use. Bump mapping, even without antialiasing or environment mapping is not commonly used in real-time systems due to its heavy demands on the computing resources. This is partly, because the traditional approach to bump mapping [2] includes the calculation of the derivatives of the bump function.

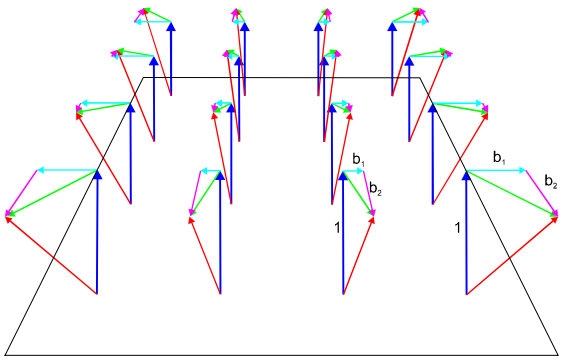

Fig. 5: *Representation of bump map with offset vectors.*

Bump mapping with Precalculated Derivatives

A possible solution that avoids the calculation of the derivatives of the bump function is to store precalculated derivatives [3,1]. Besides saving the calculation, this has the advantage, that the scaling of bump maps is as simple as the scaling of rgb-textures. If traditional bump maps are scaled with an unknown factor, it is impossible to calculate the derivatives any more. The most difficult problem that remains to be solved is to find an appropriate local coordinate system for each sample point, which consists of the normal vector of the surface in this point and the two tangential directions, for which the derivatives of the bump function have to be calculated (we have the same problem with the traditional representation of the bump maps). Once this coordinate system has been established, the calculation of the new normal vector is performed by adding the offset vector specified by the precalculated derivatives in the local coordinate system to the surface normal.

The Local Coordinate System

We use a local coordinate system $\mathbf{n}$, $\mathbf{e}_1$, $\mathbf{e}_2$ that meets two conditions:

- the directions of the axes are a continuous function of the location.
- the coordinate system is an orthogonal system.

The local coordinate system is derived from the normal vector $\mathbf{n}$ and a main direction $\mathbf{h}$. The unit vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ are perpendicular to $\mathbf{n}$. With the help of $\mathbf{h}$ they are defined such, that $\mathbf{e}_2$ is perpendicular to $\mathbf{h}$ and $\mathbf{e}_1$ is in the plane of $\mathbf{n}$ and $\mathbf{h}$ [3]. The main direction $\mathbf{h}$ can be interpolated across triangles or be a constant vector for a whole object. A good example is the mapping onto a sphere like e.g. the earth with spherical coordinates. The direction of the axis of the earth would serve as $\mathbf{h}$ and we would get $\mathbf{e}_1$ pointing always in west-east direction, and $\mathbf{e}_2$ in south north direction. An important advantage of a constant main direction $\mathbf{h}$ is that besides the normal vector, no other vector needs to be interpolated across triangles. In addition, the coordinate system can be calculated in hardware in the rasterizer/shader and needs not to be calculated at all vertices by a setup process.

The perturbed normal vector is then expressed as

$$\mathbf{n}_{mod} = \mathbf{n} + b_1\mathbf{e}_1 + b_2\mathbf{e}_2 .$$

The calculation of the local coordinate system $\mathbf{n}, \mathbf{e}_1, \mathbf{e}_2$ from the interpolated normal vector $\mathbf{n}_I$ and the main direction $\mathbf{h}$ is performed using the following formula (Fig. 6):

$$\mathbf{n} = \frac{\mathbf{n}_I}{|\mathbf{n}_I|},\ \mathbf{e}_1 = \frac{\mathbf{h}\times\mathbf{n}}{|\mathbf{h}\times\mathbf{n}|},\ \mathbf{e}_2 = \mathbf{n}\times\mathbf{e}_1 .$$

In this way, we get two tangential vectors:

- $\mathbf{e}_2$ in the plane defined by $\mathbf{n}$ and $\mathbf{h}$, and
- $\mathbf{e}_1$ perpendicular to that plane.

If the vectors need not be normalized, we multiply the three vectors by $|\mathbf{n}_I|$ for simpler calculation and get:

$$\mathbf{n}|\mathbf{n}_I| = \mathbf{n}_I, \mathbf{e}_1|\mathbf{n}_I| = \frac{\mathbf{h}\times\mathbf{n}}{|\mathbf{h}\times\mathbf{n}|}|\mathbf{n}_I|,$$

$$\mathbf{e}_2|\mathbf{n}_I| = \mathbf{n}_I \times \mathbf{e}_1.$$

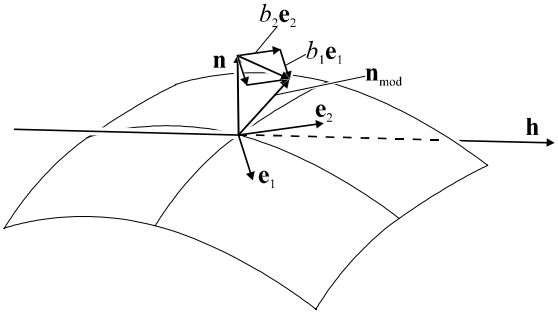

Fig. 6: *The construction of the local coordinate system for the bump map using a main direction* $\mathbf{h}$.

The roughness information for the bump map is stored in a roughness pyramid, that is calculated from the covariance matrices of b_1 and b_2 within the area represented by one texel (or better: roughxel). If the covariance matrix is

$$\mathbf{K} = \frac{1}{n}\begin{pmatrix} \sum_{i=0..n}(b_{1,i}-\overline{b_1})^2 & \sum_{i=0..n}(b_{1,i}-\overline{b_1})(b_{2,i}-\overline{b_2}) \\ \sum_{i=0..n}(b_{1,i}-\overline{b_1})(b_{2,i}-\overline{b_2}) & \sum_{i=0..n}(b_{2,i}-\overline{b_2})^2 \end{pmatrix} = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$$

,

- the roughness pyramid stores the matrix $\mathbf{B}$ with $\mathbf{B}\mathbf{B}^T = \mathbf{K}$.

We get a favorable representation, if we choose

$$\mathbf{B} = \begin{pmatrix} d_1 & d_2 \\ 0 & d_3 \end{pmatrix}$$

and store the three numbers d_1, d_2 and d_3, where d_1 and d_3 can be chosen to be nonnegative.

$$d_1 = \sqrt{a - \frac{b^2}{c}},$$

$$d_2 = \frac{b}{\sqrt{c}}, \text{ and}$$

$$d_3 = \sqrt{c}.$$

We have now the parameters of an ellipse, that describes the distribution of the perturbation vectors and by this the distribution of the normal vectors.

Bibliography

[1] Bennebroek, K., Ernst, I., Rüsseler, H., Wittig, O., Design Principles of Hardware-based Phong Shading and Bump Mapping, in Proceedings of the 11th Eurographics Workshop on Graphics Hardware, Poitiers Aug. 1996.

[2] Blinn, J., Simulation of Wrinkled Surfaces, SIGGRAPH 78, pp 286-292.

[3] Schilling, A., Knittel, G., Straßer, W., Texram: A Smart Memory for Texturing, Computer Graphics & Applications, May 1996, pp. 32-41.