Simulation of Wrinkled Surfaces Revisited

Morten Mikkelsen
morten_mikkelsen@naughtydog.com

March 26, 2008

Abstract

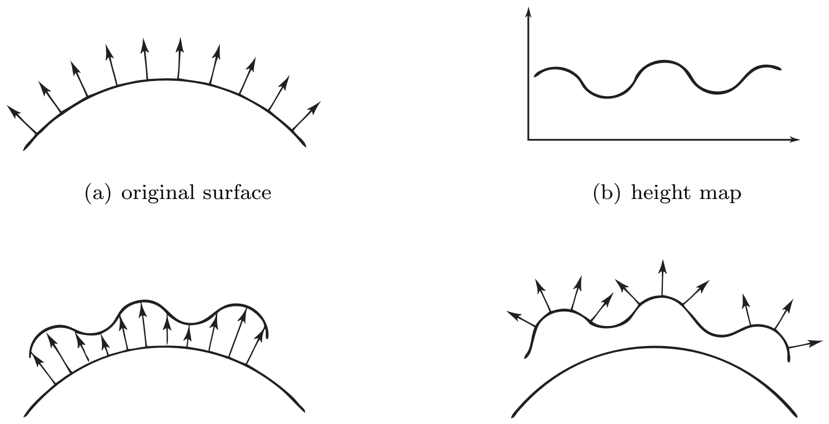
In this thesis, the work on simulation of wrinkled surfaces by Jim Blinn is revisited. Approximations for first–order derivatives were used for evaluation of perturbation of normals but an analysis of the impact of this approximation was never provided. Furthermore, Blinn only provides a solution for his simulation on shapes with a known bivariate surface parametrization which excludes triangular meshes with normals assigned at vertex level. Today, triangular meshes is the de facto standard, in the 3D graphics industry, to approximate arbitrary shapes including surfaces which are piecewise smooth. Blinn used the restriction that the 2D coordinate passed to evaluate the surface position is also used as the texture sampling coordinate. To tile or rotate a texture on a patch, modulation of the sampling coordinate is required. This restriction is significant, especially for triangular meshes, since texture coordinates assigned to each triangle is a modulation of the barycentric coordinate. This thesis remedies all these shortcommings. Finally, recent research on bump map filtering is described and evaluated.

A method known as normal mapping was introduced in recent years and it will be shown here how this approach relates to Jim Blinn's work. Today, normal mapping is almost as common as texture mapping and directly supported in a wide range of graphics tools. However, normal mapping is hard to get right without following a strict set of rules. A wrong implementation can lead to discontinuities in the shading. This thesis is focused on eliminating these visual artifacts. The result is an improved and consistent bump/normal mapping method that results in error and artifact free lighting of bumpy surfaces. The results are verified and compared visually and shown to outperform leading commercial graphics tools.

Contents

1 Introduction

In 1978, James Blinn published his paper [Bli78] on simulation of wrinkled surfaces. The aim of his paper was to mimic the high frequency surface irregularities that we see in reality on objects everywhere. His work is particularly focused on parametric surface patches and as Blinn points out these are smooth surfaces and tend to look artificial because of this. Since modeling such rich detail using patches would be very expensive and most likely a very daunting task, Blinn proposes recording such detail in a bump function/map over the surface patch. Hence the method is known as *bump mapping*. Conceptually the idea is to displace every point on the original patch by the assigned bump value along the surface normal (see figure 1). In particular this will perturb the surface


(a) original surface (b) height map

(c) new surface (d) perturbed normals

Figure 1: The process of normal perturbation is shown here in figures 1(a)-1(d).

normal of the original patch. Blinn argues that for simulation of wrinkles the displacement height is relatively small so the primary effect of the bumps will be in how the perturbation of the normal influences the reflection of the light and not so much the displaced position itself. Thus the initial surface is drawn but the perturbed normal is given, as opposed to the actual surface normal of the patch, as input to the lighting model. By this method, completion of an actual displacement of the surface patch is avoided.

Later on, it is pointed out by Cook [Coo84] that though the illusion is convincing, the lack of actual detail is revealed particularly in the silhouette around the patch as seen from the camera. His suggestion is to apply an actual displacement of a dense tessellation of the patch and then draw this instead. Of course for high frequency detail this tessellation must be very dense. He refers to this technique as *displacement mapping*.

Blinn bases his work on the assumption that the displacements can be considered arbitrarily small relative to the patch itself. As a result of this an approximation is made which will be covered in section 2.1. Though displacement mapping might provide pleasing results for significantly larger displacement values then those of wrinkled surfaces and minor surface irregularities the significance of Blinn's approximation was not reevaluated by Cook. In section 2.3 we will derive an accurate equation for the evaluation of the perturbed normal and provide a closer study of what the actual requirements are for Blinn's approximate perturbation to be successful. Computer graphics has a tendency to be very forgiving in terms of approximations made so it is quite possible the difference in terms of visual appearance in this case could be negligible. Nevertheless, our work in section 2.3 will be a study of the criteria necessary for a mathematical success and not a visual one. We will follow up in section 2.6 with a comparison of the visual differences.

In 1998, an alternative to Blinn's bump mapping was proposed by J. Cohen [Coh98] to simply replace the perturbed normals entirely by a normal map. In section 2.2, we will show how this method relates mathematically to Blinn's work.

In Blinn's paper, the patch and the bump map are both functions of the same parameter space. In general, this is not the case. For instance the position on a triangle is generally given as a function of the barycentric coordinate. The bump value, on the other hand, is given as a function of the texture coordinate. A reparametrization will allow us to specify both as functions of the same parameter space. This will be covered in section 2.4.

A significant problem never dealt with by Blinn is how to bump map triangular meshes. For such meshes it is common to let adjacent triangles share vertex normals derived from some weighted averaging process of face level normals. Such surface descriptions are approximations of a collection of curved surfaces of generally unknown parametrizations. Since Blinn's paper is based on the presence of a known surface parametrization, this initially appears to be a dead-end. As discussed in section 3.2 the problem has been identified before, but the amount of existing documentation is sparse and a thorough analysis of it has not been documented until now. Commercial products provide their own proprietary and thus undocumented solutions to the problem. The implications of this in terms of compatibility issues and other related problems are discussed in sections 3.2, 3.3 and 3.4. Specifically, several tests are presented which show a selection of leading industry products failing to provide good results given topological circumstances which will be explained. In most cases, the error appears as a discontinuity in the shading. Solutions are provided in these sections.

Ironically, when using rasterization–based algorithms for rendering, parametrized surfaces are tesselated and subsequently rasterized as triangular meshes with averaged vertex normals. Surprizingly, Blinn never addresses this issue nor does he indicate that his method should somehow depend on a ray tracing–based algorithm. We will discuss this further in section 2.5.3.

Technically, bump maps cannot be filtered like textures which represent color. The problem was identified by Blinn himself in his paper. Doing so anyway will simply smooth the bumps and thus neutralize the effect. Details are given in section 2.5.1 and a follow-up is made in section 4.1 of novel research on the subject which was presented at Siggraph 2007.

The reader is expected to be familiar with introductory level differential geometry and computer graphics. Furthermore, the reader must also be acquainted with the principles of mip mapping and trilinear filtering.

The source code was written in C++ and is available in a printable copy on the accompanying CD. Furthermore, a pdf version of this thesis is also available on the CD. The pictures in this thesis may, in some cases, not appear clear in the printed copy. In such a case the reader is referred to the pdf version.

2 Approximation and Reparametrization

Blinn's approximation of the perturbed normal is derived and given in section 2.1. Next, the close relation to the more recent technique normal mapping is explained in section 2.2. Specifically, the distinction between sampled normal maps and converted bump maps is made.

Since Blinn never gave an analysis of the implications of his approximation, or a clear justification for it, such an analysis will be given in section 2.3 along with a derivation of two equations for evaluation of the exact perturbed normal as opposed to Blinn's approximate normal. Additionally Jim Blinn uses the restriction that the 2D coordinate passed to evaluate the surface position is also used as the texture coordinate. This limitation is removed in section 2.4.

Blinn points out in his paper that applying a traditional filtration technique to a bump map will blur the bumps which simply cancels the effect of bump mapping. This is discussed in section 2.5.1. Furthermore, in section 2.5.2 a technique is given which converts a bump map, initially given as a single channel bitmap, into the form described in section 2.2. This makes it possible to deal with sampled normal maps and bump maps during shader execution in a uniform way. Finally in section 2.6 results are given.

2.1 Blinn's original simulation

Let $\beta : (u, v) \to \mathbf{R}$ represent a given height map (see figure 1(b)) and let the surface to be bump mapped be given as $\sigma : (u, v) \to \mathbf{R^3}$ where $(u, v) \in \mathbf{R^2}$ (see figure 1(a)). The corresponding unit normal on σ is $\vec{n} = \frac{\sigma_u \times \sigma_v}{\|\sigma_u \times \sigma_v\|}$.
Furthermore, let $\tau : (u, v) \to \mathbf{R^3}$ be the displaced surface defined as $\tau = \sigma + \beta \cdot \vec{n}$ which corresponds to the illustration shown in figure 1(c).
The first–order derivatives of τ are computed by the following equations

$$\begin{aligned} \tau_u &= \sigma_u + \beta_u \cdot \vec{n} + \beta \cdot \vec{n}_u \\ \tau_v &= \sigma_v + \beta_v \cdot \vec{n} + \beta \cdot \vec{n}_v \end{aligned}$$

The aim is to obtain an expression for the normal on τ shown in figure 1(d). Blinn argues that for simulation of wrinkles we can assume that β is small relative to the extent of the surface and subsequently the approximation is made that the first–order derivatives can be obtained as

$$\begin{aligned} \tau_u &\simeq \sigma_u + \beta_u \cdot \vec{n} \\ \tau_v &\simeq \sigma_v + \beta_v \cdot \vec{n} \end{aligned}$$

Which leads to

$$\begin{aligned} \tau_u \times \tau_v &\simeq (\sigma_u + \beta_u \cdot \vec{n}) \times (\sigma_v + \beta_v \cdot \vec{n}) \\ &= \sigma_u \times \sigma_v + \beta_u \cdot \vec{n} \times \sigma_v + \beta_v \cdot \sigma_u \times \vec{n} \end{aligned} \quad (1)$$

This will be referred to as Blinn's normal or alternatively the approximate surface normal of τ. Note that the vector given by equation (1) is generally not a unit vector and will have to be normalized before use in the shader.

2.2 Normal Mapping

While bump mapping perturbs the existing normal on a surface, normal mapping [Coh98] replaces the normal entirely by doing a look-up into a *normal map*. In [Coh98] the normals of the normal map are stored in the space in which σ is contained, ie. *object space*. By using normal maps instead of bump maps, we are no longer limited to perturbation of the existing normals on σ. As an example of this acquired generality, a method to enhance visual quality is suggested by [Coh98], where normals are sampled from a detailed high resolution surface description τ and stored as a texture map. These will be referred to as *sampled normal maps*.

Each mapped texel corresponds to a point on a low resolution surface σ and from this point a ray is traced along the surface normal of σ and onto τ. The surface normal on τ at the intersection point is then stored in the current texel of the texture map (see figure 2).

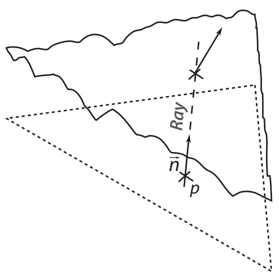


Figure 2: For every texel, a ray is shot from the corresponding surface point p on the low resolution surface and along the normal $\vec{n}$. At the intersection with the high resolution surface, the normal there is sampled and recorded.

Normal mapping can also be used to achieve bump mapping. As an offline process, for every texel in the normal map which corresponds to a surface position on σ, we evaluate the perturbed normal using equation (1). Note that normal maps are three–channel textures as opposed to bump maps which consist of a single channel only.

A downside is that such object space normal maps depend on the surface σ. In contrast, when bump mapping, β is independent of σ and can be used to perturb the normals of any surface. This is clearly a very strong advantage that should be preserved. It will be explained in the following how reusability of bump maps can be preserved by using *tangent space* normal maps as opposed to object space.

In differential geometry, tangent space is defined as a two-dimensional space such that for any $(u, v) \in \mathbf{R^2}$, the vector $u \cdot \sigma_u + v \cdot \sigma_v$ is contained in tangent space. In the graphics community, tangent space is generally defined as a three-dimensional space and its basis is the linearly independent set $\{\sigma_u, \sigma_v, \vec{n}\}$ (see

[Kil00]). This definition is also used in this thesis. The extended transformation out of tangent space for a vector $(u, v, w) \in \mathbf{R^3}$ is achieved by the linear combination $u \cdot \sigma_u + v \cdot \sigma_v + w \cdot \vec{n}$. This transformation can be applied using the matrix

$$N = \left[\begin{array}{c|c|c} \sigma_u & \sigma_v & \vec{n} \end{array} \right]$$

and the inverse transformation can easily be determined using the following

$$\begin{aligned} det(N) &= \sigma_u \bullet (\sigma_v \times \vec{n}) \\ &= \sigma_v \bullet (\vec{n} \times \sigma_u) \\ &= \vec{n} \bullet (\sigma_u \times \sigma_v) \\ &= \|\sigma_u \times \sigma_v\| \end{aligned}$$

where the symbol $\bullet$ denotes the dot product between two vectors. The inverse is thus given as

$$N^{-1} = \frac{1}{\|\sigma_u \times \sigma_v\|} \left[\begin{array}{c} \sigma_v \times \vec{n} \\ \hline \vec{n} \times \sigma_u \\ \hline \sigma_u \times \sigma_v \end{array} \right]$$

since the dot product between the ith row of N^{-1} and the jth column of N is zero when $i \neq j$ and one otherwise for $i, j \in \{1, 2, 3\}$.

Let $\vec{\gamma} : (u, v) \rightarrow \mathbf{R^3}$ represent a tangent space normal map. Since normals like planes are transformed using the inverse transposed, we obtain the object space normal given the following expression

$$(N^{-1})^T \cdot \vec{\gamma} = \frac{\vec{\gamma}_1 \cdot \sigma_v \times \vec{n} + \vec{\gamma}_2 \cdot \vec{n} \times \sigma_u + \vec{\gamma}_3 \cdot \sigma_u \times \sigma_v}{\|\sigma_u \times \sigma_v\|}$$

Here $\vec{\gamma}_1$, $\vec{\gamma}_2$ and $\vec{\gamma}_3$ denote the first, second, and third component function of $\vec{\gamma}$.

Let $h : (u, v) \rightarrow \mathbf{R^3}$ be the graph of β such that $h(u, v) = (u, v, \beta)$. Now let us examine the special case where $\vec{\gamma} = h_u \times h_v$, that is the case where $h_u \times h_v$ by definition equals the accurate tangent space normal map of σ. If we apply the transformation to this map we obtain the following result.

$$\begin{aligned} (N^{-1})^T \cdot \vec{\gamma} &= (N^{-1})^T \cdot h_u \times h_v \\ &= (N^{-1})^T \cdot (1, 0, \beta_u) \times (0, 1, \beta_v) \\ &= (N^{-1})^T \cdot (-\beta_u, -\beta_v, 1) \\ &= \frac{\sigma_u \times \sigma_v + \beta_u \cdot \vec{n} \times \sigma_v + \beta_v \cdot \sigma_u \times \vec{n}}{\|\sigma_u \times \sigma_v\|} \end{aligned} \tag{2}$$

Since the normal will have to be normalized before use, we can omit the denominator in which case we rediscover Blinn's original equation (1). However since the problem has been redefined the equation is technically no longer based on approximation, in practice the result for this particular case is the very same. Given equation (2), bump maps will be converted, into normal maps, by storing $(-\beta_u, -\beta_v, 1)$ as the tangent space normal. These are clearly defined independently of σ and will be referred to as *converted bump maps*. Transformation by

$(N^{-1})^T$ is applied in the shader which completes the transformation into object space.

In order to achieve a uniform processing of sampled normal maps and converted bump maps, we will transform the sampled normals into tangent space by applying N^T to them since $(N^{-1})^T \cdot N^T$ is the identity. This way in either case, we obtain the final object space normal in the shader by applying $(N^{-1})^T$. Note that unlike converted bump maps, sampled normal maps transformed into tangent space depend on σ. The transformation applied in the shader should be the exact inverse of the transformation applied when the normal map was generated. This is the only way to obtain the original sampled object space normal which is used during execution of the lighting model.

2.3 Removing Blinn's Approximation step

In this section, we will retrace Blinn's steps and reevaluate equation (1), but this time we will omit the part which is based on approximation.
Let the coefficients of the first fundamental form be given as

$$\begin{aligned} E &= \sigma_u \bullet \sigma_u \\ F &= \sigma_u \bullet \sigma_v \\ G &= \sigma_v \bullet \sigma_v \end{aligned}$$

and those of the second fundamental form as

$$\begin{aligned} e &= \sigma_{uu} \bullet \vec{n} \\ &= -\vec{n}_u \bullet \sigma_u \\ f &= \sigma_{uv} \bullet \vec{n} \\ &= -\vec{n}_u \bullet \sigma_v \\ g &= \sigma_{vv} \bullet \vec{n} \\ &= -\vec{n}_v \bullet \sigma_v \end{aligned}$$

Furthermore, we define the symmetric 2×2 matrices

$$\begin{aligned} \mathcal{F}_{\mathcal{I}} &= \begin{bmatrix} E & F \\ F & G \end{bmatrix} \\ \mathcal{F}_{\mathcal{II}} &= \begin{bmatrix} e & f \\ f & g \end{bmatrix} \end{aligned}$$

The matrix $\mathcal{W} = \mathcal{F}_{\mathcal{I}}{}^{-1}\mathcal{F}_{\mathcal{II}}$ is known from differential geometry as the *Weingarten matrix*. Let the negated coefficients be defined as

$$\begin{aligned} \begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix} &= -\mathcal{W} \qquad (3) \\ &= \frac{1}{EG - F^2} \begin{bmatrix} fF - eG & gF - fG \\ eF - fE & fF - gE \end{bmatrix} \end{aligned}$$

it is additionally known from differential geometry that

$$\vec{n}_u = w_1\sigma_u + w_3\sigma_v \tag{4}$$
$$\vec{n}_v = w_2\sigma_u + w_4\sigma_v \tag{5}$$

This is also given in proposition 6.4 of [Pre01]. The *principal curvatures* κ_1 and κ_2 are the eigenvalues of $\mathcal{W}$. By definition, if $\begin{pmatrix} \xi & \eta \end{pmatrix}^T$ is the eigenvector corresponding to κ_i then $\vec{p_i} = \xi\sigma_u + \eta\sigma_v$ is the *principal vector* corresponding to κ_i.
The matrix $\mathcal{W}$ is generally not a symmetric matrix so the spectral theorem does not apply. However, proposition 6.3(i) of [Pre01] tells us that $\mathcal{W}$ does in fact have eigenvalues which by definition are real. Proposition 6.3(ii) tells us that if $\kappa_1 = \kappa_2$, then any direction in the tangent plane is a principal vector and finally from proposition 6.3(iii) it follows that when $\kappa_1 \neq \kappa_2$, then $\vec{p}_1$ and $\vec{p}_2$ are perpendicular to each other. From corollary 8.16 of Robert Messer [Mes97] an $m \times m$ matrix with m distinct, eigenvalues is diagonalizable. We know that the eigenvalues of $\mathcal{W}$ exist so if they are additionally distinct it follows that $\mathcal{W}$ is diagonalizable.

$$\begin{aligned}\mathcal{W} &= -\begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix} \\ &= \frac{1}{ad-bc}\begin{bmatrix} a & c \\ b & d \end{bmatrix}\begin{bmatrix} \kappa_1 & 0 \\ 0 & \kappa_2 \end{bmatrix}\begin{bmatrix} d & -c \\ -b & a \end{bmatrix} \qquad (6) \\ &= \frac{1}{ad-bc}\begin{bmatrix} ad\kappa_1 - bc\kappa_2 & ac(\kappa_2-\kappa_1) \\ bd(\kappa_1-\kappa_2) & ad\kappa_2 - bc\kappa_1 \end{bmatrix} \qquad (7)\end{aligned}$$

Which by definition of the principal vectors leads to

$$\vec{p}_1 = a\sigma_u + b\sigma_v \tag{8}$$
$$\vec{p}_2 = c\sigma_u + d\sigma_v \tag{9}$$

If $\|\vec{p}_1\| \neq 1$ we can substitute a by $\frac{a}{\|\vec{p}_1\|}$ and b by $\frac{b}{\|\vec{p}_1\|}$. By applying the same principle to $\vec{p}_2$ we can proceed under the assumption that $\vec{p}_1$ and $\vec{p}_2$ are both unit vectors and it follows that $\{\vec{p}_1, \vec{p}_2, \vec{n}\}$ is an orthonormal set.
Furthermore, proposition 7.1 in [Pre01] tells us that the determinant of $\mathcal{W}$ equals the Gaussian curvature $\kappa_1\kappa_2$ and the trace equals $\kappa_1 + \kappa_2$. This gives us the following equations.

$$\kappa_1\kappa_2 = w_1w_4 - w_3w_2 \tag{10}$$
$$\kappa_1 + \kappa_2 = -(w_1 + w_4) \tag{11}$$

From equation (7) it follows that

$$\begin{aligned}\begin{bmatrix} w_4 & -w_3 \\ -w_2 & w_1 \end{bmatrix} &= \frac{-1}{ad-bc}\begin{bmatrix} ad\kappa_2 - bc\kappa_1 & bd(\kappa_2-\kappa_1) \\ ac(\kappa_1-\kappa_2) & ad\kappa_1 - bc\kappa_2 \end{bmatrix} \\ &= \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}\begin{bmatrix} -\kappa_2 & 0 \\ 0 & -\kappa_1 \end{bmatrix}\begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad (12)\end{aligned}$$

which we will be using shortly. The exact perturbed normal is evaluated by crossing the first–order derivatives of τ.

$$\begin{aligned}
\tau_u \times \tau_v &= (\sigma_u + \beta_u \cdot \vec{n} + \beta \cdot \vec{n}_u) \times (\sigma_v + \beta_v \cdot \vec{n} + \beta \cdot \vec{n}_v) \\
&= \sigma_u \times \sigma_v + \beta^2 \vec{n}_u \times \vec{n}_v + \beta\,(\vec{n}_u \times \sigma_v + \sigma_u \times \vec{n}_v) + \\
&\quad \beta_u \vec{n} \times \sigma_v + \beta_v \sigma_u \times \vec{n} + \\
&\quad \beta\,(\beta_u \vec{n} \times \vec{n}_v + \beta_v \vec{n}_u \times \vec{n})
\end{aligned}$$

This expression is complicated but using equations (4) and (5) allows us to get rid of the terms $\vec{n}_u$ and $\vec{n}_v$ and provides a nice matrix formulation.

$$\begin{aligned}
\tau_u \times \tau_v &= \big(1 + \beta^2 (w_1 w_4 - w_3 w_2) + \beta\,(w_1 + w_4)\big)\, \sigma_u \times \sigma_v + \\
&\quad \beta_u \vec{n} \times \sigma_v + \beta_v \sigma_u \times \vec{n} + \\
&\quad \beta\,(\beta_u \vec{n} \times \vec{n}_v + \beta_v \vec{n}_u \times \vec{n}) \\
&= \big(1 + \beta^2 (w_1 w_4 - w_3 w_2) + \beta\,(w_1 + w_4)\big)\, \sigma_u \times \sigma_v + \\
&\quad \beta_u \vec{n} \times \sigma_v + \beta_v \sigma_u \times \vec{n} + \\
&\quad \beta\,(\beta_u w_4 \vec{n} \times \sigma_v - \beta_v w_3 \vec{n} \times \sigma_v) + \\
&\quad \beta\,(-\beta_u w_2 \sigma_u \times \vec{n} + \beta_v w_1 \sigma_u \times \vec{n}) \\
&= (\beta_u + \beta\,(\beta_u w_4 - \beta_v w_3))\, \vec{n} \times \sigma_v + \\
&\quad (\beta_v + \beta\,(-\beta_u w_2 + \beta_v w_1))\, \sigma_u \times \vec{n} + \\
&\quad \big(1 + \beta^2 (w_1 w_4 - w_3 w_2) + \beta\,(w_1 + w_4)\big)\, \sigma_u \times \sigma_v \\
&= \|\sigma_u \times \sigma_v\| (N^{-1})^T \begin{bmatrix} 1 + \beta w_4 & -\beta w_3 & 0 \\ -\beta w_2 & 1 + \beta w_1 & 0 \\ 0 & 0 & 1 + \beta^2 (w_1 w_4 - w_3 w_2) + \beta\,(w_1 + w_4) \end{bmatrix} \begin{bmatrix} -\beta_u \\ -\beta_v \\ 1 \end{bmatrix}
\end{aligned}$$

This leaves us with two matrices on the left side. Let

$$\begin{aligned}
A &= \|\sigma_u \times \sigma_v\| (N^{-1})^T \\
&= \left[\ \sigma_v \times \vec{n} \ \middle|\ \vec{n} \times \sigma_u \ \middle|\ \sigma_u \times \sigma_v\ \right] \\
M &= \begin{bmatrix} 1 + \beta w_4 & -\beta w_3 & 0 \\ -\beta w_2 & 1 + \beta w_1 & 0 \\ 0 & 0 & 1 + \beta^2 (w_1 w_4 - w_3 w_2) + \beta\,(w_1 + w_4) \end{bmatrix} \qquad (13)
\end{aligned}$$

Now we can write the normal as

$$\tau_u \times \tau_v = A \cdot M \cdot \begin{bmatrix} -\beta_u \\ -\beta_v \\ 1 \end{bmatrix}$$

By using equations (10) and (11) we can express the coefficient M_{33} as

$$\begin{aligned}
1 + \beta^2 (w_1 w_4 - w_3 w_2) + \beta\,(w_1 + w_4) &= 1 + \beta^2 \kappa_1 \kappa_2 - \beta\,(\kappa_1 + \kappa_2) \\
&= (1 - \beta \kappa_1)(1 - \beta \kappa_2)
\end{aligned}$$

When κ_1 and κ_2 are distinct we can use equation (12) and write the matrix M given the following equation

$$M = \frac{1}{ad - bc} \begin{bmatrix} d & -b & 0 \\ -c & a & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 - \beta\kappa_2 & 0 & 0 \\ 0 & 1 - \beta\kappa_1 & 0 \\ 0 & 0 & (1 - \beta\kappa_1)(1 - \beta\kappa_2) \end{bmatrix} \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and subsequently by defining the following matrices

$$\begin{aligned} B &= \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ S &= \begin{bmatrix} 1 - \beta\kappa_2 & 0 & 0 \\ 0 & 1 - \beta\kappa_1 & 0 \\ 0 & 0 & (1 - \beta\kappa_1)(1 - \beta\kappa_2) \end{bmatrix} \end{aligned}$$

the matrix M is given by the equation

$$M = B^{-1} \cdot S \cdot B \tag{14}$$

As previously mentioned, $\{\vec{p}_1, \vec{p}_2, \vec{n}\}$ is an orthonormal set which means the matrix

$$\begin{aligned} P &= \left[\ \vec{p}_1 \mid \vec{p}_2 \mid \vec{n}\ \right] \\ &= N \cdot B^T \end{aligned} \tag{15}$$

is orthogonal which leads to $P^{-1} = P^T$. Furthermore, equation (15) follows from equations (8) and (9) and from equation (15) we are given

$$\begin{aligned} B &= \left(N^{-1} \cdot P\right)^T \\ &= P^T \cdot \left(N^{-1}\right)^T \end{aligned}$$

We can now rewrite the transformation sequence as

$$\begin{aligned} A \cdot M &= A \cdot B^{-1} \cdot S \cdot B \\ &= A \cdot N^T \cdot (P^{-1})^T \cdot S \cdot P^T \cdot (N^{-1})^T \\ &= \|\sigma_u \times \sigma_v\| \cdot P \cdot S \cdot P^T \cdot (N^{-1})^T \\ &= P \cdot S \cdot P^T \cdot A \end{aligned}$$

And the final normal of τ is

$$\tau_u \times \tau_v = A \cdot M \cdot \begin{bmatrix} -\beta_u \\ -\beta_v \\ 1 \end{bmatrix} \tag{16}$$

$$= P \cdot S \cdot P^T \cdot A \cdot \begin{bmatrix} -\beta_u \\ -\beta_v \\ 1 \end{bmatrix} \tag{17}$$

In practice, we will be using equation (16) for evaluation of the perturbed normal because it is simpler since it does not require that we find the principal curvatures nor the principal vectors and it does not have a singularity at $\kappa_1 = \kappa_2$. However, what is very interesting about the form seen in equation (17) is the insight it gives us into the nature of Blinn's approximate form (1). On the right side of (17) we have the transformation by A which according to section 2.2 and the definition of A is the original approximate perturbed normal. On the left side, we have $P \cdot S \cdot P^T$ which is symmetric and corresponds to scaling by $(1 - \beta\kappa_2)$, $(1 - \beta\kappa_1)$ and $(1 - \beta\kappa_1)(1 - \beta\kappa_2)$ along the directions $\vec{p}_1$, $\vec{p}_2$ and $\vec{n}$ respectively. It is obvious that $P \cdot S \cdot P^T$ approaches the identity matrix as $\beta\kappa_1$ and $\beta\kappa_2$ approach zero. This observation provides a more accurate interpretation of the approximation used by Blinn. The fact is the approximation holds well when β is small relative to the principal curvatures or vice versa. Blinn is not entirely wrong when he says small relative to the extent of the surface since the curvature is diminished as σ is physically enlarged (scaled up). However, if at a given point on σ there is an extreme amount of curvature, then σ would have to be scaled up accordingly by a very large value for the approximation to hold. On the other hand, for a plane since the curvature is zero it does not matter how far up or down the plane is scaled, in this case equation (1) will provide accurate results.

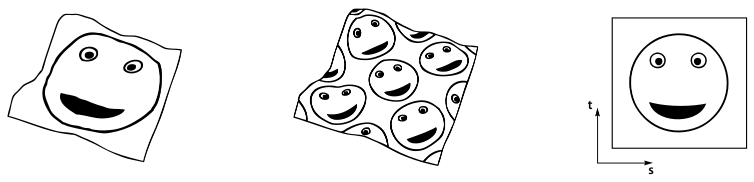
Technically equation (17) is only valid when $\kappa_1 \neq \kappa_2$. However, as previously mentioned, any direction in the tangent plane is a principal vector when $\kappa_1 = \kappa_2$. So in this case we can arbitrarily choose $\vec{p}_1$ and $\vec{p}_2$ as any orthonormal set in the tangent plane such that P remains an orthogonal matrix. And so our analysis based on equation (17) of Blinn's approximation is still valid when the principal curvatures are not distinct.

2.4 Reparametrized Surface

In Blinn's paper, the surface σ and the height map β are both functions of (u, v). Since texture coordinates are often specified or manipulated by artists, we instead generalize by allowing these to be given as a diffeomorphism of (u, v). The concept is illustrated in figure 3. In other words, we allow the bump map to be given as a reparametrization of β and the surface to be given as a reparametrization of σ. In this section, we will examine how reparametrization of β and σ affects evaluation of the perturbed normal on τ. Let the map $\Phi : (s, t) \to (u, v) \in \mathbf{R^2}$ be a diffeomorphism and let

$$\begin{aligned}
\Lambda(s,t) &= \tau(\Phi(s,t)) \\
\chi(s,t) &= \sigma(\Phi(s,t)) \\
\alpha(s,t) &= \beta(\Phi(s,t))
\end{aligned}$$

such that Λ, χ and α are reparametrizations of τ, σ and β. The objective is to find nice forms similar to equations (16) and (17) to evaluate the normal of τ



(a) (b) (c)

Figure 3: The image in 3(c) represents the bump map. The surface σ is shown in figures 3(a) and 3(b), but in 3(a) the sampling coordinate is not subjected to modulation which maps the bump map directly onto the surface. In 3(b) modulation takes place prior to sampling.

as a function of (s,t), that is $\tau_u \times \tau_v(\Phi(s,t))$.

$$\begin{aligned} J(\Phi) &= \begin{bmatrix} \frac{d\Phi_1}{ds} & \frac{d\Phi_1}{dt} \\ \frac{d\Phi_2}{ds} & \frac{d\Phi_2}{dt} \end{bmatrix} \\ \chi_s \times \chi_t &= \left(\frac{d\Phi_1}{ds}\sigma_u + \frac{d\Phi_2}{ds}\sigma_v\right) \times \left(\frac{d\Phi_1}{dt}\sigma_u + \frac{d\Phi_2}{dt}\sigma_v\right) \\ &= \left(\frac{d\Phi_1}{ds}\frac{d\Phi_2}{dt} - \frac{d\Phi_2}{ds}\frac{d\Phi_1}{dt}\right)\sigma_u \times \sigma_v \\ &= \det[J(\Phi)]\sigma_u \times \sigma_v \end{aligned} \tag{18}$$

Since χ is the reparametrization of σ, we will interpret $\sigma_u \times \sigma_v$ as the true orientation of the surface which is important in the evaluation of τ since we displace along the normal and, as equation (18) indicates, the orientation of the surface is reversed due to reparametrization when $\det[J(\Phi)] < 0$ and the orientation is preserved when $\det[J(\Phi)] > 0$. The function Φ is only a permissible reparametrization if $\det[J(\Phi)] \neq 0$ everywhere and since Φ is smooth and thereby continuous, it follows that

$$\begin{aligned} \rho &= \frac{\det[J(\Phi)]}{|\det[J(\Phi)]|} \\ &= \pm 1 \end{aligned}$$

is constant and either one or minus one. Let $\vec{m}$ be the unit normal of χ equivalent to $\vec{n}$ of σ

$$\begin{aligned} \vec{m} &= \frac{\chi_s \times \chi_t}{\|\chi_s \times \chi_t\|} \\ \vec{n} &= \rho \cdot \vec{m} \end{aligned} \tag{19}$$

As previously mentioned, displacement takes place along $\vec{n}$ so using equation (19) the reparametrization of τ is given as

$$\Lambda(s,t) = \chi(s,t) + \alpha(s,t)(\rho \cdot \vec{m})$$

If we use the following rewrite of Λ

$$\begin{aligned} \delta(s,t) &= \rho \cdot \alpha(s,t) \\ \Lambda(s,t) &= \chi(s,t) + \delta(s,t) \cdot \vec{m} \end{aligned} \tag{20}$$

where δ is the new bump map, then equation (20) corresponds to the definition of τ in section 2.1 which allows us to evaluate the surface normal of Λ using equation (16). In order to do so, the matrices A and M of section 2.3 need to be evaluated. The matrix A is trivially initialized using the vectors χ_s, χ_t and $\vec{m}$ as input but the matrix M requires work. We need to analyze what happens to the weingarten map under reparametrization. The first–order derivatives are given by the chain rule as

$$\begin{aligned} \chi_s &= \frac{d\Phi_1}{ds}\sigma_u(\Phi(s,t)) + \frac{d\Phi_2}{ds}\sigma_v(\Phi(s,t)) \\ \chi_t &= \frac{d\Phi_1}{dt}\sigma_u(\Phi(s,t)) + \frac{d\Phi_2}{dt}\sigma_v(\Phi(s,t)) \end{aligned}$$

and the resulting second–order derivatives are as follows

$$\begin{aligned} \chi_{ss} &= \left(\frac{d\Phi_1}{ds}\right)^2 \sigma_{uu} + \left(\frac{d\Phi_2}{ds}\right)^2 \sigma_{vv} + 2\frac{d\Phi_1}{ds}\frac{d\Phi_2}{ds}\sigma_{uv} + \frac{d^2\Phi_1}{ds^2}\sigma_u + \frac{d^2\Phi_2}{ds^2}\sigma_v \\ \chi_{st} &= \frac{d\Phi_1}{ds}\frac{d\Phi_1}{dt}\sigma_{uu} + \frac{d\Phi_2}{ds}\frac{d\Phi_2}{dt}\sigma_{vv} + \left(\frac{d\Phi_1}{ds}\frac{d\Phi_2}{dt} + \frac{d\Phi_2}{ds}\frac{d\Phi_1}{dt}\right)\sigma_{uv} + \frac{d^2\Phi_1}{dtds}\sigma_u + \frac{d^2\Phi_2}{dtds}\sigma_v \\ \chi_{tt} &= \left(\frac{d\Phi_1}{dt}\right)^2 \sigma_{uu} + \left(\frac{d\Phi_2}{dt}\right)^2 \sigma_{vv} + 2\frac{d\Phi_1}{dt}\frac{d\Phi_2}{dt}\sigma_{uv} + \frac{d^2\Phi_1}{dt^2}\sigma_u + \frac{d^2\Phi_2}{dt^2}\sigma_v \end{aligned}$$

From the first–order derivatives, the first fundamental form is

$$\begin{aligned} E_\chi &= \chi_s \bullet \chi_s \\ &= \left(\frac{d\Phi_1}{ds}\right)^2 E_\sigma + 2\frac{d\Phi_1}{ds}\frac{d\Phi_2}{ds}F_\sigma + \left(\frac{d\Phi_2}{ds}\right)^2 G_\sigma \\ F_\chi &= \chi_s \bullet \chi_t \\ &= \frac{d\Phi_1}{ds}\frac{d\Phi_1}{dt}E_\sigma + \left(\frac{d\Phi_1}{ds}\frac{d\Phi_2}{dt} + \frac{d\Phi_1}{dt}\frac{d\Phi_2}{ds}\right)F_\sigma + \frac{d\Phi_2}{ds}\frac{d\Phi_2}{dt}G_\sigma \\ G_\chi &= \chi_t \bullet \chi_t \\ &= \left(\frac{d\Phi_1}{dt}\right)^2 E_\sigma + 2\frac{d\Phi_1}{dt}\frac{d\Phi_2}{dt}F_\sigma + \left(\frac{d\Phi_2}{dt}\right)^2 G_\sigma \end{aligned}$$

and next the second fundamental form yields

$$\begin{aligned}
e_\chi &= \vec{m} \bullet \chi_{ss} \\
&= \rho\left(\left(\frac{d\Phi_1}{ds}\right)^2 e_\sigma + 2\frac{d\Phi_1}{ds}\frac{d\Phi_2}{ds} f_\sigma + \left(\frac{d\Phi_2}{ds}\right)^2 g_\sigma\right) \\
f_\chi &= \vec{m} \bullet \chi_{st} \\
&= \rho\left(\frac{d\Phi_1}{ds}\frac{d\Phi_1}{dt} e_\sigma + \left(\frac{d\Phi_1}{ds}\frac{d\Phi_2}{dt} + \frac{d\Phi_1}{dt}\frac{d\Phi_2}{ds}\right) f_\sigma + \frac{d\Phi_2}{ds}\frac{d\Phi_2}{dt} g_\sigma\right) \\
g_\chi &= \vec{m} \bullet \chi_{tt} \\
&= \rho\left(\left(\frac{d\Phi_1}{dt}\right)^2 e_\sigma + 2\frac{d\Phi_1}{dt}\frac{d\Phi_2}{dt} f_\sigma + \left(\frac{d\Phi_2}{dt}\right)^2 g_\sigma\right)
\end{aligned}$$

And from these we obtain the following matrix formulations

$$\begin{aligned}
\mathcal{F}_{\chi\mathcal{I}} &= \begin{bmatrix} E_\chi & F_\chi \\ F_\chi & G_\chi \end{bmatrix} = J(\Phi)^T \cdot \mathcal{F}_{\sigma\mathcal{I}} \cdot J(\Phi) \\
\mathcal{F}_{\chi\mathcal{I}\mathcal{I}} &= \begin{bmatrix} e_\chi & f_\chi \\ f_\chi & g_\chi \end{bmatrix} = \rho \cdot \left(J(\Phi)^T \cdot \mathcal{F}_{\sigma\mathcal{I}\mathcal{I}} \cdot J(\Phi)\right)
\end{aligned}$$

Now, similar to equation 3, we define the following matrix

$$\begin{bmatrix} \zeta_1 & \zeta_2 \\ \zeta_3 & \zeta_4 \end{bmatrix} = -J(\Phi)^{-1} \cdot \mathcal{W}_\sigma \cdot J(\Phi) \tag{21}$$

such that by evaluation of the weingarten matrix, it follows that

$$\begin{aligned}
\mathcal{W}_\chi &= {\mathcal{F}_{\chi\mathcal{I}}}^{-1} \cdot \mathcal{F}_{\chi\mathcal{I}\mathcal{I}} \\
&= \rho \cdot \left(J(\Phi)^{-1} \cdot {\mathcal{F}_{\sigma\mathcal{I}}}^{-1} \cdot \mathcal{F}_{\sigma\mathcal{I}\mathcal{I}} \cdot J(\Phi)\right) \\
&= \rho \cdot \left(J(\Phi)^{-1} \cdot \mathcal{W}_\sigma \cdot J(\Phi)\right) \qquad (22) \\
&= (-\rho) \cdot \begin{bmatrix} \zeta_1 & \zeta_2 \\ \zeta_3 & \zeta_4 \end{bmatrix}
\end{aligned}$$

Now, similar to section 2.3, we evaluate the matrix A and use $-\mathcal{W}_\chi$ and $\delta(s,t)$ to evaluate the matrix M.

$$\begin{aligned}
A_\chi &= \|\chi_s \times \chi_t\| \left(\left[\begin{array}{c|c|c} \chi_s & \chi_t & \vec{m} \end{array}\right]^{-1}\right)^T \qquad (23) \\
&= \left[\begin{array}{c|c|c} \chi_t \times \vec{m} & \vec{m} \times \chi_s & \chi_s \times \chi_t \end{array}\right] \\
M_{\chi,\delta} &= \begin{bmatrix} 1+\delta\cdot(\rho\cdot\zeta_4) & -\delta\cdot(\rho\cdot\zeta_3) & 0 \\ -\delta\cdot(\rho\cdot\zeta_2) & 1+\delta\cdot(\rho\cdot\zeta_1) & 0 \\ 0 & 0 & 1+\delta^2\rho^2\left(\zeta_1\zeta_4-\zeta_3\zeta_2\right)+\delta\cdot\rho\cdot(\zeta_1+\zeta_4) \end{bmatrix} \\
&= \begin{bmatrix} 1+\alpha\zeta_4 & -\alpha\zeta_3 & 0 \\ -\alpha\zeta_2 & 1+\alpha\zeta_1 & 0 \\ 0 & 0 & 1+\alpha^2\left(\zeta_1\zeta_4-\zeta_3\zeta_2\right)+\alpha\left(\zeta_1+\zeta_4\right) \end{bmatrix} \qquad (24)
\end{aligned}$$

This completes evaluation of the matrices A_χ and $M_{\chi,\delta}$ and using equation 16 gives the normal $\Lambda_s \times \Lambda_t$. The normal we actually need though is $\tau_u \times \tau_v(\Phi(s,t))$ but similar to equation (18) we have $\Lambda_s \times \Lambda_t = \det[J(\Phi)]\tau_u \times \tau_v$ and as in equation (19) it follows that

$$\frac{\tau_u \times \tau_v}{\|\tau_u \times \tau_v\|} = \rho \cdot \frac{\Lambda_s \times \Lambda_t}{\|\Lambda_s \times \Lambda_t\|}$$

Since the perturbed normal must be normalized before use anyway, the length is insignificant so the above equation can be scaled on both sides by $\|\Lambda_s \times \Lambda_t\|$ and thus using equation (16) on $\Lambda_s \times \Lambda_t$, the result is

$$\begin{aligned}
\frac{\|\Lambda_s \times \Lambda_t\|}{\|\tau_u \times \tau_v\|} \cdot \tau_u \times \tau_v &= \rho \cdot \Lambda_s \times \Lambda_t \\
&= \rho \cdot A_\chi \cdot M_{\chi,\delta} \cdot \begin{bmatrix} -\delta_s \\ -\delta_t \\ 1 \end{bmatrix} \\
&= \rho \cdot A_\chi \cdot M_{\chi,\delta} \cdot \begin{bmatrix} -\rho \cdot \alpha_s \\ -\rho \cdot \alpha_t \\ 1 \end{bmatrix} \\
&= \rho \cdot \left[\ \rho \cdot \chi_t \times \vec{m} \mid \rho \cdot \vec{m} \times \chi_s \mid \chi_s \times \chi_t\ \right] \cdot M_{\chi,\delta} \cdot \begin{bmatrix} -\alpha_s \\ -\alpha_t \\ 1 \end{bmatrix} \\
&= \rho \cdot \left[\ \chi_t \times \vec{n} \mid \vec{n} \times \chi_s \mid \chi_s \times \chi_t\ \right] \cdot M_{\chi,\delta} \cdot \begin{bmatrix} -\alpha_s \\ -\alpha_t \\ 1 \end{bmatrix} \\
&= \rho \cdot (\rho \cdot \|\chi_s \times \chi_t\|) \cdot \left(\left[\ \chi_s \mid \chi_t \mid \vec{n}\ \right]^{-1}\right)^T \cdot M_{\chi,\delta} \cdot \begin{bmatrix} -\alpha_s \\ -\alpha_t \\ 1 \end{bmatrix} \\
&= \|\chi_s \times \chi_t\| \cdot \left(\left[\ \chi_s \mid \chi_t \mid \vec{n}\ \right]^{-1}\right)^T \cdot M_{\chi,\delta} \cdot \begin{bmatrix} -\alpha_s \\ -\alpha_t \\ 1 \end{bmatrix}
\end{aligned}$$

The above equation provides a consistent evaluation equation for $\tau_u \times \tau_v$ even under reparametrization.
In the following, a similar result will be derived for equation (17). Using equation (22) and assuming κ_1 and κ_2 are distinct, we can write the weingarten matrix as

$$\mathcal{W}_\chi = \frac{1}{ad-bc} J(\Phi)^{-1} \cdot \begin{bmatrix} a & c \\ b & d \end{bmatrix} \begin{bmatrix} \rho \cdot \kappa_1 & 0 \\ 0 & \rho \cdot \kappa_2 \end{bmatrix} \begin{bmatrix} d & -c \\ -b & a \end{bmatrix} \cdot J(\Phi)$$

where equation (6) is used for $\mathcal{W}_\sigma$. By defining the matrix

$$Q = \begin{bmatrix} q_1 & q_3 \\ q_2 & q_4 \end{bmatrix} = J(\Phi)^{-1} \cdot \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

the columns of Q are the eigenvectors of $\mathcal{W}_\chi$ so by definition the principal vectors are

$$\begin{aligned}\vec{p}_{\chi 1} &= q_1 \cdot \chi_s + q_2 \cdot \chi_t \\ \vec{p}_{\chi 2} &= q_3 \cdot \chi_s + q_4 \cdot \chi_t\end{aligned}$$

If we rewrite this to matrix form and take advantage of the first–order derivative equations

$$\begin{aligned}\begin{bmatrix} \vec{p}_{\chi 1} & \vec{p}_{\chi 2} \end{bmatrix} &= \begin{bmatrix} \chi_s & \chi_t \end{bmatrix} \cdot Q \\ &= \big(\begin{bmatrix} \sigma_u(\Phi(s,t)) & \sigma_v(\Phi(s,t)) \end{bmatrix} \cdot J(\Phi) \big) \cdot \left(J(\Phi)^{-1} \cdot \begin{bmatrix} a & c \\ b & d \end{bmatrix} \right) \\ &= \begin{bmatrix} \sigma_u(\Phi(s,t)) & \sigma_v(\Phi(s,t)) \end{bmatrix} \cdot \begin{bmatrix} a & c \\ b & d \end{bmatrix} \\ &= \begin{bmatrix} \vec{p}_{\sigma 1}(\Phi(s,t)) & \vec{p}_{\sigma 2}(\Phi(s,t)) \end{bmatrix}\end{aligned}$$

we rediscover the principal vectors we had before reparametrization, so they are unchanged and we will continue to refer to these as $\vec{p}_1$ and $\vec{p}_2$. As in section 2.3, we proceed by evaluating the matrices P and S but for the reparametrized surface.

$$\begin{aligned}P_\chi &= \begin{bmatrix} \vec{p}_1 \mid \vec{p}_2 \mid \vec{m} \end{bmatrix} \\ &= P_\sigma \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \rho \end{bmatrix} \\ S_{\chi,\delta} &= \begin{bmatrix} 1-\delta\rho\kappa_2 & 0 & 0 \\ 0 & 1-\delta\rho\kappa_1 & 0 \\ 0 & 0 & (1-\delta\rho\kappa_1)(1-\delta\rho\kappa_2) \end{bmatrix} \\ &= \begin{bmatrix} 1-\alpha\kappa_2 & 0 & 0 \\ 0 & 1-\alpha\kappa_1 & 0 \\ 0 & 0 & (1-\alpha\kappa_1)(1-\alpha\kappa_2) \end{bmatrix} \\ &= S_{\sigma,\beta}\end{aligned}$$

Since we have $S_{\chi,\delta} = S_{\sigma,\beta}$, we will henceforth refer to the matrix simply as S. Next we will use P_χ and S to evaluate equation (17) which gives us the second form under reparametrization.

$$\begin{aligned}
\frac{\|\Lambda_s \times \Lambda_t\|}{\|\tau_u \times \tau_v\|} \cdot \tau_u \times \tau_v &= \rho \cdot \Lambda_s \times \Lambda_t \\
&= \rho \cdot P_\chi \cdot S \cdot P_\chi^T \cdot A_\chi \cdot \begin{bmatrix} -\delta_s \\ -\delta_t \\ 1 \end{bmatrix} \\
&= P_\sigma \cdot S \cdot P_\sigma^T \cdot (\rho \cdot A_\chi) \cdot \begin{bmatrix} -\rho \cdot \alpha_s \\ -\rho \cdot \alpha_t \\ 1 \end{bmatrix} \\
&= P_\sigma \cdot S \cdot P_\sigma^T \cdot \|\chi_s \times \chi_t\| \cdot \left(\left[\begin{array}{c|c|c} \chi_s & \chi_t & \vec{n} \end{array} \right]^{-1} \right)^T \cdot \begin{bmatrix} -\alpha_s \\ -\alpha_t \\ 1 \end{bmatrix}
\end{aligned}$$

The matrices P_χ and P_σ are generally not the same. However, given the result of the equation above, we only have a need to refer to P_σ which will henceforth be denoted simply P.

Based on the findings of this section, we clean up notation by replacing the matrices A and M by N' and M' respectively

$$\begin{aligned}
M' &= \begin{bmatrix} 1+\alpha\zeta_4 & -\alpha\zeta_3 & 0 \\ -\alpha\zeta_2 & 1+\alpha\zeta_1 & 0 \\ 0 & 0 & 1+\alpha^2(\zeta_1\zeta_4 - \zeta_3\zeta_2) + \alpha(\zeta_1+\zeta_4) \end{bmatrix} \\
N' &= \left[\begin{array}{c|c|c} \chi_s & \chi_t & \vec{n} \end{array} \right]
\end{aligned}$$

where M' as given by equation (24) corresponds to M but evaluated based on the coefficients given by (21) as opposed to those given by (3). The matrix N' corresponds to N but using the **forward** facing unit normal in the third column and the first–order derivatives of the reparametrized surface in the first and second column. So in conclusion, what we have established is that under reparametrization of σ and β the direction of the perturbed normal in the intended orientation is evaluated using the following equation

$$\begin{aligned}
\frac{\tau_u \times \tau_v}{\|\sigma_u \times \sigma_v\|} &= \left(N'^{-1}\right)^T \cdot M' \cdot \begin{bmatrix} -\alpha_s \\ -\alpha_t \\ 1 \end{bmatrix} & (25) \\
&= P \cdot S \cdot P^T \cdot \left(N'^{-1}\right)^T \cdot \begin{bmatrix} -\alpha_s \\ -\alpha_t \\ 1 \end{bmatrix} & (26)
\end{aligned}$$

Here the change in scalars on both sides is the result of

$$\begin{aligned}
\|\chi_s \times \chi_t\| &= |\det[J(\Phi)]| \cdot \|\sigma_u \times \sigma_v\| \\
\|\Lambda_s \times \Lambda_t\| &= |\det[J(\Phi)]| \cdot \|\tau_u \times \tau_v\|
\end{aligned}$$

2.5 Practicalities

The bump map $\alpha(s,t)$ is interpreted as a smooth function but is often simply supplied by artists as a grayscale texture. However this data is still interpreted as a fixed finite selection of sampled data from some unknown smooth (or at least continuous) function in the way filtering is performed during sampling. Blinn mentions that for color textures filtering is performed by averaging the region of texels covered by the current pixel to be drawn. He adds that applying the same approach to bump textures would simply smooth the bumps as opposed to average the varied light intensities that result from the bumps. Blinn argues that the correct solution is to sample all the bumps covered by the current pixel, computing the light intensity for every perturbed normal and then average these intensities instead. This approach is of course ineffective and Blinn himself does not follow this path and comments that for the more offensive aliasing artifacts filtering the bumps using a traditional approach such as for color textures will do.

2.5.1 Bump map representation and filtering

In Blinn's paper, only the normal is replaced by the perturbed variant during light intensity calculation. The position used lies on σ and not the displaced surface τ and the matrix N' is also entirely derived from surface properties of σ. Given these observations we could consider it an approximation that in terms of filtering for each pixel on the screen only one matrix N' and only one position on σ is used in the execution of the lighting model regardless of the amount of bump map texels in the coverage. This results in a single vector towards the light and subsequently also a single vector towards the camera and finally a single halfway vector (see section 16.1.4 in [FvDFH95]) used for Blinn-Phong specular high–lights. Filtering the dot product between such a fixed direction $\vec{l}$ and each of the perturbed normals using a fixed N' is equal to simply filtering the derivatives α_s and α_t

$$\sum_{i=1}^{I} w_i \cdot \vec{l} \bullet \left((N'^{-1})^T \cdot \begin{bmatrix} -\alpha[i]_s \\ -\alpha[i]_t \\ 1 \end{bmatrix} \right) = \vec{l} \bullet \left((N'^{-1})^T \cdot \sum_{i=1}^{I} w_i \cdot \begin{bmatrix} -\alpha[i]_s \\ -\alpha[i]_t \\ 1 \end{bmatrix} \right)$$

where I is the amount of texels covered and w_i is the weight assigned to each texel such that $\sum_{i=1}^{I} w_i = 1$. Of course this does not fully justify simply filtering the derivatives since for actual evaluation of a diffuse intensity, you also need clamping against zero which does not result in the same inside the summation as opposed to outside. The same problem exists in regards to the exponent associated with the specular intensity. A recent paper [HSRG07] provides new research in the area of normal map filtering and this will be covered in section 4.1.

As mentioned in section 2.2, there exists a close correspondence between bump mapping and normal mapping. We can achieve a uniform processing of the two forms by always using a four-channel texture with the tangent space normal in the first three channels and the height in the fourth. Bump maps must initially be converted such that the tangent space normal is computed as $\begin{bmatrix} -\alpha_s & -\alpha_t & 1 \end{bmatrix}^T$. Conversely sampled normal maps by definition already have the normal handy but are not equipped with heights, so these will have the fourth channel set to zero. When the heights are zero the matrix M' will result in the identity matrix which yields the correct equation for normal mapping as covered in sections 2.2 and 2.3. While rendering, these four-channel textures will be sampled using traditional trilinear filtering as described in [EWWL98]. A thorough walkthrough of trilinear filtering is considered beyond the scope of this thesis and so only details specific to parametrized surfaces, which are not covered in [EWWL98], will be discussed in the following. The interested reader is referred to the paper itself.

To use the method described in this paper, the texture coordinates must be expressed as a linear function of the surface position $p = (x, y, z)$. The paper provides an equation to achieve such a map for triangles specifically.

$$\begin{aligned} s(x,y,z) &= \begin{bmatrix} s_1 & s_2 & s_3 \end{bmatrix} \cdot \begin{bmatrix} p_1 \mid p_2 \mid p_3 \end{bmatrix}^{-1} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} \\ t(x,y,z) &= \begin{bmatrix} t_1 & t_2 & t_3 \end{bmatrix} \cdot \begin{bmatrix} p_1 \mid p_2 \mid p_3 \end{bmatrix}^{-1} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} \end{aligned}$$

In order to make the approach work not just for triangles but also for parametrized surfaces, we take advantage of the fact that we know the first–order derivatives χ_s and χ_t. In the following these and the intersection point p_0 are assumed to be given in camera space. So for such a given intersection point p_0 with the corresponding texture coordinate (s_0, t_0) we substitute in the above equation using the following positions p_0, $p_0 + \chi_s$ and $p_0 + \chi_t$ and the texture coordinates (s_0, t_0), $(s_0 + 1, t_0)$ and $(s_0, t_0 + 1)$. After substitution we obtain the equations

$$\begin{aligned} s(x,y,z) &= \begin{bmatrix} s_0 & s_0 + 1 & s_0 \end{bmatrix} \cdot \begin{bmatrix} p_0 \mid p_0 + \chi_s \mid p_0 + \chi_t \end{bmatrix}^{-1} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} \\ &= \frac{s_0 \cdot (\chi_s \times \chi_t) + \chi_t \times p_0}{p_0 \bullet (\chi_s \times \chi_t)} \bullet \begin{pmatrix} x \\ y \\ z \end{pmatrix} \qquad (27) \\ t(x,y,z) &= \begin{bmatrix} t_0 & t_0 & t_0 + 1 \end{bmatrix} \cdot \begin{bmatrix} p_0 \mid p_0 + \chi_s \mid p_0 + \chi_t \end{bmatrix}^{-1} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} \\ &= \frac{t_0 \cdot (\chi_s \times \chi_t) + p_0 \times \chi_s}{p_0 \bullet (\chi_s \times \chi_t)} \bullet \begin{pmatrix} x \\ y \\ z \end{pmatrix} \qquad (28) \end{aligned}$$

An expression for the reciprocal of z as a function of the pixel coordinate is also required by [EWWL98] for filtering. We may provide such a function from the first–order derivatives and the intersection point.

$$\begin{aligned}
(\chi_s \times \chi_t) \bullet \left(\begin{pmatrix} x \\ y \\ z \end{pmatrix} - p_0 \right) &= 0 \\
&\Longleftrightarrow \\
(\chi_s \times \chi_t) \bullet \begin{pmatrix} x \\ y \\ z \end{pmatrix} &= p_0 \bullet (\chi_s \times \chi_t) \\
&\Longleftrightarrow \\
\frac{(\chi_s \times \chi_t) \bullet \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \\ 1 \end{pmatrix}}{p_0 \bullet (\chi_s \times \chi_t)} &= \frac{1}{z} \qquad (29)
\end{aligned}$$

As in [EWWL98] the following substitutions $X = \frac{x}{z}$ and $Y = \frac{y}{z}$ are used to represent pixel coordinates. Let the coefficients given in equations (27), (28) and (29) be known as

$$\begin{aligned}
\vec{v}_1 &= \frac{s_0 \cdot (\chi_s \times \chi_t) + \chi_t \times p_0}{p_0 \bullet (\chi_s \times \chi_t)} \\
\vec{v}_2 &= \frac{t_0 \cdot (\chi_s \times \chi_t) + p_0 \times \chi_s}{p_0 \bullet (\chi_s \times \chi_t)} \\
\vec{v}_3 &= \frac{\chi_s \times \chi_t}{p_0 \bullet (\chi_s \times \chi_t)}
\end{aligned}$$

From this it follows that we can express the texture coordinate divided by z and the reciprocal of z as functions of the pixel coordinate.

$$\begin{aligned}
\frac{s(x,y,z)}{z} &= \vec{v}_1 \bullet \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \qquad (30) \\
\frac{t(x,y,z)}{z} &= \vec{v}_2 \bullet \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \qquad (31) \\
\frac{1}{z} &= \vec{v}_3 \bullet \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \qquad (32)
\end{aligned}$$

The optimal choice for texture dimension is given by [EWWL98] as

$$d(z) = \frac{1}{\sqrt{|k \cdot z^3|}} \qquad (33)$$

where the value k is given as the determinant of the 3×3 matrix containing $\vec{v}_1$, $\vec{v}_2$ and $\vec{v}_3$ in the columns.

$$V = \left[\ \vec{v}_1 \mid \vec{v}_2 \mid \vec{v}_3\ \right]$$

Since input is given in camera space, $(X, Y) = (\frac{x}{z}, \frac{y}{z})$ does not really correspond to actual pixel coordinates. If we assume that s_x and s_y are user–defined values which will scale X and Y into the proper range, then we can substitute X and Y by $\frac{s_x \cdot x}{z}$ and $\frac{s_y \cdot y}{z}$ respectively. To preserve results in equations (30), (31) and (32), the first and second component of $\vec{v}_1$, $\vec{v}_2$ and $\vec{v}_3$ is scaled by $\frac{1}{s_x}$ and $\frac{1}{s_y}$ respectively. As an alternative, since it is the objective to evaluate equation (33), we can simply adjust evaluation of k. The constant k is the determinant of the matrix V and scaling a single row or column results in scaling the determinant. From this it follows that k equals

$$\begin{aligned} k &= \frac{1}{s_x \cdot s_y} \cdot \det(V) \\ &= \frac{1}{s_x \cdot s_y} \cdot \vec{v}_1 \bullet (\vec{v}_2 \times \vec{v}_3) \\ &= \frac{1}{s_x \cdot s_y} \cdot \frac{1}{p_0 \bullet (\chi_s \times \chi_t)} \end{aligned}$$

We can now reevaluate the optimal texture dimension as

$$\begin{aligned} d(z) &= \frac{1}{\sqrt{|k \cdot z^3|}} \\ &= \sqrt{\left| \frac{(s_x \cdot s_y) \cdot p_0 \bullet (\chi_s \times \chi_t)}{z^3} \right|} \end{aligned} \qquad (34)$$

This evaluation is based on properties which are known for a parametrized surface, that is the surface point and the first–order derivatives. When a ray tracing algorithm is used, equation (34) is evaluated using the third component of the intersection point p_0 as the input into $d(z)$. Note that the term $p_0 \bullet (\chi_s \times \chi_t)$, which is also seen in the denominator for $\vec{v}_1$, $\vec{v}_2$ and $\vec{v}_3$, is zero exactly when the tangent plane contains the eye-point of the camera. In such a limit case the surface at the intersection point is invisible seen from the camera. We need, however, to convince ourselves that the result in equation (34) still makes sense as this term approaches zero. In this case the requested resolution d will also approach zero which corresponds to a small, in terms of resolution, mip map level. This makes sense since when the surface is almost perpendicular to the view-plane of the camera, a larger area of the surface is covered by a single pixel which means more texels are covered. As for the third component of the intersection point in the denominator, we know this is non zero since the intersection point would otherwise not have been determined visible.

Trilinear mip mapping does not take the shape or the orientation of the coverage into account when sampling is performed. It is referred to as an isotropic method which means directionally independent. The concept is shown in figure 4(b) where the red region represents the true coverage, in the texture map, of the screen pixel shown in figure 4(a). The square transparent green region with a dashed outline represents the approximation of the true coverage used in trilinear mip mapping. The square region is chosen such that its area size is identical

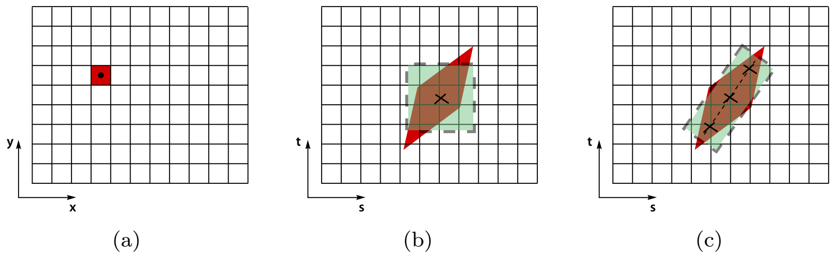


Figure 4: In 4(b) the red region represents the coverage in a texture map by the screen pixel shown in 4(a). The green transparent square represents the approximate sampling area used when performing trilinear sampling. The green oriented rectangular region shown in 4(c) represents the approximate sampling area used when performing anisotropic texture filtering.

to the true coverage. Anisotropic texture filtering takes the shape and orientation into account as shown in figure 4(c) where the coverage is approximated by an oriented rectangular region. Trilinear sampling is performed repeatedly along the rectangular region, roughly as many times as the width tiles along the height. Different techniques and variants of trilinear sampling and anisotropic texture filtering exist. Such techniques are used to determine the distribution of weights associated with the texels prior to sampling. A detailed description of such techniques is beyond the scope of this thesis. The primary concern in section 4.1 is how the chosen distribution of weights should be used in conjunction with bump/normal mapping. For additional details on trilinear sampling and anisotropic texture filtering, the reader is referred to [EWWL98] and [MPFJ99] respectively.

2.5.2 Conversion of bump maps

To convert a bump map into a normal map as explained in section 2.5.1, a procedure for evaluation of the derivatives α_s and α_t must be determined. This presents a problem since an exact function which describes the applied bump map is generally not known, only a two dimensional array of samples is known. For such an array of $m \times n$ samples, these are typically considered to be recorded values of an unknown function $\alpha(s,t) \rightarrow [0;1]$ where $(s,t) \in [0;1] \times [0;1]$ and the distance between consecutive samples is $\frac{1}{m}$ and $\frac{1}{n}$ in the horizontal and vertical directions respectively. For a given coordinate (s_0, t_0), an approximation for the first–order derivatives with respect to s and t can be determined based on the surrounding texels using central differencing.

$$\begin{aligned}
\alpha_s &\simeq \frac{\alpha(s_0+\frac{1}{m},t_0)-\alpha(s_0-\frac{1}{m},t_0)}{\left(s_0+\frac{1}{m}\right)-\left(s_0-\frac{1}{m}\right)} \\
&= \frac{m}{2}\cdot\left(\alpha(s_0+\frac{1}{m},t_0)-\alpha(s_0-\frac{1}{m},t_0)\right) \qquad (35)\\
\alpha_t &\simeq \frac{\alpha(s_0,t_0+\frac{1}{n})-\alpha(s_0,t_0-\frac{1}{n})}{\left(t_0+\frac{1}{n}\right)-\left(t_0-\frac{1}{n}\right)} \\
&= \frac{n}{2}\cdot\left(\alpha(s_0,t_0+\frac{1}{n})-\alpha(s_0,t_0-\frac{1}{n})\right) \qquad (36)
\end{aligned}$$

If we apply equation (35) at a given texel of the height map, then clearly the term on the right side of the product is the difference between the sample on the right and the one on the left. Similarly for equation (36), it is the difference between the upper sample and the lower. We can calculate these differences for every texel using the edge detection kernel in a convolution.

$$E_s = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \qquad E_t = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

Optionally, to reduce noise, we can apply a Gaussian blur in the opposite direction of the edge detection. Due to separability, this translates into the Sobel kernels.

$$\begin{aligned}
G_s &= \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \\
G_t &= \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}
\end{aligned}$$

The operator $*$ represents convolution. If sobel is used to evaluate the difference, then a scale by $\frac{1}{4}$ must be applied to normalize the blur.

As previously mentioned, the recorded values are considered to be in the range $[0;1]$. Proportionally, this range might not be suitable relative to the extent of the surface $\chi(s,t)$. To solve this, a user–defined scalar value λ is introduced which is applied to $\alpha(s,t)$. As a result of this, the derivatives α_s and α_t are scaled by the same value.

2.5.3 Drawing parametrized surfaces

As mentioned in the introduction, Blinn's paper is based on surfaces for which the parametrization is known. However, rasterizers cannot render such surfaces accurately and do so simply by tessellation to a triangular mesh. As tessellation takes place, the parameter value (u,v) corresponding to each generated vertex is stored, and as rasterization takes place, these are barycentrically interpolated and at pixel level the resulting parameters are used to evaluate (s,t) and subsequently the matrices M' and N' as defined in section 2.4.

Alternatively, ray tracing could be used to avoid the tessellation step. However, even ray tracers have problems when it comes to parametrized surfaces. Often the expression for the intersection between a ray and a parametrized surface is either unknown or too complex for evaluation. A good example of this is Bezier patches, the approach often used today by ray tracers (see [Ben06]) is subdivision followed by Newton–Raphson.

2.6 Results

As mentioned in section 1, a significant aspect of Blinn's original work we wish to reevaluate is an approximation on which his work is based (see section 2.1). In other words, we would like to compare the results we get using Blinn's bump mapping to an accurate implementation. As explained throughout section 2, we need a surface parametrization to explore this issue in order to have accurate first– and second–order derivatives. It is tempting to resort to a graphics API like OpenGL for the implementation. However, as mentioned in section 2.5.3, we cannot accurately render a parametrized surface because tessellation is required. Additionally graphics chips internally use approximations for math functions and interpolation of attributes across triangles. As mentioned, ray tracing does not provide a general solution to these problems. However, we do not need a solution which works for all parametrized surfaces. We can make do with a few chosen test-cases on which the intersection can be determined using an exact approach. Furthermore, with a ray tracer we still have the ability to draw triangular meshes. So for these reasons, ray tracing has been chosen for this implementation.

2.6.1 Monkey Saddle

For the first test, the monkey saddle as a parametrized surface is used.

$$z = x^3 - 3x \cdot y^2, \quad x, y \in [-1;1] \times [-1;1]$$

By inserting the parametrization of the ray into this equation we obtain a polynomial of third degree and the intersections are given by the roots.
Since $(u,v) \in [0;1]$, we define the equations

$$\begin{aligned} x(u,v) &= 2 \cdot u - 1 \\ y(u,v) &= 2 \cdot v - 1 \end{aligned}$$

and furthermore, the diffeomorphism $\Phi : (s,t) \rightarrow (u,v)$ initially mentioned in section 2.4 is in this case set to the identity map. So (u,v) is used directly for sampling of the converted height map. The initial height map which is applied to this surface is seen in figure 5(a). The stored values of the first three texel components of the converted texture are $(-\alpha_s, -\alpha_t, 1)$. The logic behind this was explained in section 2.2 and 2.5.1. In the fourth and last component the height value itself is stored. The height is used for evaluation of the matrix M' given at the end of section 2.4. For accuracy and convenience, the converted texture is stored as 32–bit floating point values. As mentioned in section 2.5.2, the heights are understood by default to be values between zero and one, so a user-defined value is given which scales the displacements into a range which is proportionally suitable relative to the surface. Figures 5(b), 5(c) and 5(d) show the first three channels of the converted maps given the user–defined scales $\frac{1}{32}$, $\frac{1}{16}$ and $\frac{1}{4}$. To be able to show these as images, a normalization is applied and

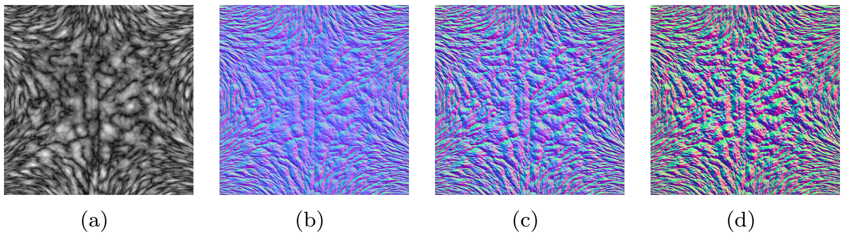
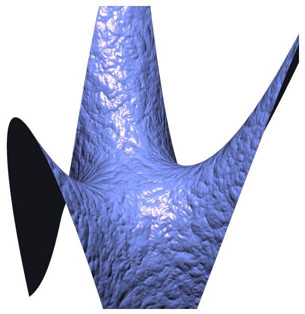
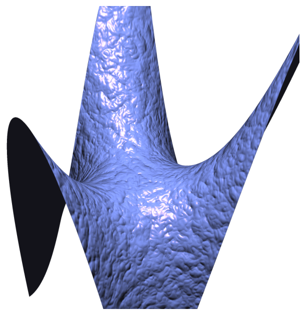
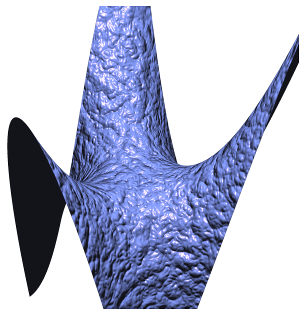


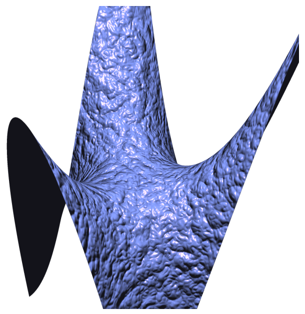
Figure 5: In 5(a) a bump map is shown. In 5(b) the corresponding converted bump map is shown given a small user–scale $\frac{1}{32}$ applied to the heights. It appears blue because the normals tend to point upward in the Z-direction. The user–scale is doubled in 5(c) and doubled again in 5(d). Based on the diminished presence of blue we can see that the normals are pulled outward in the tangent plane.

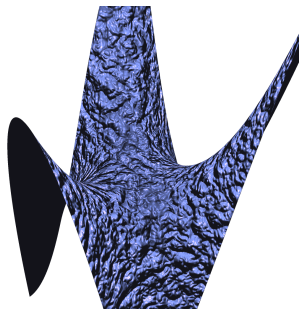
finally to fit the range to $[0; 1]$, the resulting three channels are scaled and added by $\frac{1}{2}$. As mentioned in section 2.5.2, when the displacements $\alpha(s, t)$ are scaled by the user–defined value so are the derivatives α_s and α_t. We see the effect of this when comparing the three conversions since 5(b) appears to be more blue compared to the other two. This is due to the smaller derivative values stored in red and green. In contrast, figure 5(d) has a lot of variance in red and green and the amount of blue is diminished.
In the following, a single point light (see section 16.1.4 in [FvDFH95]) is used for illumination of each scene. Figure 6(a) shows the results of an exact evaluation compared to Blinn's approximate evaluation, figure 6(b). Both were made with the scale set to $\frac{1}{32}$. There does not appear to be any significant visual distinction between the two. Overall the illusion appears convincing though as pointed out in section 1 the true shape of the surface is revealed at the silhouette. For a comparison, in figure 7(a) we see the same scene with the actual displacement physically applied to the surface. The interior looks much the same as before but the silhouette has changed and now looks correct. In figure 6(c), we see the results of the scale value set to $\frac{1}{16}$. Again there does not appear to be any clear distinction between the exact and the approximate (see figure 6(d)) evaluation. The effect of having doubled the scale to the previous value is evident from


(a)


(b)


(c)


(d)


(e)

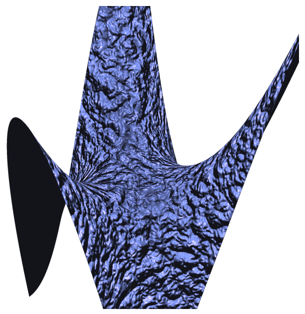

(f)

Figure 6: The bump–mapped monkey saddle is shown in 6(a) using $\frac{1}{32}$ for a user–scale and shaded using the exact perturbation of the normal. In 6(b) we see the same test using Blinn's approximate normal and visually there appears to be no difference. In figures 6(c) and 6(d) we see the same test with the user–scale $\frac{1}{16}$ and still there is no visual distinction. The surface does appear more rugged though because the scale was doubled. In figures 6(e) and 6(f) the scale was doubled again and the effect no longer appears convincing. Nevertheless, there still does not appear to be a visual difference.

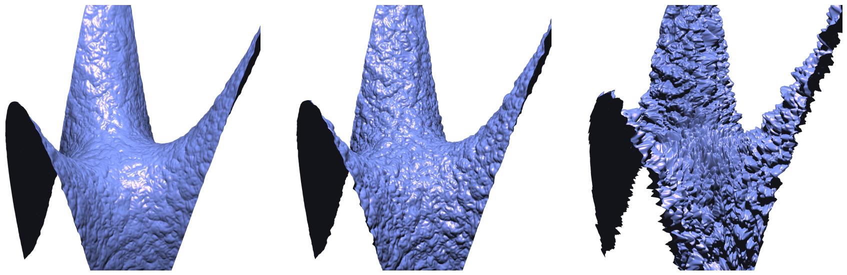


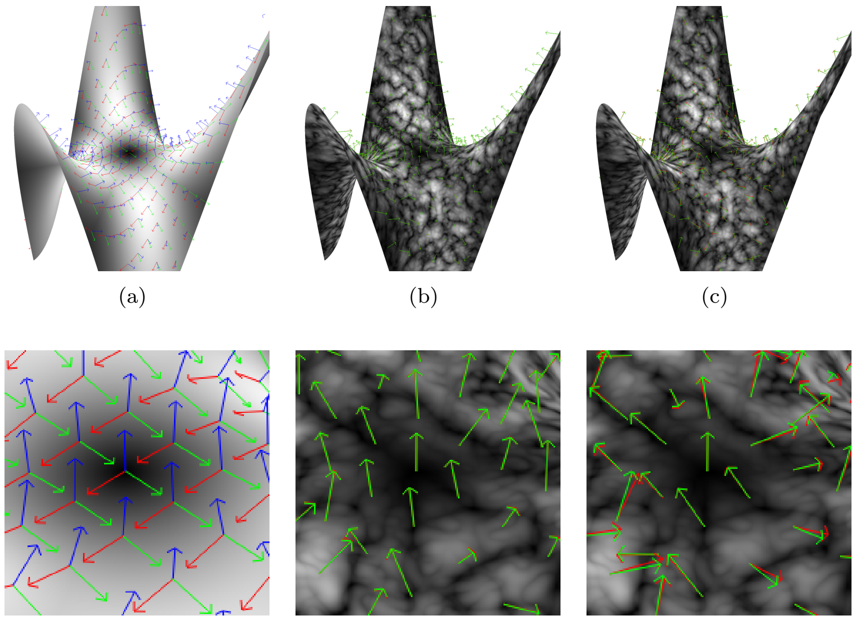
Figure 7: The displacement mapped monkey saddle is shown in figures 7(a)-7(c) with the user–scales $\frac{1}{32}$, $\frac{1}{16}$ and $\frac{1}{4}$ applied respectively. The surface appears more rugged as the scale is increased.

the rugged appearance of the surface. The result still seems convincing though the lack of surface irregularities along the silhouette has become increasingly obvious. We can observe the displaced surface in figure 7(b) which, as opposed to before, now also deviates from the interior seen in the bump mapped result.

Finally, we see the same scene in figure 6(e) where the scale this time was set to $\frac{1}{4}$ and again there does not appear to be any clear distinction between the exact and the approximate evaluation. The displacements have now become too large relative to the surface itself to obtain convincing results from bump mapping. In contrast we see the corresponding displaced surface in figure 7(c) which looks extremely different. We now see the presence of very large surface irregularities, in some cases these occlude the light which results in shadows. Since, for these three cases, we did not find any significant visual differences between the exact and the approximate evaluation of the normal, we will try a different approach where we compare the normals themselves as opposed to comparing the effect they have on the lighting.
In section 2.3, we discovered that the difference between the exact perturbed normal and the approximate one depends on the product between the height and the principal curvatures. For every pixel which represents a valid intersection, we see in figure 8(a) the value

$$\kappa_{max} = \max\left(|\kappa_1|, |\kappa_2|\right)$$

In pixels with a low intensity, a small value for κ_{max} was found, those with a bright intensity had a large maximum curvature. The smallest maximum curvature found was approximately 0.0026 and the highest was 2.4495. At the origin, we see very dark intensities. This is where the monkey saddle is known to attain a curvature of zero. Furthermore, we see on the figure, at a few selected points, the calculated tangent spaces. The concept of tangent space used in this thesis was initially explained in section 2.2. The red vector shows χ_s, the green vector shows χ_t and the blue one shows the forward facing normal which was


(a) (b) (c)

(d) (e) (f)

Figure 8: The maximum principal curvature κ_{max} is shown in 8(a). Figures 8(b) and 8(c) also show this but multiplied by the height from the bump map $\alpha(s,t)$. Furthermore, Blinn's normal is shown in green and the exact perturbed normal is shown in red. In 8(b) a user–scale of $\frac{1}{16}$ was used as opposed to figure 8(c) where $\frac{1}{4}$ was used. The corresponding close–ups are shown in figures 8(d)-8(f). Notice that deviation between the exact and the approximate perturbed normal is primarily found at brighter intensities and more so when the user–scale is increased.

clarified in section 2.4. The magnitude of the displayed vectors was set to a fixed length.

In figure 8(b), we see the product between the sampled height and κ_{max}. Furthermore, the accurate perturbed normal is shown as a red vector and Blinn's approximate normal is shown in green. In figure 8(b), the scale $\frac{1}{16}$ was used and we see very little deviation between the two normals. Figure 8(c) was done using the scale $\frac{1}{4}$ and it shows several instances of deviations. According to the analysis, differences between the two can only occur where we find nonzero intensities since the matrix M' will otherwise be the identity matrix. The brighter the intensity is, the more M' will deviate from the identity matrix. However, this alone does not result in deviation of the accurate and the approximate perturbed normal. The reason for this is that when α_s and α_t are zero, evaluation of equation (25) after normalization equals the forward facing normal $\vec{n}$ which is the original surface normal. In other words, the two vectors will be almost identical when sampling occurs where there is a local minimum or maximum of $\alpha(s,t)$. Figure 8(c) confirms this quite well since no deviations are found where the intensity is close to black, but several are found in the brighter locations where the slope of $\alpha(s,t)$ appears to be steep.

2.6.2 Torus

For an additional case, we will perform the same tests on the torus. The torus is given by two radii $a, b \in \mathbf{R}$ such that $a > b > 0$ and the level surface

$$(x^2+y^2+z^2+a^2-b^2)^2 - 4a^2(x^2+y^2) = 0$$

As in the previous case, we find the intersection by inserting the parametrization of the ray into this equation and finding the roots of the resulting polynomial which in this case is a fourth degree polynomial.
The corresponding surface parametrization is given by

$$\sigma(\theta,\varphi) = ((a+b\cdot\cos(\varphi))\cos(\theta), (a+b\cdot\cos(\varphi))\sin(\theta), b\sin(\varphi))$$

Where the angles θ and φ are given as

$$\begin{aligned}\theta &= \pi\cdot(2u-1)\\ \varphi &= \pi\cdot(2v-1)\end{aligned}$$

The first fundamental form with respect to u and v of this surface is equal to

$$\begin{aligned}E &= (2\cdot\pi)^2\cdot(a+b\cdot\cos(\varphi))^2\\ F &= 0\\ G &= (2\cdot\pi)^2\cdot b^2\end{aligned}$$

and the second fundamental form is

$$\begin{aligned}e &= -(2\cdot\pi)^2\cdot(a+b\cdot\cos(\varphi))\cdot\cos(\varphi)\\ f &= 0\\ g &= -(2\cdot\pi)^2\cdot b\end{aligned}$$

From this it follows that the principal curvatures are

$$\begin{aligned}\kappa_1 &= \frac{-\cos(\varphi)}{a + b \cdot \cos(\varphi)} \\ \kappa_2 &= \frac{-1}{b}\end{aligned}$$

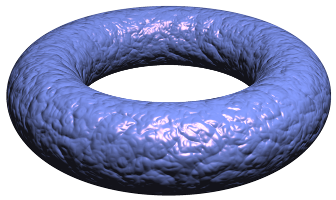
The first principal curvature reaches its maximum when $\cos(\varphi) = -1$ and the value here is $\frac{1}{a-b}$. From this we conclude that if $b < \frac{a}{2}$, we have $\kappa_{max} = |\kappa_2| = \frac{1}{b}$ which is constant.

In figure 9(a) we see the results of rendering the torus with the radii $a = 1$ and $b = 0.3$ which means $\kappa_{max} = 3\frac{1}{3}$. Once again, the diffeomorphism Φ is the identity and the bump map is similar to the one used for the monkey saddle. When looking at the subfigures in figure 9, just as for the monkey saddle, we see no clear distinction between results based on the exact evaluation of the perturbed normal and those based on the approximation and once again the illusion of bump mapping is no longer convincing for user value $\frac{1}{4}$. For completion the corresponding displaced results are supplied in figures 10(a), 10(b) and 10(c). Notice the strong resemblance between the bump mapped result and the displaced result for user value $\frac{1}{32}$ and in contrast observe the explicit differences for user value $\frac{1}{4}$.

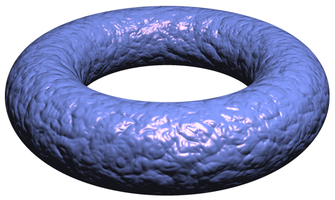
In figure 11(a), we see the product between the height and κ_{max}. However, since κ_{max} is constant the intensities seen in the image are actually the sampled heights. As before, the red vector shows the accurate perturbed normal and the green vector shows Blinn's approximation. The result was generated with user–scale $\frac{1}{16}$ and as for the monkey saddle we see only few examples of deviation between the two. Figure 11(b) on the other hand - which was made with user–scale $\frac{1}{4}$ - appears to have more instances of deviation compared to the monkey saddle (see figure 8(c)) and these are found everywhere on the surface of the torus. This is due to the consistent value of κ_{max} which for the torus of the chosen radii is higher than the maximum value of κ_{max} found on the monkey saddle. Figures 11(c) and 11(d) show the same two tests but with the radii set to $a = 1$ and $b = \frac{1}{10}$. The value $\kappa_{max} = |\kappa_2| = 10$ is three times that of the previous figure which is quite visible in the results. This time, we see several instances of deviation for user–scale $\frac{1}{16}$ and an increase in deviation for user–scale $\frac{1}{4}$ compared to figure 11(b).

Up until now, the results were done using the identity map for generation of the texture coordinates. The following tests show the effect of using an invertible affine transformation, which is a diffeomorphism, to modulate the sampling coordinate. The user–scale is set to $\frac{1}{32}$ in each case and the height map is seen in figure 12(a) with the corresponding conversion in figure 12(b). The first example in figure 13(a) shows the texture tiled twelve times along the torus in the u direction and four times along v. This is done by using the following scale matrix on (u, v).

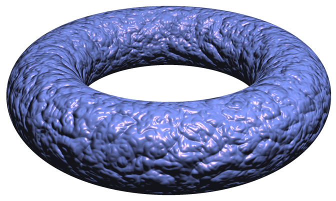
$$S = \begin{bmatrix} 12 & 0 \\ 0 & 4 \end{bmatrix}$$



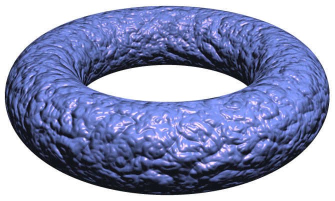
(a)



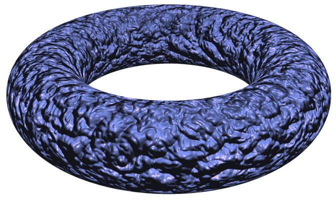
(b)



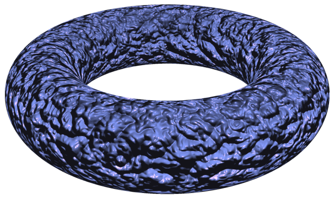
(c)



(d)



(e)



(f)

Figure 9: The bump mapped torus is shown in 9(a) using $\frac{1}{32}$ for a user–scale and shaded using the exact perturbation of the normal. In 9(b) we see the same test using Blinn's approximate normal and visually there appears to be no difference. In figures 9(c) and 9(d), we see the same test with the user–scale $\frac{1}{16}$ and still there is no visual distinction. The surface does appear more rugged though because the scale was doubled. In figures 9(e) and 9(f), the scale was doubled again and the effect no longer appears convincing. Nevertheless, there still does not appear to be a visual difference.

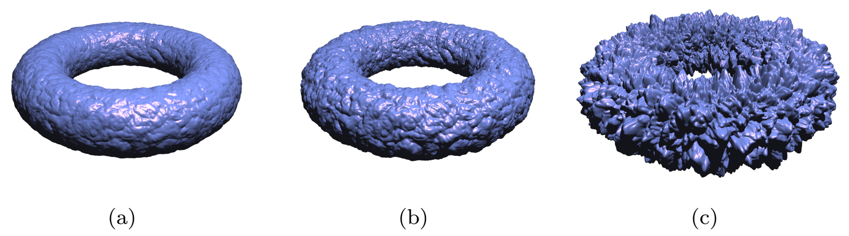


Figure 10: The displacement mapped torus is shown in 10(a)-10(c) with the user–scales $\frac{1}{32}$, $\frac{1}{16}$ and $\frac{1}{4}$ applied respectively. The surface appears more rugged as the scale is increased.

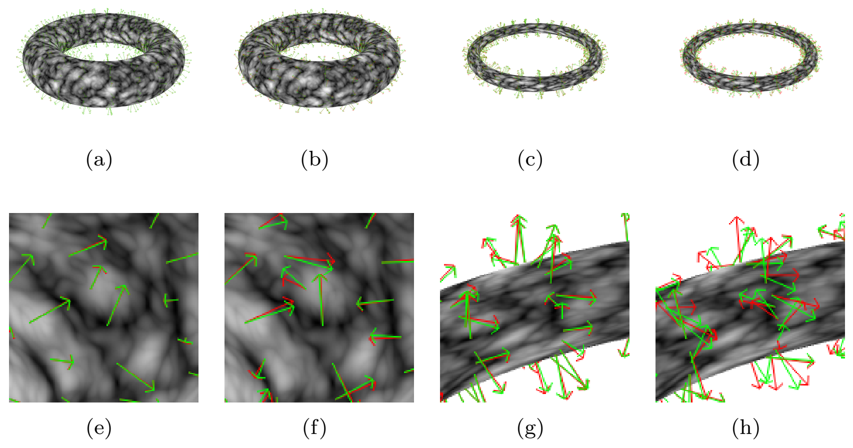


Figure 11: In these figures, Blinn's normal is shown in green and the exact perturbed normal is shown in red, and furthermore, κ_{max} is constant. Figures 11(a) and 11(b) show a comparison given user–scales $\frac{1}{16}$ and $\frac{1}{4}$ respectively. Based on the corresponding close–ups in figures 11(e) and 11(f), we see that deviation between the exact and the approximate perturbed normal is primarily found at brighter intensities and more so when the user–scale is increased. The same test is shown in figures 11(c) and 11(d) with a smaller outer radius which increases κ_{max}. The corresponding close–ups in figures 11(g) and 11(h) show that the deviation becomes more extreme.

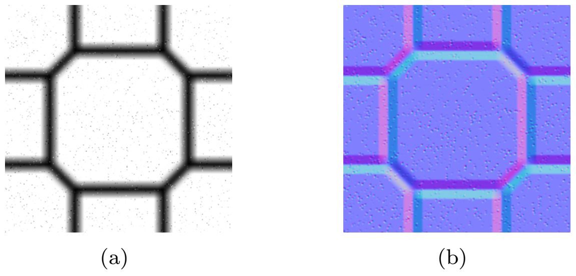


Figure 12: A bump map is shown in 12(a) and using the scale $\frac{1}{32}$ the converted bump map is shown in 12(b).

Furthermore, the figure shows the evaluated tangent space at various selected points. The red vectors which represent χ_s show the orientation of the horizontal lines of the texture and the green vectors which represent χ_t show the orientation of the vertical lines. Since F of the first fundamental form is zero, these two directions are perpendicular to each other, which appears to be the case on the figure as well. Finally, as previously mentioned, the blue vectors show the original forward facing normal.
The next example in figure 13(b) is the result of applying a shear matrix K after the scale by S.

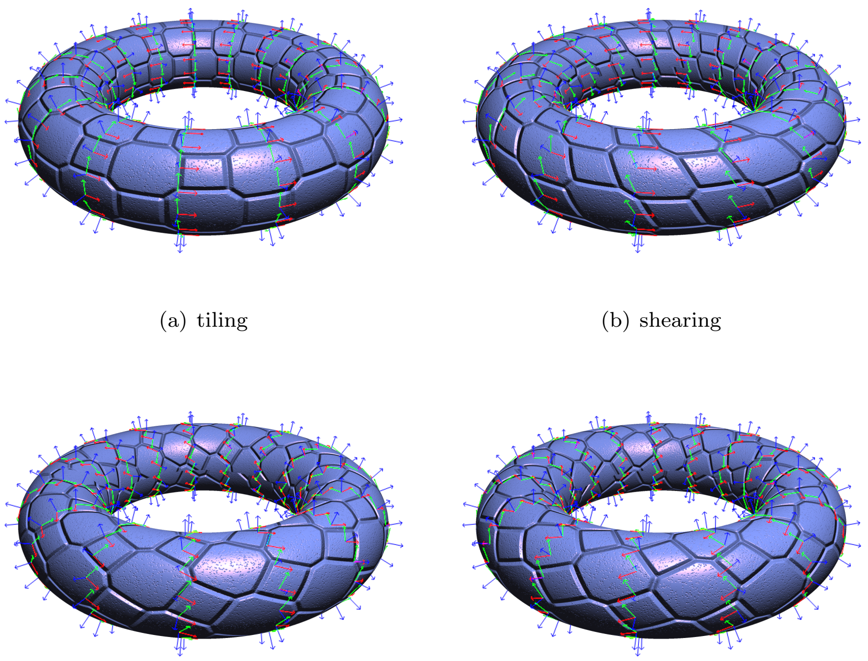
$$K = \begin{bmatrix} 1 & \frac{2}{5} \\ 0 & 1 \end{bmatrix}$$

The effect of this transformation is quite evident on the figure and also the fact that χ_s and χ_t are no longer perpendicular to each other but still follow the horizontal and vertical lines of the texture. Furthermore, when looking at the inner surface region of the torus, we now see a discontinuity which was not visible before. The discontinuity occurs where $v = 0$ meets $v = 1$ on the torus which was not visible before since the texture itself is tileable (meaning periodic). The next example in figure 13(c) is the result of replacing the shear matrix with a counterclockwise rotation R by thirty degrees.

$$R = \begin{bmatrix} \cos(\frac{\pi}{6}) & -\sin(\frac{\pi}{6}) \\ \sin(\frac{\pi}{6}) & \cos(\frac{\pi}{6}) \end{bmatrix}$$

As we see on the figure, the map is transformed clockwise around the front facing normal which is correct since to transform (s, t) to (u, v) the inverse is applied and the inverse of R is the corresponding clockwise rotation. For the last test, we try an orientation reversing parametrization. We do this by rewriting the matrix S such that a negative scale in the u direction is applied and subsequently the same rotation R as before is applied.

$$S = \begin{bmatrix} -12 & 0 \\ 0 & 4 \end{bmatrix}$$


(a) tiling (b) shearing

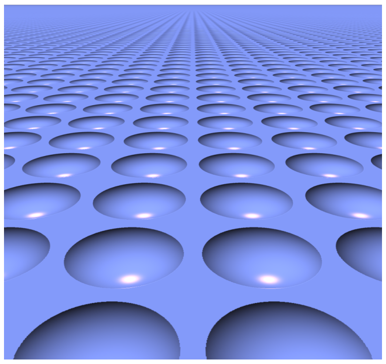
(c) rotation (d) reversal

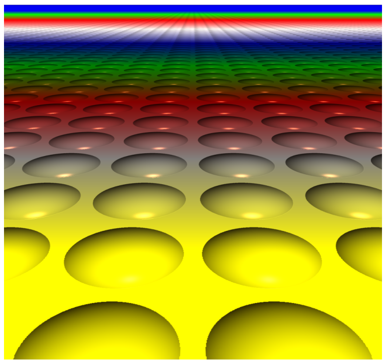
Figure 13: In 13(a), we see a torus with a tiled bump map applied. Reparametrization is performed in 13(b) by shearing the domain. In 13(c), rotation is applied on the domain and finally a negative scale in 13(d) which results in an orientation reversing reparametrization.

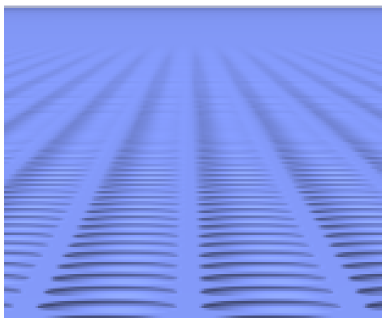
The result is seen in figure 13(d) and as we see the map on the torus including the assigned tangent spaces are reversed. In particular, the tangent spaces are now all left–hand coordinate systems as opposed to the previous tests where they were right–hand coordinate systems. Furthermore, the blue vectors sampled from the assigned tangent spaces are still forward facing as they should be according to the analysis of section 2.4. The tiles on the torus appear to bulge which agrees with the given height map and the previous test results.

2.6.3 Filtering

In figure 14(a), we see a simple bump–mapped infinite plane and in figure 14(b), we see the same plane but color-coded to identify transitions through mip map levels. A close–up of figure 14(a) is shown in figure 14(c) and the result is clearly more desirable compared to the same close–up shown in figure 14(d) which has mip mapping disabled. This supports Blinn's choice to enable traditional texture filtering as opposed to no filtering at all. The result seen with filtering enabled, however, is not ideal either but this is caused by the perspective projection squeezing the circular bumps faster in the vertical direction compared to the horizontal one which causes a deviation in ideal choice of mip map level between the two.


(a)


(b)


(c)

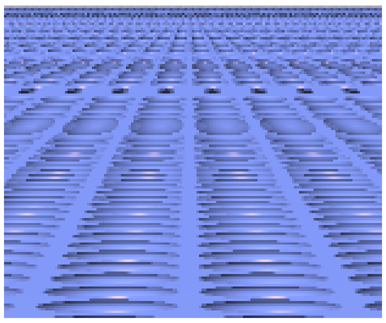

(d)

Figure 14: In 14(a), a plane is shown with a tiled bump map applied to it. Transition through mip map levels is shown in color-codes in 14(b) where the color yellow corresponds to the initial full resolution mip map level. A close–up is shown in 14(d) with filtering disabled. The same close–up is shown in 14(c) with filtering enabled and though it is not perfect, it is clearly superior to filtering disabled.

3 Triangular meshes

In this section the problem of bump/normal mapping a triangular mesh is treated. For the case of a single triangle, it is shown in section 3.1 how assigned texture coordinates can be understood as a reparametrization. Furthermore, the first– and second–order derivatives are derived. When a triangular mesh is used to approximate a model which is either entirely smooth or piecewise smooth, then tangent space evaluation is a nontrivial problem. An introduction to the complexity of tangent space evaluation at vertex level is given in section 3.2 along with test results from two existing commercial products. In section 3.3 a suggestion for vertex level tangent space evaluation is made, initially based on intuition, and the analysis of section 2.4 on reparametrized surfaces. Subsequently, a series of mathematical observations are made to confirm the logic behind the approach.

A triangular mesh, used to represent a smooth surface, is by definition an approximation. Assuming the true parametrization of the surface is unknown then there is no correct way to determine tangent space at a given point on the mesh. This problem is identified and discussed in section 3.4. An additional very important observation of this section is that tangent space evaluation at such a point must be determined the same way in the tool used to sample normal maps as in the shader. Otherwise they are technically incompatible. This is a result of the observation made in section 2.2, i.e., that the exact inverse must be used in the shader to obtain the original sampled normal. Finally, in section 3.6 results are given.

3.1 Evaluation of derivatives

As mentioned in the beginning of section 2.4 artists are permitted to generate texture coordinates as diffeomorphisms of (u, v). In practice the most common case for such a map is of the following form

$$\begin{bmatrix} s \\ t \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \cdot \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} \tag{37}$$

where $k_1, k_2 \in \mathbf{R^2}$ is constant and the matrix $[m_{ij}]$ is nonsingular. This map is smooth and has a smooth inverse.

For any parametrized surface, for instance a Bezier patch, this map (37) might typically be the composition of rotations, scalings and translations specified by an artist. We are of course not limited to these, in fact we are not even limited to the form (37). It is only pointed out here because it is the most common form.

Though it may initially not be obvious, a classic example of such a case is a simple triangle defined counterclockwise by three distinct vertices $p_1, p_2, p_3 \in \mathbf{R^3}$ and three distinct corresponding mapping coordinates $t_1, t_2, t_3 \in \mathbf{R^2}$. One valid parametrization for a plane through this triangle is

$$\begin{aligned}\sigma(u,v) &= (p_2 - p_1)\cdot u + (p_3 - p_1)\cdot v + p_1 \\ &= p_1 \cdot (1-u-v) + p_2 \cdot u + p_3 \cdot v \qquad (38)\end{aligned}$$

and given this parametrization a point is exactly inside the boundary of the triangle when $u, v \geq 0$ and $u+v \leq 1$. The coordinates t_1, t_2, t_3 define a triangular area in $\mathbf{R^2}$ and points inside the triangle in $\mathbf{R^3}$ are mapped to this area by a linear function $f : \mathbf{R^3} \to \mathbf{R^2}$ such that $t_1 = f(p_1)$, $t_2 = f(p_2)$ and $t_3 = f(p_3)$. The 2×3 matrix C that fits these criteria is unique and it is given by

$$\begin{aligned} C \cdot \left[\begin{array}{c|c|c} p_1 & p_2 & p_3 \end{array}\right] &= \left[\begin{array}{c|c|c} t_1 & t_2 & t_3 \end{array}\right] \\ &\Leftrightarrow \\ C &= \left[\begin{array}{c|c|c} t_1 & t_2 & t_3 \end{array}\right] \cdot \left[\begin{array}{c|c|c} p_1 & p_2 & p_3 \end{array}\right]^{-1}\end{aligned}$$

Note that f, as a linear function, only exists when the plane given in equation (38) is not a hyperplane of $\mathbf{R^3}$ which implies $\operatorname{span}\{p_1, p_2, p_3\} = \mathbf{R^3}$, thus for any point p on σ we obtain the corresponding mapping coordinate by $(s,t) = C \cdot p$. We are now ready to identify the diffeomorphism as

$$\begin{aligned}(s,t) &= \Phi^{-1}(u,v) \\ &= C \cdot \sigma(u,v) \\ &= C \cdot (p_1 \cdot (1-u-v) + p_2 \cdot u + p_3 \cdot v) \\ &= t_1 \cdot (1-u-v) + t_2 \cdot u + t_3 \cdot v \\ &= (t_2 - t_1)\cdot u + (t_3 - t_1)\cdot v + t_1 \qquad (39)\end{aligned}$$

The result in equation (39) is very similar in form to equation (38) and furthermore does not depend on whether or not f is linear. Next, by defining the matrix

$$T = \left[\begin{array}{c|c} t_2 - t_1 & t_3 - t_1 \end{array}\right]$$

it follows that the expression can be rewritten to the previously (eq. (37)) mentioned form

$$\begin{bmatrix} s \\ t \end{bmatrix} = T \cdot \begin{bmatrix} u \\ v \end{bmatrix} + t_1$$

and by inverting this function we obtain the equation

$$\begin{aligned}\begin{bmatrix} u \\ v \end{bmatrix} &= \Phi(s,t) \\ &= T^{-1} \cdot \left(\begin{bmatrix} s \\ t \end{bmatrix} - t_1 \right) \qquad (40)\end{aligned}$$

and from this we can evaluate the first–order derivatives of Φ

$$\begin{aligned} J(\Phi) &= T^{-1} \qquad (41) \\ &= \frac{\begin{bmatrix} (t_3 - t_1)_y & -(t_3 - t_1)_x \\ -(t_2 - t_1)_y & (t_2 - t_1)_x \end{bmatrix}}{(t_2 - t_1)_x \cdot (t_3 - t_1)_y - (t_2 - t_1)_y \cdot (t_3 - t_1)_x}\end{aligned}$$

In this case, the notation t_x and t_y is understood as first and second component of t and not the first–order derivative.

And now, given the chain rule and the reparametrization $\chi(s,t) = \sigma(\Phi(s,t))$ the first–order derivatives and the forward facing normal follow as

$$\begin{aligned}
\chi_s &= \frac{d\Phi_1}{ds} \cdot \sigma_u(\Phi(s,t)) + \frac{d\Phi_2}{ds} \cdot \sigma_v(\Phi(s,t)) \\
&= \frac{(t_3 - t_1)_y \cdot (p_2 - p_1) - (t_2 - t_1)_y \cdot (p_3 - p_1)}{(t_2 - t_1)_x \cdot (t_3 - t_1)_y - (t_2 - t_1)_y \cdot (t_3 - t_1)_x} \qquad (42) \\
\chi_t &= \frac{d\Phi_1}{dt} \cdot \sigma_u(\Phi(s,t)) + \frac{d\Phi_2}{dt} \cdot \sigma_v(\Phi(s,t)) \\
&= \frac{-(t_3 - t_1)_x \cdot (p_2 - p_1) + (t_2 - t_1)_x \cdot (p_3 - p_1)}{(t_2 - t_1)_x \cdot (t_3 - t_1)_y - (t_2 - t_1)_y \cdot (t_3 - t_1)_x} \qquad (43) \\
\vec{n} &= \frac{\sigma_u(\Phi(s,t)) \times \sigma_v(\Phi(s,t))}{\|\sigma_u(\Phi(s,t)) \times \sigma_v(\Phi(s,t))\|} \\
&= \frac{(p_2 - p_1) \times (p_3 - p_1)}{\|(p_2 - p_1) \times (p_3 - p_1)\|} \qquad (44)
\end{aligned}$$

Since the second–order derivatives are zero, the weingarten matrix will have zero in all entries which results in M' being the identity matrix. The matrix N' on the other hand, as given in section 2.4, simply contains χ_s, χ_t and $\vec{n}$ in the columns.

An interesting observation is that equations (42), (43) and (44) are all independent of the sequence in which the vertices of the triangle are given. Since the vertices must be defined counterclockwise relative to the intended visible side, this leaves us with the existing cyclic permutations only as valid options: The initial assumed to be valid order (p_1, p_2, p_3) but also (p_3, p_1, p_2) and (p_2, p_3, p_1) and of course the texture coordinates are given accordingly. Given these options, equation (44) is independent of the sequence since it is the surface normal. We can observe that equations (42) and (43) are also order independent by rearranging the terms in both the nominator and the denominator.

$$\begin{aligned}
\chi_s &= \frac{(t_2 - t_3)_y \cdot p_1 + (t_3 - t_1)_y \cdot p_2 + (t_1 - t_2)_y \cdot p_3}{(t_{1x} \cdot t_{2y} - t_{1y} \cdot t_{2x}) + (t_{2x} \cdot t_{3y} - t_{2y} \cdot t_{3x}) + (t_{3x} \cdot t_{1y} - t_{3y} \cdot t_{1x})} \\
&= \frac{\sum_{i=0}^{2} (t_{2\otimes i} - t_{3\otimes i})_y \cdot p_{1\otimes i}}{\sum_{i=0}^{2} \det(\left[\begin{array}{c|c} t_{1\otimes i} & t_{2\otimes i} \end{array}\right])} \qquad (45) \\
\chi_t &= \frac{(t_3 - t_2)_x \cdot p_1 + (t_1 - t_3)_x \cdot p_2 + (t_2 - t_1)_x \cdot p_3}{(t_{1x} \cdot t_{2y} - t_{1y} \cdot t_{2x}) + (t_{2x} \cdot t_{3y} - t_{2y} \cdot t_{3x}) + (t_{3x} \cdot t_{1y} - t_{3y} \cdot t_{1x})} \\
&= \frac{\sum_{i=0}^{2} (t_{3\otimes i} - t_{2\otimes i})_x \cdot p_{1\otimes i}}{\sum_{i=0}^{2} \det(\left[\begin{array}{c|c} t_{1\otimes i} & t_{2\otimes i} \end{array}\right])} \qquad (46)
\end{aligned}$$

In fact, given equations (45) and (46), the first–order derivatives χ_s and χ_t do not even depend on the vertices being defined counterclockwise. In contrast σ_u and σ_v do clearly depend on the input ordering. The operator $\otimes$ was used here to represent a periodic version of the standard plus operator such that $4 \mapsto 1$, $5 \mapsto 2$.

In summary, it has been shown in this section that texture coordinates assigned to a triangle is equivalent to a surface reparametrization. The first–order derivatives of the reparametrized surface have been derived and are given by equations (42) and (43).

3.2 Shading continuity across patches

Bezier–patches can represent surfaces of arbitrary topological type by partitioning the model into a collection of individual Bezier–patches. Commercial modeling tools provide functionality which allows artists to stitch patches together and maintain various levels of continuity G_0, C_0 and C_1 between adjacent patches.

Each patch is a surface parametrization of $(u, v) \in [0; 1] \times [0; 1]$ and each patch is supplied with a diffeomorphism $\Phi^{-1}(u, v)$ to generate texture coordinates from (u, v). If these maps are chosen so they provide a smooth transition in (s, t) between adjacent patches, then continuity levels of the reparametrized surface $\sigma(\Phi(s, t))$ are inherited from the original. This follows from the chain rule. A good example of how such maps may be chosen can be found by once

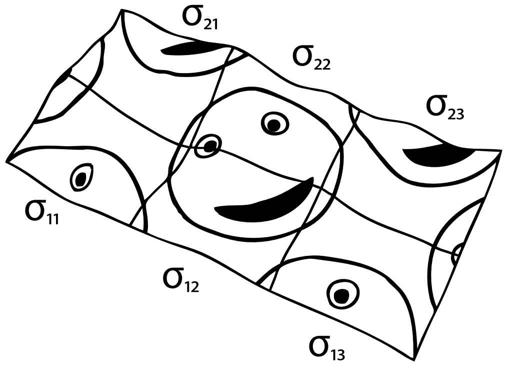


Figure 15: This figure shows 2×3 patches stitched together with a coherent mapping of the bump map shown in figure 3(c).

again using equation (37). If for some $n, m \in \mathbf{N}$ we imagine a collection of patches double indexed σ_{ij} by the order in which they are stitched together (see figure 15) where $i \in \{0, 1, ..., m-1\}$ and $j \in \{0, 1, ..., n-1\}$, then we simply replace (u, v) with $(u+i, v+j)$ before use in equation (37). This way the local parametric coordinate is mapped to a larger global grid followed by a shared linear map and a translation which provides an entirely smooth transition in

(s, t) between adjacent patches.

$$\begin{bmatrix} s \\ t \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \cdot \begin{bmatrix} u+i \\ v+j \end{bmatrix} + \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} \tag{47}$$

The preservation of continuity levels after reparametrization of σ is significant because it is necessary in order for the perturbed normal to maintain a continuous transition between adjacent patches.

Blinn's paper unfortunately does not cover triangular meshes. If we interpret a mesh as the exact shape formed by the collection of triangles from which it is constructed, then the face normal of each triangle is used in the illumination process and any two adjacent triangles which are not coplanar form a *hard edge* between them. Given this interpretation, we know the piecewise parametrizations from which the mesh is formed and subsequently we can apply bump mapping to the mesh. If on the other hand we allow averaged normals at vertex level, we face several problems to solve. Such configurations are used to approximate curved surfaces usually of an unknown parametrization.
The amount of existing documentation on bump mapping applied to a triangular mesh is surprizingly sparse, nevertheless, the problem has been identified before. It was suggested by Nelson L. Max in [Max88] that one could simply average at each vertex the first–order derivatives of the surrounding triangles. A different path however is chosen by Nelson: Instead of averaging, interpolation is done of the vertex normal only and subsequently as rasterization takes place, the first–order derivatives of the triangle parametrization are used. These were shown in section 3.1 to be constant. This idea was recently also adopted in [Sch06] which suggests a method specifically intended for 3D accelerated rendering. The method evaluates the first–order derivatives of the triangle in the fragment shader using specialized instructions, known as *ddx* and *ddy.* Either way, the concept is a very bad idea because it leads to discontinuities in tangent spaces between adjacent triangles. This causes the surface normal, after perturbation, to have a discontinuous transition across edges. There is also a problem in regards to sampled normal maps because as the sampled normals are transformed into tangent space, as explained in section 2.2, they are stored as a texture with discontinuities between adjacent triangles. These discontinuities will not be taken into account when texture filtering takes place and thus averaging of normals stored in distinctly different spaces will occur.

Since preservation of continuity of normals is an issue, a solution which maintains a continuous transition of tangent space is needed. One way to achieve this could be, as suggested in [Max88], to simply average the first–order derivatives at each vertex. This principle is used by the middleware tool *Melody* by Nvidia which is used for sampling of normal maps as explained in section 2.2. However, the solution is naive because we cannot simply average the contribution of any two triangles which share a vertex and a normal since some of the triangle level first–order derivatives surrounding a vertex can point in significantly different directions depending on the assigned texture coordinates.

A 3D model of a character-head, used in the game "Kane & Lynch: Dead Men", has been provided by IO-Interactive for testing. Two versions are given, one in low resolution with texture coordinates and one in high resolution with no texture coordinates. Both models are mirrored such that the left half is a copy of the right half with a negative scale by one applied along the X-axis. Subsequently, two indices are flipped in every copied triangle to maintain a counterclockwise ordering. The texture coordinates are directly copied with no modifications applied.

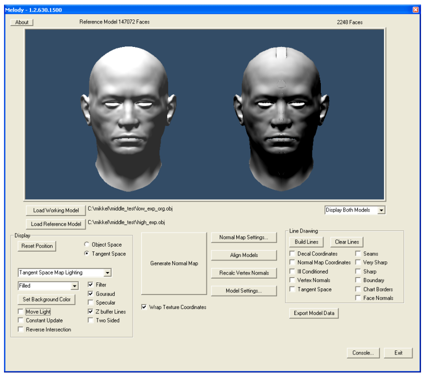
3.2.1 Shading continuity in Melody

The normal mapped result output by Melody is shown in figure 16(a) with the high resolution model on the left side and the low resolution model on the right side. If we zoom in on the low resolution model generated by Melody, we see errors down the middle of the face, see figure 16(b). The normal map generated by Melody is shown in figure 16(d) and there are very clear signs of errors along the part of the border which corresponds to the middle of the head. Figure 16(c) shows the tangent spaces assigned by Melody. The tangent spaces assigned to each half are clearly oriented in separate directions which is the result of the model being mirrored. Only one space is assigned to every vertex along the middle which causes extreme variations in the assigned tangent spaces. To summarize: Discontinuity in tangent space between all triangles is not a good solution but the results made with Melody indicate that forcing one tangent space at every vertex does not provide a general solution either. Some form of occasional discontinuity is required. It might seem tempting to simply solve the problem using a vector quantization algorithm. However, such an algorithm generally requires the user to provide the size of the codebook in the first step at which point it is unknown to us but more importantly such algorithms are order dependent. The significance of this is due to the dependency between sampled normal maps and σ as explained in section 2.2. To preserve continuity of normals, tangent spaces assigned to the copied half must be the exact mirrored tangent spaces of the original half.

3.2.2 Shading continuity using Crytek source code

As a case study, tangent space on a mirrored triangular mesh is evaluated using source code provided by the company Crytek. The source code is also used in their tool Polybump, for generation of sampled normal maps. It has been given to me upon request and is generally not available, not even to those who purchase Polybump.

Looking at figure 17(f) shows that, unlike Nelson's approach, tangent spaces are definitely shared at most vertices. Unlike Melody, the Crytek code has assigned two tangent spaces for every shared vertex between the two halfs, so it is clear that Crytek has also identified the requirement for occasional discontinuity in tangent space assignment at vertex level. Nevertheless, it appears they have not identified the need for an order independent algorithm. A close–up is shown in figure 17(a) where the triangles in question have been marked red and additionally the three tangent spaces assigned to each of the two triangles have



Melody - 1.2.630.1500

About | Reference Model 147072 Faces | 2248 Faces

Load Working Model: C:\mikkel\middle_test\low_exp_org.obj

Load Reference Model: C:\mikkel\middle_test\high_exp.obj

Display Both Models

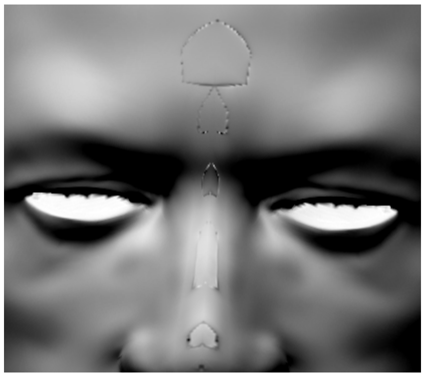
Display: Reset Position; ( ) Object Space; (•) Tangent Space; Tangent Space Map Lighting; Filled; [✓] Filter; [✓] Gouraud; [ ] Specular; [✓] Z buffer Lines; [ ] Two Sided; Set Background Color; [ ] Move Light; [ ] Constant Update; [ ] Reverse Intersection

Generate Normal Map; Normal Map Settings...; Align Models; Recalc Vertex Normals; Model Settings...; [✓] Wrap Texture Coordinates

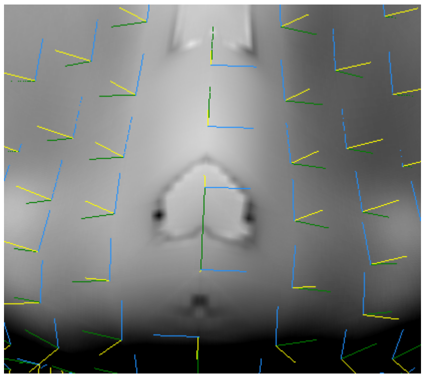
Line Drawing: Build Lines; Clear Lines; [ ] Decal Coordinates; [ ] Normal Map Coordinates; [ ] Ill Conditioned; [ ] Vertex Normals; [ ] Tangent Space; [ ] Seams; [ ] Very Sharp; [ ] Sharp; [ ] Boundary; [ ] Chart Borders; [ ] Face Normals

Export Model Data; Console...; Exit

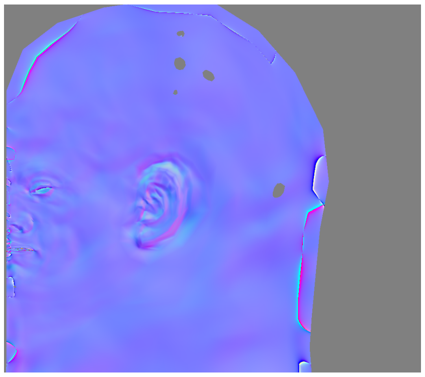
(a)



(b)

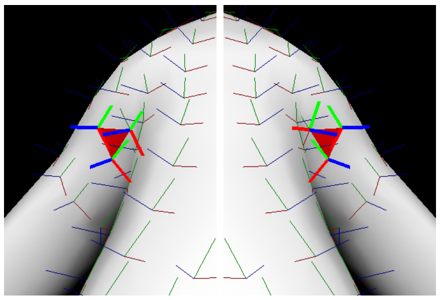


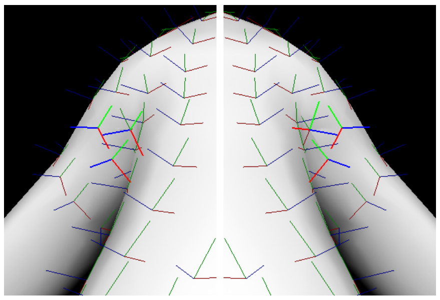
(c)

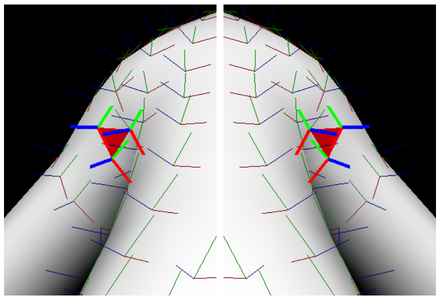


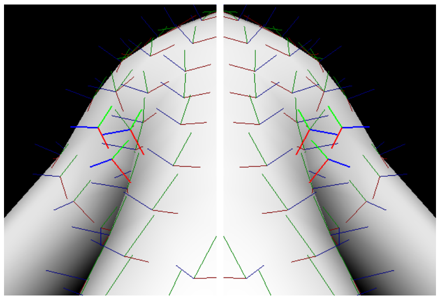
(d)

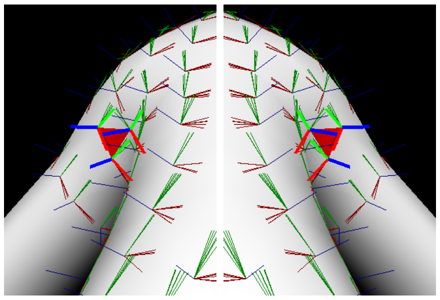
Figure 16: The normal map sampling tool Melody by the company Nvidia is shown in 16(a). On the left side a high resolution head is seen and on the right side the corresponding normal mapped low resolution head is shown. A close–up of the normal mapped head is shown in 16(b) and reveals some very noticeable errors. In 16(c) the vertex level tangent spaces assigned by Melody is shown. Only one tangent space is assigned per vertex down the middle of the head. The normal map generated by Melody is shown in 16(d) and there are several problem locations along the parts which correspond to the middle of the head.


(a)


(b)


(c)


(d)


(e)

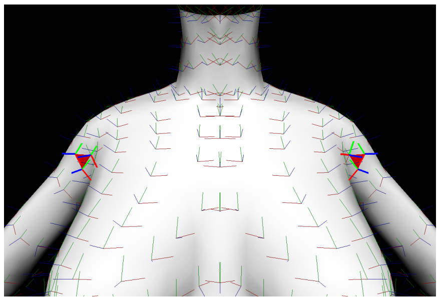
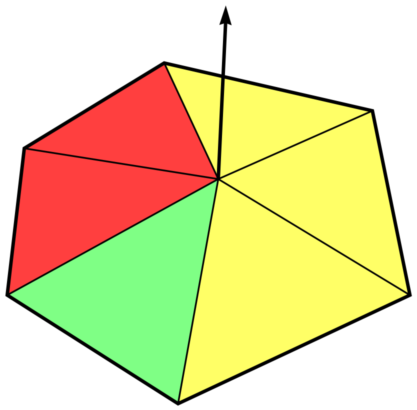

(f)

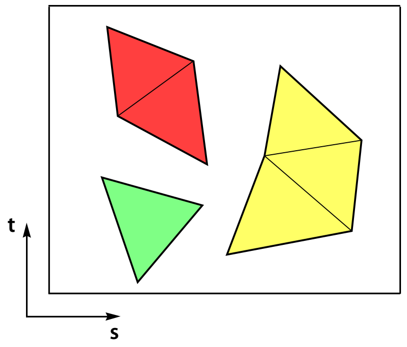
Figure 17: A mirrored female body is shown in 17(f) where vertex level tangent spaces were generated with source code provided by the company Crytek. An instance of a location where the source code failed to provide matched results between, the left side and the right side, is marked with red triangles. The three tangent spaces assigned to each of the two triangles have been drawn using thicker lines and brighter colors. A close–up is shown in 17(a) and removal of the red triangles reveals a shading seam on the right side shown in 17(b). The corresponding results using the algorithm of this thesis is shown in figures 17(c) and 17(d). Setting the angular threshold to a strict value such as 2° still provides a perfectly mirrored matching triangle pair as shown in 17(e).

been drawn using thicker lines and with brighter colors. The figure reveals that though the selection of tangent spaces available on each side are quite similar, the three spaces assigned to the triangle on the right side are not the correct mirrored results of the spaces assigned to the triangle on the left side. Since the normal map is sampled entirely based on the left side, a shading seam appears on the right side due to an inconsistency in tangent space evaluation (see figure 17(b)). The problem will be analyzed further in section 3.3.

3.3 Averaging tangent spaces

In section 3.1, it was explained how a triangle can be given by the surface parametrization in equation (38) defined on the domain $U = \{u, v \in \mathbf{R} | u, v \geq 0, u + v \leq 1\}$. Additionally, the reparametrization Φ in equation (40) is given such that the domain is the triangular area defined by the assigned texture coordinates. Intuitively, it makes sense to consider two triangles as part of an approximation for a single surface parametrization of (s, t) if they are adjacent in object space as well as the domain. In figure 18(a) we see a vertex p with a vertex normal $\vec{n}$ assigned to it and a collection of surrounding triangles. The corresponding domains are shown in figure 18(b). Triangles which are sequentially adjacent (in the domain as well) are grouped and shown in color. Each


(a)



(b)

Figure 18: Triangles surrounding an arbitrary vertex of a mesh are shown in 18(a). Furthermore, the triangles have been color–coded to identify triangles which are adjacent in the domain where the domains are defined by the texture coordinates. These domains are shown in 18(b).

group corresponds to a surface parametrization $\chi(s, t)$, and so conceptually a tangent space, dependent on the location of p on $\chi(s, t)$, should be assigned to

p for each surrounding group. Since tangent space on a single triangle is constant, this will be approximated by averaging the first–order derivatives with respect to s and t separately. The motivation behind averaged tangent spaces is preservation of a continuous transition of the surface normal, used for shading, between adjacent triangles. For this to be of any relevance, the two triangles must also share two normals (*soft edge*) for there to be an initial continuity between them to preserve.

3.3.1 Connectivity rules

In section 2.4 it was said that a reparametrization is not permissible if $\det[J(\Phi)] = 0$ anywhere on the given domain. Specifically, such a case would cause the surface normal to vanish, meaning attain a magnitude of zero. We also know from section 2.4 that the orientation is reversed when $\det(J(\Phi)) < 0$ and preserved when $\det(J(\Phi)) > 0$. So to consider adjacent triangles as belonging to the same parametrization, they must both be orientation preserving or both orientation reversing. Equation (41) leads to the following

$$\begin{aligned}\det(J(\Phi)) &= \det(T^{-1}) \\ &= \frac{1}{\det(T)}\end{aligned}$$

which tells us, the orientation is reversed exactly when $\det(T) < 0$ and it follows that

$$\det(T) = (t_2 - t_1)_x \cdot (t_3 - t_1)_y - (t_2 - t_1)_y \cdot (t_3 - t_1)_x \tag{48}$$

is less than zero exactly when t_1, t_2 and t_3 are defined clockwise. It was mentioned in section 3.2 that, when mirroring, two indices of every copied triangle are flipped to maintain a counterclockwise ordering of the triangle vertices. However, since the operation also reorders the texture coordinates, the sign of $\det(T)$ will be flipped. If the original triangle is orientation preserving, then the copy will be orientation reversing, and similarly if the original triangle is orientation reversing, then the copy will be orientation preserving. Thus it follows that triangles on separate sides of the middle must not share tangent space as they do in figure 16(c).

To summarize: Two triangles can be considered adjacent and part of the same surface parametrization when the following four rules are obeyed:

1. two vertices are shared.
2. vertex normals at the two vertices are shared.
3. texture coordinates at the two vertices are shared.
4. the triangles must have the same sign of $\det(T)$.

So, henceforth, a group as shown in figure 18(a), is understood as triangles surrounding p and sequentially connected edge by edge such that the above four rules are obeyed. By this definition, the groups surrounding p are disjoint. Additionally, all triangles in a group, will at p, be assigned the accumulated tangent space associated with the group. Finally, multiple tangent spaces associated with p, i.e., multiple surrounding groups, will henceforth be known as *tangent space splits*. As an example there is a tangent space split in figure 17(b) near the region of the shading seam. To evaluate the tangent space of a group, we proceed as follows

1. Set for the current group the initial value of the two accumulated first–order derivatives and their magnitudes to zero.
2. For each triangle in the group
   (a) Evaluate first–order derivatives of the current triangle.
   (b) Accumulate the magnitudes.
   (c) Project the first–order derivatives χ_s and χ_t into the tangent plane specified by the vertex normal $\vec{n}$ at p.
   (d) Normalize the projected χ_s and χ_t and subsequently accumulate them.
3. Divide the two accumulated magnitudes by the amount of triangles in the group.
4. Normalize the two accumulated first–order derivatives.

Since triangles which at an endpoint share a vertex, a vertex normal and a texture coordinate do not necessarily belong to the same group, it should be clear that tangent space splits may occur. Note that the algorithm given here is independent of the order in which the triangles are given. Evaluation using this algorithm as opposed to the source code by Crytek leads to correct results, see figure 17(c). Subsequently, there is no resulting shading seam, see figure 17(d).

It may initially not be clear how the four rules also guarantee adjacency in the domain. To explain this, let two adjacent triangles be given, and let the first triangle be defined by the vertices p_1, p_2 and p_3 and the texture coordinates t_1, t_2 and t_3. Furthermore, the second triangle is defined by the vertices q_1, q_2 and q_3 and the texture coordinates r_1, r_2 and r_3. By definition, the vertices of each triangle are defined counterclockwise. Let the triangles share an edge at $p_2 = q_3$ and $p_3 = q_2$ as shown in figure 19(a). This confirms the first rule. Assume that vertex normals at these two vertices are also shared, this confirms the second rule. The third rule dictates that the texture coordinates must be shared at the shared vertices so we know $t_2 = r_3$ and $t_3 = r_2$. This is obeyed in both configurations, in the domain, shown in figures 19(b) and 19(c). However, in figure 19(b) the domains are actually overlapping and cannot be considered adjacent. This is because the fourth rule is being violated. The texture coordinates r_1, r_2 and r_3 are defined clockwise which as previously mentioned, given equation (48), results in a negative sign. The texture coordinates t_1, t_2 and t_3 on the other hand are defined counterclockwise which results in a positive sign.

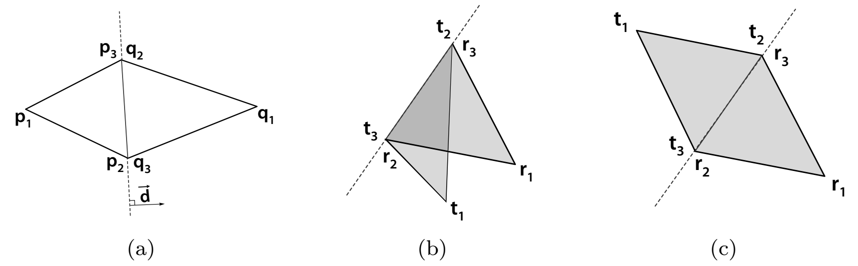


Figure 19: Two triangles which are adjacent in object space are shown in 19(a). In 19(b) and 19(c) the assigned texture coordinates are shown which define the respective domains. The fourth rule is violated in 19(b) but not in 19(c).

The configuration in figure 19(c), on the other hand, obeys the fourth rule as well. The fourth rule is there to guarantee that the texture coordinates t_1 and r_1 are assigned to different sides of the dashed line through the shared segment t_2t_3, thus adjacency of the domains is guaranteed by the fourth rule. Note that mirroring is actually a special case of a violation of the fourth rule.

3.3.2 Coherency of first–order derivatives

It is important that the alignment of the first–order derivatives within a group is approximately uniform since they are to be averaged into two representative directions, i.e., one for each of the two parameters s and t. The following is an analysis of, to what extent, can we expect first–order derivatives of adjacent triangles within a group to be coherent. The projection of χ_s into the tangent plane in step 2c is performed using the traditional equation

$$\chi_s' = \chi_s - \vec{n} \cdot (\chi_s \bullet \vec{n})$$

and similar for χ_t. This map is in fact linear and the corresponding matrix is defined as

$$G = \begin{bmatrix} 1 - \vec{n}_x \cdot \vec{n}_x & -\vec{n}_x \cdot \vec{n}_y & -\vec{n}_x \cdot \vec{n}_z \\ -\vec{n}_y \cdot \vec{n}_x & 1 - \vec{n}_y \cdot \vec{n}_y & -\vec{n}_y \cdot \vec{n}_z \\ -\vec{n}_z \cdot \vec{n}_x & -\vec{n}_z \cdot \vec{n}_y & 1 - \vec{n}_z \cdot \vec{n}_z \end{bmatrix}$$

A vector is projected by simply transforming it by G. The projection of an arbitrary vertex v into the tangent plane is done by

$$g(v) = p + G \cdot (v - p)$$

As it turns out, step 2c actually corresponds to projecting each triangle into the tangent plane specified by p and $\vec{n}$ before evaluation of tangent space. We can see this by simply applying this projection to equation (42).

$$\begin{aligned}
G\cdot\chi_s &= \frac{G\cdot\Big((t_3-t_1)_y\cdot(p_2-p_1)-(t_2-t_1)_y\cdot(p_3-p_1)\Big)}{(t_2-t_1)_x\cdot(t_3-t_1)_y-(t_2-t_1)_y\cdot(t_3-t_1)_x}\\
&= \frac{(t_3-t_1)_y\cdot G\cdot(p_2-p_1)-(t_2-t_1)_y\cdot G\cdot(p_3-p_1)}{(t_2-t_1)_x\cdot(t_3-t_1)_y-(t_2-t_1)_y\cdot(t_3-t_1)_x}\\
&= \frac{(t_3-t_1)_y\cdot(g(p_2)-g(p_1))-(t_2-t_1)_y\cdot(g(p_3)-g(p_1))}{(t_2-t_1)_x\cdot(t_3-t_1)_y-(t_2-t_1)_y\cdot(t_3-t_1)_x}
\end{aligned}$$

The same principle applies to (43). Next, let two times the area of the triangle defined by p_1, p_2 and p_3 be known as $\mathcal{A} > 0$. This area can be computed by crossing any two distinct edge vectors on the triangle such as $p_2 - p_1$ and $p_3 - p_1$ and then computing the length of this cross product. From this, and since triangles are defined counterclockwise, we may evaluate $\mathcal{A}$ given either of the two following equations

$$\begin{aligned}
\mathcal{A} &= \det\big[\ p_3-p_2 \ \big|\ p_1-p_2 \ \big|\ \vec{n}\ \big]\\
&= (p_1-p_2)\bullet\vec{n}\times(p_3-p_2)\\
&= (p_2-p_1)\bullet(p_3-p_2)\times\vec{n} \qquad (49)\\
\mathcal{A} &= \det\big[\ p_1-p_3 \ \big|\ p_2-p_3 \ \big|\ \vec{n}\ \big]\\
&= (p_1-p_3)\bullet(p_2-p_3)\times\vec{n}\\
&= (p_3-p_1)\bullet(p_3-p_2)\times\vec{n} \qquad (50)
\end{aligned}$$

where $\vec{n}$ is as given by equation (44).

Now, due to the previous observation that projection of the first–order derivatives into the tangent plane corresponds to projection of the triangle vertices into the tangent plane, and thus evaluate the first–order derivatives there, we may proceed under the assumption that the two triangles are coplanar and the normal to this plane is given by $\vec{n}$. It will now be shown that there exists a vector in the tangent plane such that the first–order derivative with respect to s of both the first and the second triangle deviate from this vector by at most 90°. We will do this by dotting equation (42) by $\vec{d} = (p_3 - p_2) \times \vec{n}$ and by using equations (49) and (50). Note that $\vec{d}$ is contained in the tangent plane and perpendicular to the dashed line through the shared line segment p_2p_3.

$$\begin{aligned}
\chi_s\bullet\vec{d} &= \frac{(t_3-t_1)_y\cdot(p_2-p_1)\bullet\vec{d}-(t_2-t_1)_y\cdot(p_3-p_1)\bullet\vec{d}}{(t_2-t_1)_x\cdot(t_3-t_1)_y-(t_2-t_1)_y\cdot(t_3-t_1)_x}\\
&= \frac{\mathcal{A}_1\cdot(t_3-t_2)_y}{(t_2-t_1)_x\cdot(t_3-t_1)_y-(t_2-t_1)_y\cdot(t_3-t_1)_x}
\end{aligned}$$

Here $\mathcal{A}_1$ represents the area of the first triangle times two. As for the second triangle the same procedure on equation (42) is applied, but $\vec{d}$ remains the same as before. By observing that $\vec{d} = (p_3 - p_2) \times \vec{n} = -(q_3 - q_2) \times \vec{n}$ and that $(t_3 - t_2)_y = -(r_3 - r_2)_y$, we get the nominator $(-\mathcal{A}_2) \cdot \Big(-(t_3 - t_2)_y\Big) = \mathcal{A}_2 \cdot (t_3 - t_2)_y$. Since $\mathcal{A}_1$ and $\mathcal{A}_2$ are both greater than zero, we know that the sign of the nominator will be the same in both cases. As for the denominator we know from equation (48) and the fourth rule that these will also have the same sign. So this tells us that the first–order derivatives with respect to s dotted against $\vec{d}$ will result in the same sign. In other words, they are contained in the same closed half-plane. A similar process can be done for the first–order derivative with respect to t in which case we get the nominator $\mathcal{A}_1 \cdot (t_2 - t_3)_x$ for the first triangle and $\mathcal{A}_2 \cdot (t_2 - t_3)_x$ for the second. Furthermore, we get the very same denominator as before. Note that $(t_3 - t_2)_y$ and $(t_2 - t_3)_x$ are not necessarily of the same sign which means that χ_s and χ_t are not necessarily in the same half-plane.

So what we have shown here is that for two adjacent triangles for which the four rules hold the first–order derivatives to be averaged are to a given extent similar in direction. At first glance this threshold may still seem too large but in practice these first–order derivatives to be averaged will be quite similar in direction. This is due to the natural piecewise coherency that exists between triangles in a group and the associated parameter space, see figure 18. This claim is further supported by the observation that χ_s and χ_t show how horizontal and vertical lines in the parameter space, map to the surface of χ. However, for the more extreme cases, we can solve the problem in code by introducing a user–defined angular threshold such that derivatives may only be averaged if the angle between them is smaller than the given threshold. Let us once again consider a collection of triangles in the same group and surrounding p. If, for every iteration of accumulation, we simply compare the given threshold with the angle between the current contribution and the current accumulated result, this would introduce an order dependency. To solve this, we use a different approach where the way groups are determined is adjusted. By the initial explanation given earlier, each triangle around p belongs to a single group and these groups around p are disjoint. This time, for each triangle in such a derived group, a list is made of which of the other triangles within the group are similar enough in first–order derivatives to be accumulated with. Such a list will initially result in a subgroup for every triangle and these unlike their parent groups are not disjoint. Finally, redundancies are removed so only unique subgroups remain and subsequently accumulation is performed. This approach preserves order independence because compatibilities are determined before accumulation is performed. This gives more control and may result in more tangent space splits since groups are potentially divided into several subgroups. For instance, setting the threshold to a strict value such as 2° results in a lot of tangent space splits, see figure 17(e). Notice however that the assigned tangent spaces are still correctly mirrored.

The actual parametrizations for such a triangular mesh are unknown so for these we will only support bump mapping in Blinn's approximate form. Additionally, since the first–order derivatives of each triangle are constant, the second–order derivatives are zero so accumulation of these would not make sense.

3.4 Pitfalls in the rendering process

An algorithm has been determined in section 3.3 to evaluate, for a triangular mesh, tangent space at vertex level. However for actual rendering of such a mesh, we must be able to determine tangent space for any position within a triangle which corresponds to the current pixel. It is interesting that there exists

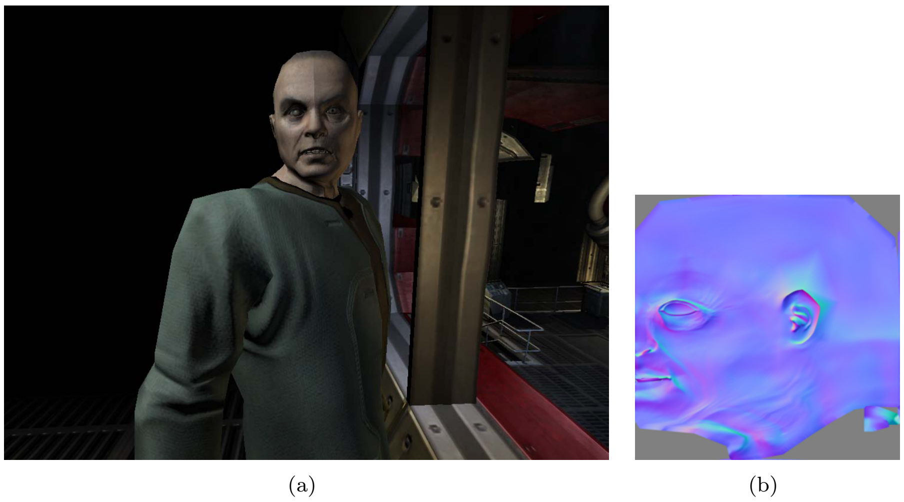
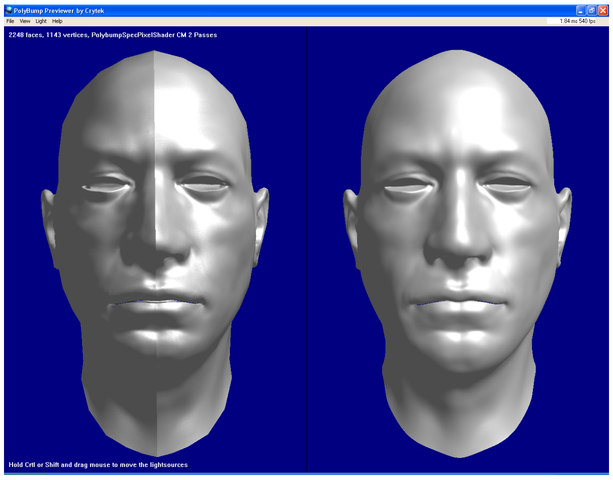


Figure 20: A character in the game "Doom 3" by the name Dr. Malcolm Betruger is shown in 20(a). The character is mirrored and normal mapped. The normal map is shown in 20(b) and does not appear to exhibit any errors. A shading seam is visible in 20(a) in the middle of the face.

no unique or correct way to interpret what is the accurate reconstruction of tangent space at pixel level. This problem is further complicated by the fact that there is no unique way to determine the approximations for vertex level tangent space either. These two observations are a strong indication of a compatibility problem in terms of sampled normal maps generated by different software. As explained at the end of section 2.2: To obtain the correct sampled normal, the transformation applied in the shader should be the exact inverse of the transformation applied when the normal map was generated. If the exact inverse is not used, the result will deviate from the original sampled normal. We have made it a point to average tangent spaces specifically to preserve continuities and subsequently if two triangles share an edge, and vertex level tangent spaces, the normal will have a continuous transition between the triangles even if there is deviation due to the exact inverse not being used. However, when there are tangent space splits, the result depends significantly on the exact inverse being used because in this case the transition of the deviation will be discontinuous and subsequently so will the transition of the transformed normal. This creates a discontinuity in the shading which is henceforth referred to as a *shading seam*.

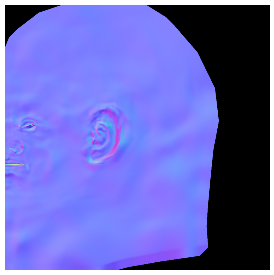
The problem is seen on several characters in the game "Doom 3" by ID Software, where there is a destinct shading seam in character faces, see figure 20(a). The mirrored model provided by IO-Interactive is tested with the normal map sampling tool Polybump by the company Crytek, see figure 21(a). Unlike the tool Melody, tangent space splits are correctly determined down the middle. However, since the shader in their viewer fails to use the correct inverse of the transformation used to make the normal map, there is a shading seam. Finally,



2248 faces, 1143 vertices, PolybumpSpecPixelShader CM 2 Passes

Hold Crtl or Shift and drag mouse to move the lightsources

(a)

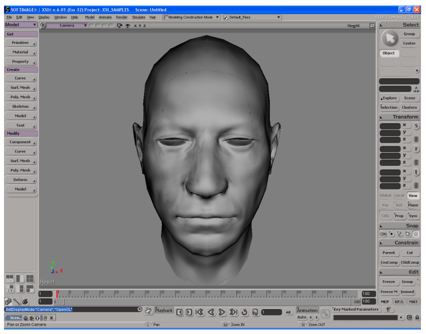


(b)

Figure 21: The normal map sampling tool Polybump by the company Crytek is shown in 21(a) with the model provided by IO-Interactive loaded. On the right side the high resolution version is seen and on the left side the corresponding normal mapped low resolution version is shown. A shading seam is visible in 21(a) in the middle of the face. The normal map generated by Polybump is shown in 21(b) and does not appear to exhibit any errors.

the model is tested using the product XSI by the company SoftImage which is a full modeler and animation tool also equipped with a normal map sampler. The normal map in this case was generated from the right side. Looking closely at the result in figure 22(a), the left side appears smudged and slightly distorted compared to the right side. This could be the result of an order dependency in tangent space evaluation, internally in XSI. Looking at the top of the head in figure 22(b) we see a clear shading seam in the middle region. Looking at the neck in figure 23(b), there appears to be a strange series of discontinuities there. Since XSI does not show its evaluated tangent spaces like Melody does, it is difficult to say for sure if it is related to tangent space splits. However, there are no clear signs of discontinuities in the generated normal map, see figure 23(d). An additional artifact visible in figure 23(c) is a shading seam down the middle of the head. Figures 20, 21, 22 and 23 are all cases where the normal map and the viewer was provided by the same company. As previously mentioned in this section, there exists no unique interpretation of tangent space on a triangular mesh, which is potentially a significant compatibility issue. Figure 24 shows the visual implications of using a Polybump–generated normal map with XSI and vice versa. There is a very evident problem when using the imported normal map. The details are heavily distorted.

None of the current middleware tools document their method for vertex level tangent space evaluation or pixel level tangent space evaluation. Since they are evidently error prone, they are not a good source of inspiration. Because there is no correct or unique way to determine tangent space at pixel level, we must choose a procedure. A sensible strategy might be to try and approximate the properties known about χ_s, χ_t and $\vec{n}$ as much as possible. We know that χ_s and χ_t are both perpendicular to $\vec{n}$, we know that $\vec{n}$ is a unit vector and that for a triangular mesh the surface normal is traditionally determined using barycentric interpolation of the vertex normal. Following in this foot-step by barycentrically interpolating the first–order derivatives, χ_s and χ_t will correspond exactly to a barycentric interpolation of the matrix N' defined in section 2.4. This initially presents a problem since linear interpolation of matrices generally results in skewing. As an example, linear interpolation of orthogonal matrices does in general not result in an orthogonal matrix. An additional example is rotation by more than 180° around some arbitrary direction, we could not approximate


(a)

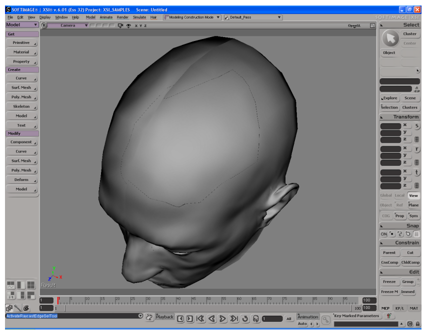
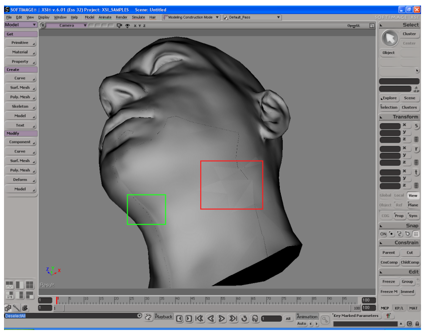
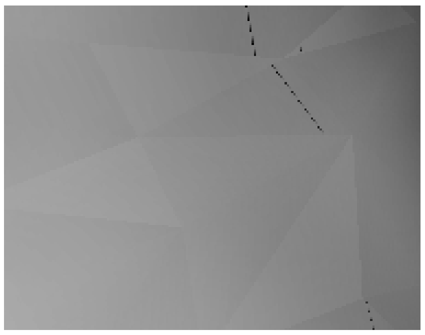
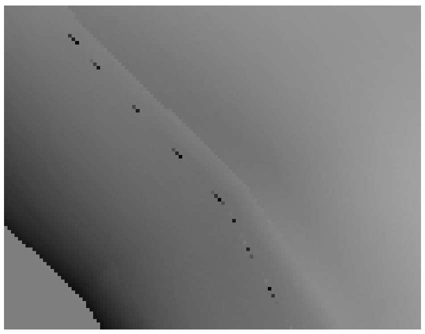

(b)

Figure 22: The modeling and animation package XSI by the company SoftImage is shown in 22(a). The normal mapped low resolution head by IO-Interactive is shown in the view and appears smudged on the left side and slightly distorted compared to the right side. Furthermore, a shading seam is visible on the top of the head in 22(b).


(a)


(b)


(c)

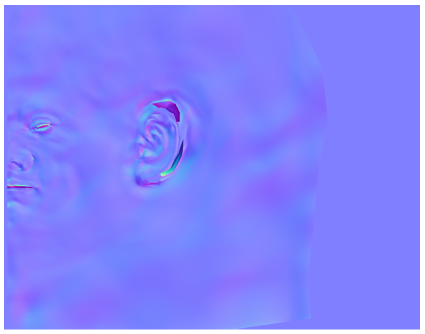
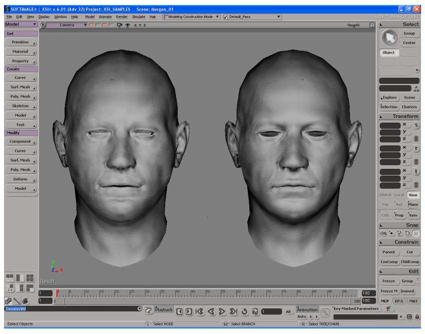
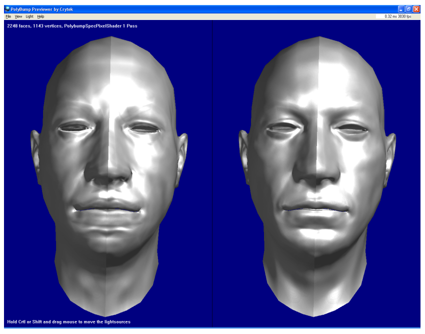

(d)

Figure 23: In 23(a) a series of discontinuities in the shading along the neck are revealed. The normal map generated by XSI (see 23(d)) does not appear to exhibit any errors or discontinuities.



(a)



(b)

Figure 24: To the left in 24(a) the normal map made with Polybump is applied and to the right the map was made by XSI. Similarly we see Polybump in 24(b) with a normal map made using XSI applied to the head on the left and with Polybump's own normal map applied to the right.

intermediate steps for such a case using linear interpolation between a source and a destination matrix. This tells us we need a good correspondence between the three tangent spaces which are assigned to a triangle at the corner vertices. In section 3.3.1, groups are determined based on connectivity rules and it was additionally argued in section 3.3.2 that we generally expect to find a good correspondence in first–order derivatives between triangles which obey the four rules. Furthermore, a user-defined angular threshold was introduced to control the correspondence required between the tangent space of a single triangle and the tangent spaces assigned at its vertices. Even though we should have a good correspondence, barycentric interpolation of N' will make the vectors χ_s, χ_t and $\vec{n}$ deviate from their intended lengths and also make χ_s and χ_t fail to be perpendicular to $\vec{n}$. In this thesis the following steps will be used to reconstruct a deformed tangent space at pixel level.

1. Normalize the interpolated normal $\vec{n}$.
2. The first–order derivatives are projected into the tangent plane which is determined by $\vec{n}$.
3. And finally these are scaled to the lengths determined by the barycentrically interpolated magnitudes corresponding to χ_s and χ_t respectively.

This will allow us to evaluate N' and thus equation (25) to obtain the approximate normal. It is important to acknowledge that the same procedure will have to be built into the software used to generate sampled normal maps to be used by this shader. It is also important that the same vertex level tangent spaces are supplied. In summary, when applying a sampled normal map to a triangular mesh, there is a dependency between the shader used for rendering and the tool used to generate the normal map.

Section 3.5 describes how the theory so far and the observations made in this section can be applied to real–time rendering while maintaining seamless shading.

3.5 Application to real-time rendering

For real-time rendering the three steps given in section 3.4 to interpolate between tangent spaces are expensive. The tests of section 2.6 showed that visual results are very resilient to Blinn's approximation. In contrast, shading seams will diminish the quality of the image. In this section examples of less accurate, but much faster, interpolation methods of tangent space will be given. These methods will not compromize continuity of the resulting surface normal. Furthermore, sampled normal maps will not be affected by this simplification as long as the same procedure is implemented in the normal map sampler to preserve compatibility with the shader.

3.5.1 Simplified normal perturbation

Let the normalized first–order derivatives be given as

$$\begin{aligned} \vec{t} &= \frac{\chi_s}{\|\chi_s\|} \\ \vec{b} &= \frac{\chi_t}{\|\chi_t\|} \end{aligned}$$

From this it follows that N' from section 2.4 can be written as

$$\begin{aligned} N' &= \left[\ \vec{t}\ \middle|\ \vec{b}\ \middle|\ \vec{n}\ \right] \begin{bmatrix} \|\chi_s\| & 0 & 0 \\ 0 & \|\chi_t\| & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ \left(N'^{-1}\right)^T &= \left(\left[\ \vec{t}\ \middle|\ \vec{b}\ \middle|\ \vec{n}\ \right]^{-1}\right)^T \begin{bmatrix} \frac{1}{\|\chi_s\|} & 0 & 0 \\ 0 & \frac{1}{\|\chi_t\|} & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{51}$$

In section 2.5.2 derivatives for an $m \times n$ bump map are precomputed using equations (35) and (36). In this section the precomputation will be changed such that the scale by m and n is not premultiplied onto the derivative.

$$\alpha'_s \simeq \frac{\lambda}{2} \cdot \left(\alpha(s_0 + \frac{1}{m}, t_0) - \alpha(s_0 - \frac{1}{m}, t_0)\right) \tag{52}$$

$$\alpha'_t \simeq \frac{\lambda}{2} \cdot \left(\alpha(s_0, t_0 + \frac{1}{n}) - \alpha(s_0, t_0 - \frac{1}{n})\right) \tag{53}$$

Note that λ represents the user–defined scalar value introduced at the end of section 2.5.2. It follows that

$$\begin{bmatrix} -\alpha_s \\ -\alpha_t \\ 1 \end{bmatrix} = \begin{bmatrix} m & 0 & 0 \\ 0 & n & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -\alpha'_s \\ -\alpha'_t \\ 1 \end{bmatrix} \tag{54}$$

and by defining the following matrices

$$\begin{aligned} T &= \left[\ \vec{t} \,\middle|\, \vec{b} \,\middle|\, \vec{n}\ \right] \\ S &= \begin{bmatrix} \frac{m}{\|\chi_s\|} & 0 & 0 \\ 0 & \frac{n}{\|\chi_t\|} & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

the perturbed normal can be rewritten the following way

$$\left(N'^{-1}\right)^T \begin{bmatrix} -\alpha_s \\ -\alpha_t \\ 1 \end{bmatrix} = \left(T^{-1}\right)^T \cdot S \begin{bmatrix} -\alpha'_s \\ -\alpha'_t \\ 1 \end{bmatrix} \tag{55}$$

Three approximations will be applied to simplify interpolation of tangent space and evaluation of the perturbed normal. The **first approximation** is $\frac{m}{\|\chi_s\|} \simeq \frac{n}{\|\chi_t\|}$ and that this factor is a constant k which simplifies S.

$$S \simeq \begin{bmatrix} k & 0 & 0 \\ 0 & k & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The intuitive interpretation of the factor $\frac{m}{\|\chi_s\|}$ is the amount of texels covered when we move one unit along χ_s. From this it follows that the intuitive interpretation of $k = \frac{m}{\|\chi_s\|} = \frac{n}{\|\chi_t\|}$ is that one unit along χ_s covers the same amount of texels as moving one unit along χ_t. So given the simplification of S we get

$$\left(N'^{-1}\right)^T \begin{bmatrix} -\alpha_s \\ -\alpha_t \\ 1 \end{bmatrix} \simeq \left(T^{-1}\right)^T \begin{bmatrix} -k \cdot \alpha'_s \\ -k \cdot \alpha'_t \\ 1 \end{bmatrix} \tag{56}$$

Since k is constant, we can simplify equation (56) further by premultiplying k onto the user–defined scalar value λ or possibly ignore k altogether and consider it a user responsibility to tweak the product of the two as one value.

The matrix T has unit length vectors $\vec{t}$, $\vec{b}$ and $\vec{n}$ in the columns. Vectors $\vec{t}$ and $\vec{b}$ are perpendicular to $\vec{n}$ but not necessarily perpendicular to each other, so T is in general not an orthogonal matrix. The **second approximation** is that we can replace $\left(T^{-1}\right)^T$ in equation (56) with the matrix T itself. This approximation holds well when T is close to orthogonal which is the case when the bump map is a conformal map onto the surface χ. Such a conformal map will also ensure that the first approximation holds well $\frac{m}{\|\chi_s\|} = \frac{n}{\|\chi_t\|}$. The second approximation leads to the following

$$\left(N'^{-1}\right)^T \begin{bmatrix} -\alpha_s \\ -\alpha_t \\ 1 \end{bmatrix} \simeq T \begin{bmatrix} -\alpha'_s \\ -\alpha'_t \\ 1 \end{bmatrix} \tag{57}$$

where k is now premultiplied onto λ on the right side of equation (57).

For a triangular mesh specifically, the matrix T is defined at vertex level and made using the algorithm outlined in section 3.3. Note that the accumulated magnitudes are unused here. The **third and final approximation** made in this section is that the matrices T during rasterization are barycentrically interpolated with no additional modifications applied. So unlike section 3.4, the vectors $\vec{t}$, $\vec{b}$ and $\vec{n}$ are not normalized and no projection of $\vec{t}$, $\vec{b}$ into the tangent plane takes place. The interpolated coefficients of T are simply used directly in equation (57). Note that the resulting perturbed normal after evaluation of equation (57) will of course have to be normalized before use in the lighting model.

The important thing to acknowledge is that although the three approximations applied in this section will cause the perturbed normal to deviate from the accurate one when bump mapping, they will not prevent a continuous transition of it. This is particularly important when using sampled normal maps. As stated in section 3.4, continuity is preserved by using the exact same evaluation of tangent space in the normal map sampling tool such that the exact inverse can be created.

Let T_1, T_2 and T_3 for a given triangle be the matrices for T assigned at vertex level and let (u, v) represent the barycentric coordinate. Equation (57) is evaluated in the shader based on

$$T = T_1 \cdot (1-u-v) + T_2 \cdot u + T_3 \cdot v \tag{58}$$

$$T^{-1} = \left(T_1 \cdot (1-u-v) + T_2 \cdot u + T_3 \cdot v\right)^{-1} \tag{59}$$

equation (58) and subsequently the sampler tool must use equation (59). On a GPU, equation (58) is evaluated in hardware before pixel shader execution.

3.5.2 Customization of tangent space

Whether to use the procedure explained in section 3.4, the procedure presented in section 3.5.1 or any other interpretation of tangent space, an important subtlety to notice is that since the sampler tool is used for preprocessing, the choice of T should be based on the purpose and the requirements of the shader used. The sampler tool should subsequently be customized to accommodate the choice made for the shader and not the other way around. This is a problem with current middleware software because they do not offer user customization of T. In fact, as previously mentioned the vendors do not even provide documentation for their own interpretation of T used in their tool.

The concept of customizing T to fit the needs of the shader or the application, using the normal map makes way for several interpretations of T. For instance, the footprint can be reduced by only storing two vectors of T per vertex in memory. As an example let us say we keep the vectors $\vec{n}$ and $\vec{t}$ in memory. Using this concept, $\vec{b}$ is optimized away by using the following approximation in the vertex shader.

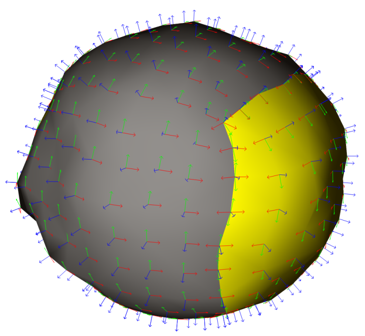
$$\vec{b} \simeq \pm\vec{n} \times \vec{t} \tag{60}$$

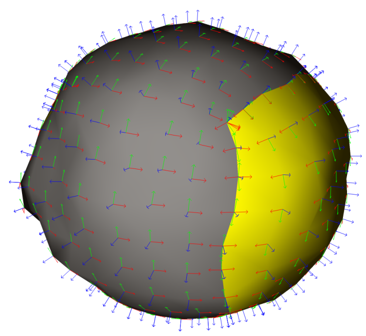
The problem with this approximation is determination of the sign. The algorithm given in section 3.3.1, for vertex level tangent space evaluation, categorizes every triangle as orientation preserving or orientation reversing. Averaging only takes place between triangles which belong to the same category. This motivates the idea of splitting the triangular mesh into two parts such that each part consists of triangles which belong to one category only. When the part consisting of triangles which belong to the orientation preserving category are drawn, approximation (60) should be evaluated using $+$ and correspondingly the part which consists of triangles which belong to the orientation reversing category should use $-$ in approximation (60) during execution of the vertex shader. This sign can trivially be passed as a shader parameter value and subsequently does not need to be stored per vertex.

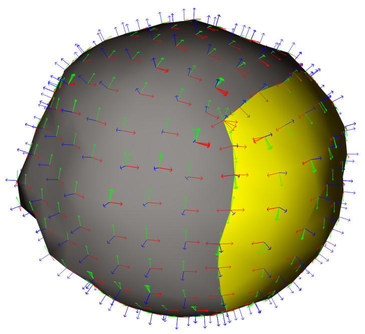
GPUs today only provide a limited set of interpolaters and sometimes it can be a struggle to design a shader so it stays within this limit. To reduce the overhead on the amount of interpolaters used, evaluation of approximation (60) can be moved to the fragment shader. As a result of this, $\vec{b}$ is no longer passed to an interpolater from the vertex shader. The previously mentioned sign is thus passed as a parameter value to the fragment shader and not the vertex shader. Given the optimization described here, a cross product and a vector scalar multiply is inserted into the fragment shader. Note that evaluation of $\vec{b}$ in the vertex shader as opposed to evaluation of $\vec{b}$ in the fragment shader does not provide the same result of the matrix T. It is essential that the chosen procedure must be properly matched in the normal map sampling tool such that the accurate inverse is evaluated there. Thus avoiding shading seams at edges surrounding tangent space splits.

3.6 Results

I have implemented a simple tool to generate sampled normal maps as explained in section 2.2 and using the algorithm described in section 3.3 for vertex level tangent space evaluation. During rendering, reconstruction of tangent space at pixel level, is done using the procedure outlined in section 3.4 as opposed to the alternatives suggested in section 3.5. This is because performance is not the focus of this thesis and because the procedure provides a better approximation for the first–order derivatives with respect to s and t which are used in my texture filtering implementation as explained in section 2.5.1. Additionally, the choice, given the improved accuracy, corresponds better to the theory of section 2. Furthermore, I have implemented the same procedure in the sampling tool for reconstruction of tangent space at texel level to maintain compliance with the shader used for rendering.


(a)


(b)


(c)

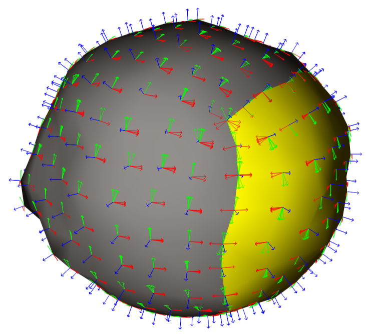

(d)

Figure 25: Here we see the effect of setting the angular threshold to different values. In 25(a) the angular threshold was disabled. In figures 25(b)-25(d) the threshold was set to 45°, 25° and 2° respectively. The amount of tangent space splits increases as the threshold is lowered.

The first test is on vertex level tangent space evaluation given different angular thresholds. The test is performed on a triangular mesh which resembles a dented ball (see figure 25(a)). The ball is made from two pieces similar to the topology of a tennis-ball, these are shown in gray and yellow on the figure. Each piece in texture coordinates is given exactly one half of the texture space, the gray piece is assigned the left half and the yellow piece receives the right half. The dents were applied using a noise modifier in 3D Studio Max and the coordinate systems seen on the figure are tangent spaces at vertex level generated

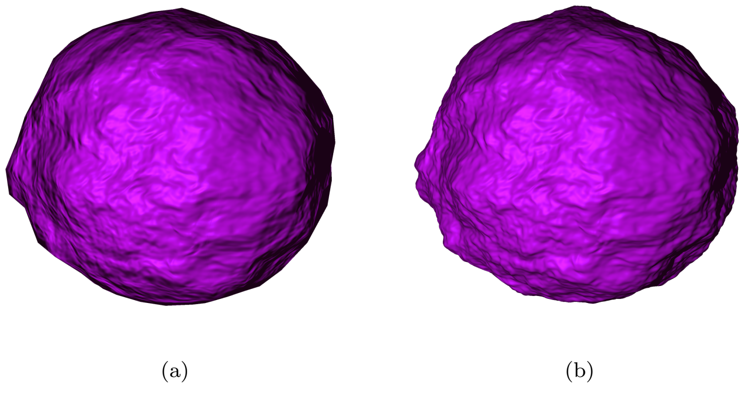


Figure 26: In 26(b) we see a high resolution version of the dented ball. In 26(a) we see the normal mapped low resolution version of the ball. The normal map was sampled with the angular threshold disabled.

by the algorithm described in section 3.3. To generate the result seen in this particular figure, the angular threshold described in section 3.3.2 was disabled. If we consider every vertex of the mesh given as a tripple of the form $(p, \vec{n}, t)$ where the position is p, the vertex normal is $\vec{n}$ and the texture coordinate is t, then this particular mesh is constructed from 450 unique tripples. Applying the tangent space generation with the angular threshold disabled in this particular case resulted in exactly 450 unique pentuples of the form $(p, \vec{n}, t, \chi_s, \chi_t)$ where χ_s and χ_t are the generated tangent vectors, so in this specific case there were no splits caused by tangent space generation. Looking at the figure we also see that all coordinate systems are right–handed, so the orientation is preserved everywhere which confirms that there are no violations of the fourth rule given in section 3.3.1.

The figure 25(b) shows the same model but with the angular threshold set to $45°$ which resulted in 475 unique pentuples. In particular, we see on the figure the tangent space splits where the corner of the yellow piece is. The coordinate system that was split up coincides with the surrounding tangent spaces of the gray piece so from this we conclude the split tangent space belongs to this piece. This conclusion is also supported by the rapid change in orientation of tangent space we can observe on the gray piece near this corner in figure 25(a).
In figure 25(c) the angular threshold was set to $25°$ which resulted in 701 unique pentuples. The effect is confirmed by the presence of multiple tangent space splits on the figure. Finally, in figure 25(d) the angular threshold was lowered to $2°$ which generated 1988 unique pentuples. Correspondingly, tangent space splits can be observed everywhere on the figure. In the next test, a high resolution version of the ball is created in 3D Studio Max by making a dense sphere

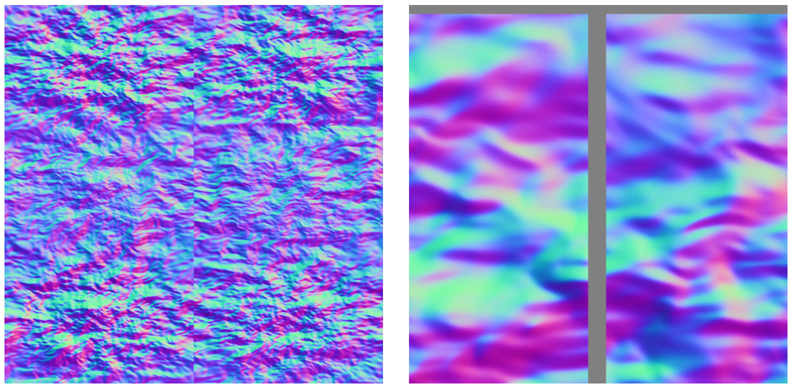

(a) (b)

Figure 27: In 27(a) a normal map is shown. It was generated based on the low and the high resolution versions of the dented ball and with the angular threshold disabled. There is a discontinuity down the middle of the normal map. Such a discontinuity can cause problems when filtering. We see in 27(b) a close–up where the domain assigned to each half of the ball has been shrunk. This creates the gap between the two parts which is shown in grey. Each pixel in the gap is subsequently replaced by the closest valid normal. This creates a dilation of the domains.

and by applying the same noise modifier to it, the result is shown in figure 26(b). The tool mentioned at the beginning of this section was used to generate a sampled normal map, where the angular threshold was disabled during tangent space generation. The normal map is shown in figure 27(a) and the resulting normal mapped ball is shown in figure 26(a). The similarity between this result and that of figure 26(b) is convincing, the primary differences are seen mainly at the silhouette.

Taking a closer look at the normal map in figure 27(a) reveals a distinct discontinuity between the two half domains, based on section 2.5.1 we should be able to observe artifacts as a result of this. In figure 29(a) we see a close–up of the same scene as in figure 26(a) but this time tangent spaces are also shown in the figure. The close–up is of the previously examined corner and clearly we now see a filtering seam where the two pieces meet. Ideally two textures instead of one should be used to preserve the discontinuity between the two halves. However, this could for more complex cases result in a lot of individual textures. As a possible alternative the two halves can be shrunk such that a small surrounding margin is created. A close–up is shown in figure 27(b) where the margin is represented by the color gray. Next, dilation is applied such that every gray pixel is replaced by the closest pixel which is not gray, this can be

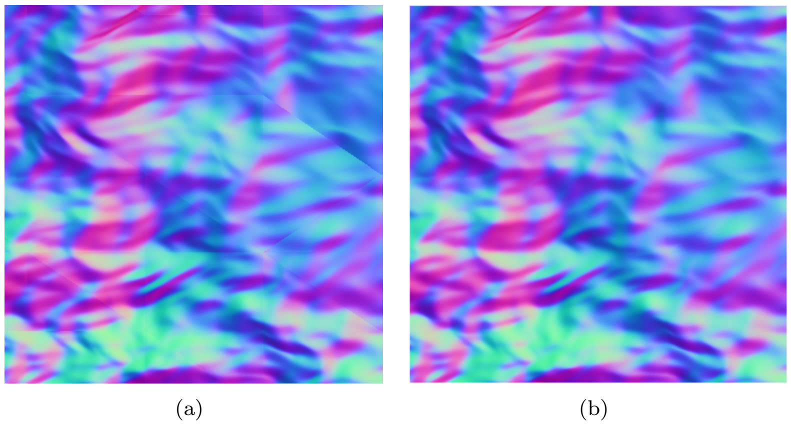
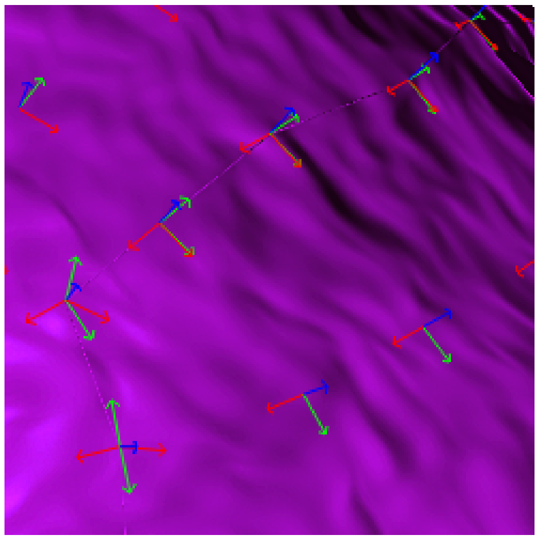


Figure 28: In 28(a) we see the effect of setting the angular threshold to a strict value, in this case the value is 2° and as a result discontinuities are visible in the sampled normal map. In contrast we see the effect of disabling the angular threshold in figure 28(b), no discontinuities are found here.

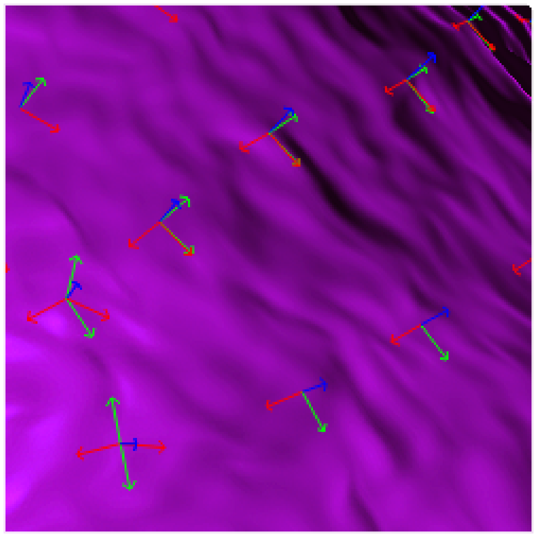
done easily either in code or in Photoshop. The result, which is successful, is shown in figure 29(b). However, the approch is not ideal since, depending on the thickness of the extra margin used, the two shrunken halves will eventually for some mip map level begin to mix also known as bleeding. However, at distance due to mip mapping, the details will begin to fade and hide such issues, for instance the smallest level is a 1×1 texture which is one normal only.

In figure 28(a) we see a close–up of the normal map generated with the angular threshold set to 2°, and as predicted in section 2.5.1, we see as a result of this discontinuities in the normal map. In figure 28(b), we see the same location of the normal map, but this time the angular threshold was disabled and here we see no discontinuities. We can see the result of applying the normal map based on the 2° threshold and the corresponding set of tangent spaces in figure 30(b). We do see filtering seams between adjacent triangles, but they are not as explicit as those seen in figure 29(a). This is because the difference in orientation of tangent space between the two pieces is much more extreme than the differences found between the split tangent spaces due to the strict 2° threshold. In figure 30(a), we see the same close–up but based on the angular threshold disabled and, as we see, the filtering seams are now gone.

Finally, the model of the head provided by IO-Interactive is tested using the code of this thesis. The first test shows the tangent spaces evaluated, see figure 31(a). Since it is shown without normal mapping applied, it does not appear very detailed. The tangent spaces were generated with the angular threshold disabled, furthermore tangent space splits appear down the middle as predicted in section 3.3.1 since mirroring is a special case of a violation of the fourth rule.

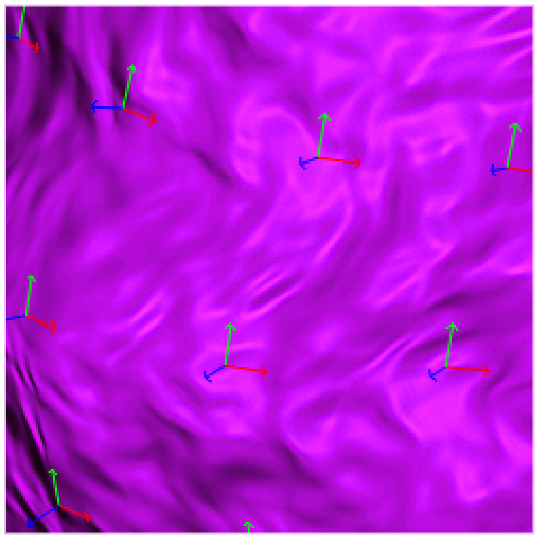


(a)



(b)

Figure 29: A close–up of the normal mapped low resolution dented ball is shown in 29(a). The figure shows a filtration seam caused by the discontinuity down the middle of the normal map. The result of shrinking the two domains and performing dilation is shown in 29(b). As we see, the error is no longer visible here.


(a)

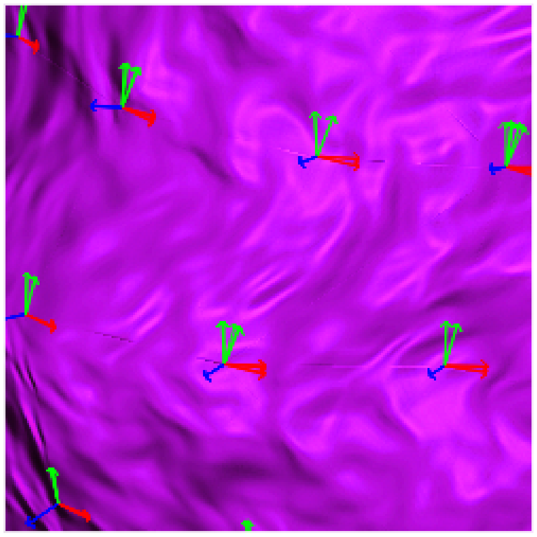
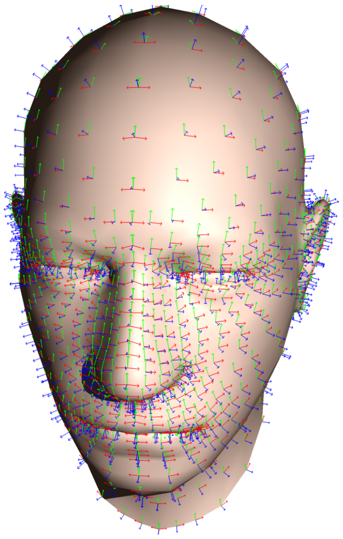

(b)

Figure 30: A close–up of the normal mapped dented ball is shown in 30(b). The result is based on the angular threshold set to 2°. The strict threshold creates tangent space splits which result in discontinuities in the generated normal map. These cause the filtration seams seen in the figure. In 30(a) we see the result when the angular threshold is disabled. The filtration seams are gone.


(a)

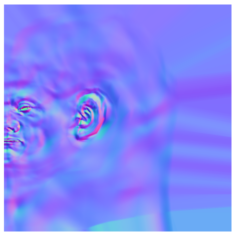

(b)

Figure 31: The low resolution head provided by IO-Interactive is shown in 31(a) with no normal mapping applied to it. Vertex level tangent spaces generated with the algorithm of this thesis with the angular threshold disabled are shown in the figure. The generated normal map is shown in 31(b).

An additional sign of mirroring is that the tangent spaces on one side appear to be the result of reflecting the tangent spaces of the opposite side. On the left side of the head, the tangent spaces are left–hand coordinate systems as opposed to the right side where they are right–hand coordinate systems.
In figure 32(b), we see a similar model but in much higher resolution and with much more fine detail. From the low resolution and the high resolution version of this head, a normal map is sampled and shown in figure 31(b). The visual result of applying the normal map to the low resolution model is shown in figure 32(a) and clearly the result is very convincing and there are no visible errors. All the fine detail we see in figure 32(b) is inherited and applied to the low resolution model and as before the silhouette is the primary location where the illusion is revealed.

In figures 32(c) and 32(d) we see a similar comparison between the two but seen from the side. As we can see, all the nice surface irregularities on the side of the high resolution head appear the same on the low resolution head and additionally the details and curves inside the ear also appear the same. In figure 32(c), a special case of the silhouette problem is revealed on the shadow across the throat and neck. The shape of a shadow is the projection of the silhouette as seen from the light source so since the properties of the high resolution model

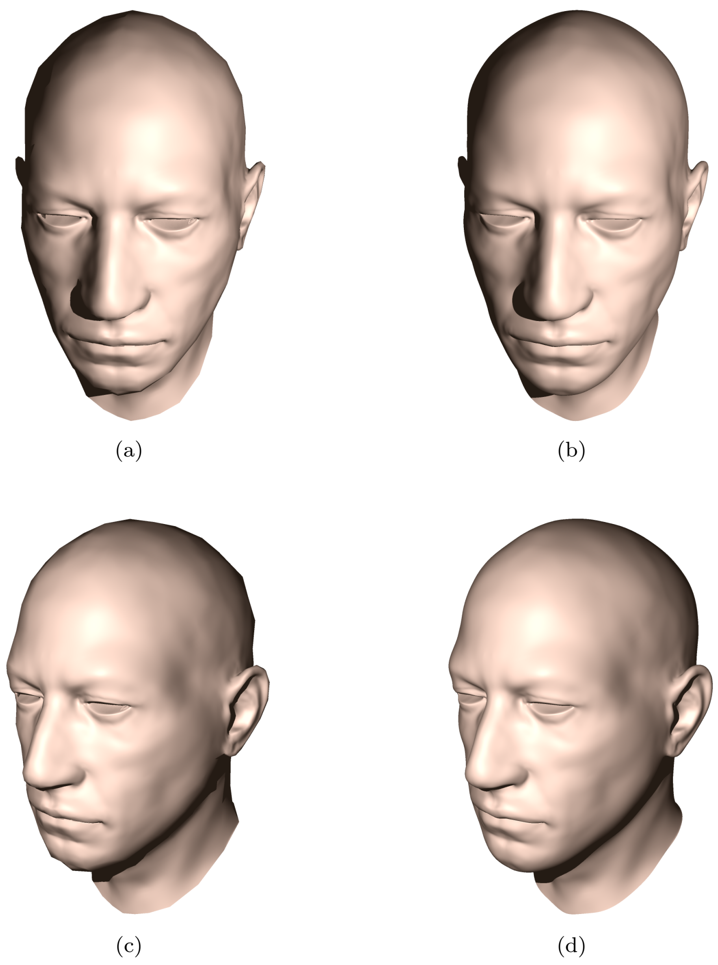
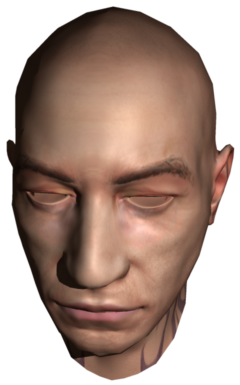
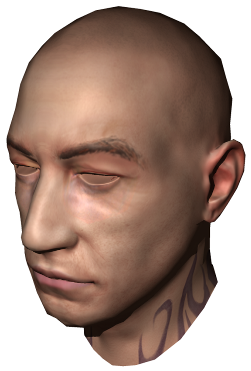


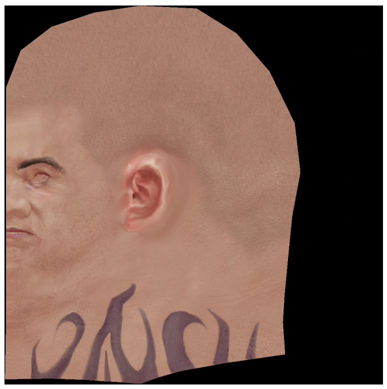
Figure 32: Here a comparison between a normal mapped low resolution head and a high resolution version is shown. In 32(a) and 32(c) we see the normal mapped head from the front and the side respectively. In 32(b) and 32(d) we see the corresponding results using the high resolution model.



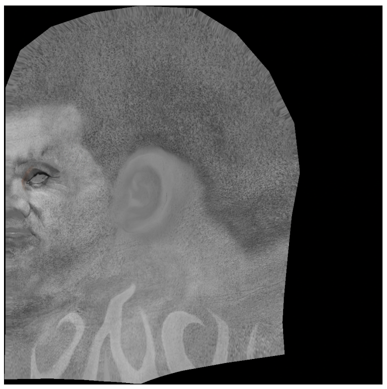
(a)



(b)



(c)



(d)

Figure 33: Here the head provided by IO-Interactive is shown, normal mapped and with a diffuse and a specular texture applied. The diffuse texture is shown in 33(c) and the specular texture is shown in 33(d). The result seen from the front and from the side is shown in figures 33(a) and 33(b).

are unused in the generation of these, this results in the distinction we see in the figure.
In figures 33(c) and 33(d), we see the diffuse and specular textures of the head, both were hand drawn in Photoshop by an artist at IO-Interactive. The end result achieved by applying these to the head is shown in figure 33. The result looks convincing though the subtle details of the normal map are not as apparent as they were before the diffuse texture was applied. In exchange, we get new details such as: the tattoo on the neck, various skin color irregularities, the more accurate color of the lips and also the subtle appearance of veins and dark tones under the eyes.

4 Recent research on normal map filtering

As explained in section 2.5.1, normal maps and subsequently bump maps cannot be filtered the traditional way. Previous authors have tried to solve the problem by approximating the distribution of normals in the coverage of a texel by some chosen function. For instance in [Sch97], the distribution of bump derivatives (α_s, α_t) in the coverage are approximated by an ellipse, and in [ON97] a single 3D Gaussian is fitted onto the distribution of normals. Since the normals in the coverage can gather in more complex distributions, such as a set of clusters, an approach to support fitting of multiple Gaussians is given in [TLQ$^+$05]. Fitting in this paper is applied to the first and second components of the tangent space unit normals in the coverage which corresponds to an initial parallel projection into the tangent plane.

In this section, the recent work on normal map filtering by Charles Han et al. [HSRG07], presented at Siggraph 2007, will be explained in detail. Like his predecessors, Han tries to solve the problem by approximation of the distribution of normals. Two novel techniques are presented, the first uses Spherical Harmonics for the approximation and is described in section 4.1. This technique is primarily suitable for low-frequency materials such as a Lambertian surface so an alternative based on clustering is suggested in [HSRG07] and is described in section 4.2. The second technique, similar to the work in [TLQ$^+$05], uses a mixture of Gaussian-like functions but defined on the spherical domain as opposed to the planar Gaussian fits used in [TLQ$^+$05]. This technique and the mixture of functions used is described in section 4.3 and fitting of this mixture onto a distribution of normals is achieved using a method known as spherical EM [BDGS05] which is described in section 4.4. Finally in section 4.5 results are given.

4.1 Representation of the normal distribution

As mentioned in section 2.5.1, Blinn states that a more correct approach to the filtering problem is to evaluate the shading against every resulting normal and then averaging these. Han follows this path and states more explicitly: In screen space, the exitant radiance or pixel color at a surface location x should represent the average radiance at the N corresponding finer-level texels q.

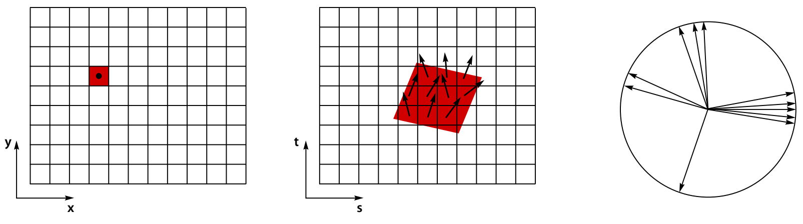
Let $\rho^{eff}(\vec{\omega}_i, \vec{\omega}_o; \vec{n})$ denote the effective BRDF, that is the bidirectional reflection distribution function. Furthermore, $\vec{\omega}_i$ is the incident light direction and $\vec{\omega}_o$ is the direction towards the eye-point and $\vec{n}$ in this context corresponds to the final replaced normal received from the normal map. The pixel to be shaded corresponds to some region in the normal map as shown in figure 34(b). Let the normals in this coverage be given by the following sequence, $\{\vec{n}_1, \vec{n}_2, ..., \vec{n}_N\}$ where N is the amount of normals in the coverage. The effective BRDF is subsequently rewritten as the following summation, also given in [HSRG07], yielding the averaged result.

$$\rho^{eff}(\vec{\omega}_i, \vec{\omega}_o; x) = \frac{1}{N} \sum_{q=1}^{N} \rho(\vec{\omega}_i, \vec{\omega}_o, \vec{n}_q) \tag{61}$$

As shown in section 2.5.1, filtering bumps or normals directly does not provide an accurate result. The key idea in [HSRG07] is to rewrite equation (61) such that it is expressed as convolution on the spherical domain. The function ρ is the filter and γ which is the *normal distribution function* (NDF) is the function (see figure 34(c)) to be convolved and is given as

$$\gamma(\vec{n}) = \frac{1}{N} \sum_{q=1}^{N} \delta(\vec{n} - \vec{n}_q)$$

where δ is Dirac's delta function on the spherical domain. This definition allows



(a) (b) (c)

Figure 34: Each pixel to be drawn 34(a) covers in the full resolution normal map some selection of normals 34(b). This selection of normals 34(c) is represented on the spherical domain by the normal distribution function γ.

us to rewrite equation (61) into its following continuous form

$$\rho^{eff}(\vec{\omega}_i, \vec{\omega}_o; x) = \int_{S^2} \rho(\vec{\omega}_i, \vec{\omega}_o, \vec{n}) \cdot \gamma(\vec{n})\, \mathrm{d}\vec{n} \tag{62}$$

At first glance, it appears as if this rewrite only made matters worse, however, the idea here is to project the functions ρ and γ onto an orthonormal set of functions. This results in a sequence of coefficients which, as opposed to the normals, can be filtered. In this case, the basis functions being used are the *Real Spherical Harmonics* functions which are defined on the domain $(\theta, \varphi) \in [0;\pi] \times [0; 2\pi[$ and given as

$$\begin{aligned} K_l^m &= \sqrt{\frac{2l+1}{2\pi} \frac{(l-|m|)!}{(l+|m|)!}} \\ Y_l^m(\theta,\varphi) &= \begin{cases} K_l^m \cos(m\varphi)\, P_l^m(\cos\theta) & m > 0 \\ K_l^m \sin(-m\varphi)\, P_l^{-m}(\cos\theta) & m < 0 \\ \frac{1}{\sqrt{2}} K_l^0 P_l^0(\cos\theta) & m = 0 \end{cases} \end{aligned} \tag{63}$$

And for $l \in \mathbf{N}_0$ and $m \in \{0, ..., l\}$ the functions $P_l^m : [-1; 1] \to \mathbf{R}$ are the *Associated Legendre Polynomials* which for a given l define a real orthogonal set of functions. We can evaluate these using the following recurrence relations.

$$\begin{aligned}
P_0^0(t) &= 1 \\
P_m^m(t) &= (1-2m)\sqrt{1-t^2}P_{m-1}^{m-1}(t) \\
P_l^m(t) &= t\left(\frac{2l-1}{l-m}\right)P_{l-1}^m(t) - \left(\frac{l+m-1}{l-m}\right)P_{l-2}^m(t) \\
P_{m+1}^m(t) &= t(2m+1)P_m^m(t)
\end{aligned}$$

Note from equation (63), that $Y_l^0(\theta, \varphi)$ only depends on θ. These basis functions, of $m = 0$ and $l \in \mathbf{N}_0$, are referred to as being *radially symmetric*. The basis function of lowest order is $Y_0^0(\theta, \varphi)$, which is constant over the entire domain, and thus represents a sphere. As the value of l increases so does the frequency level of $Y_l^m(\theta, \varphi)$.
A thorough introduction to Spherical Harmonics or the Associated Legendre Polynomials is beyond the scope of this thesis. For additional information, the reader is referred to [Mac48] and [Gla94].

As previously mentioned, the Real Spherical Harmonics functions are an orthonormal set so the following property holds

$$\int_{S^2} Y_l^m(\vec{n})Y_{l'}^{m'}(\vec{n})\,\mathrm{d}\vec{n} = \delta_{ll'} \cdot \delta_{mm'} \tag{64}$$

Now let $f : S^2 \to \mathbf{R}$ be a bounded and Riemann integrable spherical function. As usual projection onto the chosen basis is achieved by integration of the product between the given function and each basis function.

$$f_l^m = \int_{S^2} Y_l^m(\vec{n})f(\vec{n})\,\mathrm{d}\vec{n} \tag{65}$$

We then define the series associated with f as

$$F(\vec{n}) = \sum_{l=0}^{\infty}\sum_{m=-l}^{l} f_l^m Y_l^m(\vec{n})$$

Given an integer $L \in \mathbf{N}$ we denote the *partial sums* of this series by

$$s_L(f)(\vec{n}) = \sum_{l=0}^{L-1}\sum_{m=-l}^{l} f_l^m Y_l^m(\vec{n}) \tag{66}$$

For each l we sum up $2l + 1$ terms which in total results in L^2 terms for $s_L(f)$. Of course $s_L(f) \neq f$ and unfortunately even for high values of L there is no guarantee that $\max|s_L(f)(\vec{n}) - f(\vec{n})|$ is close to zero for all $\vec{n}$. However, if we switch metric to the following $\int_{S^2}(s_L(f)(\vec{n}) - f(\vec{n}))^2\,\mathrm{d}\vec{n}$, then it can be shown that this converges uniformly to zero as $L \to \infty$. So in this sense, the function f and the associated series are similar.

Let $g(\vec{n})$ be another bounded and Riemann integrable spherical function with corresponding partial sums $s_L(g)(\vec{n})$, then the following property is a direct result of equation (64).

$$\int_{S^2} s_L(f)(\vec{n}) s_L(g)(\vec{n})\, \mathrm{d}\vec{n} = \sum_{l=0}^{L-1} \sum_{m=-l}^{l} f_l^m g_l^m$$

So in other words, the integration is reduced to a simple dot product over the Spherical Harmonic coefficients.
The overall idea in [HSRG07] is to use this principal for a limited set of coefficients to evaluate equation (62). This is done by considering the partial sums $s_L(\rho)$ and $s_L(\gamma)$ as approximations of ρ and γ respectively. This optimization is demonstrated on f and g by the following equation.

$$\begin{aligned} \int_{S^2} f(\vec{n}) g(\vec{n})\, \mathrm{d}\vec{n} &\simeq \int_{S^2} s_L(f)(\vec{n}) s_L(g)(\vec{n})\, \mathrm{d}\vec{n} \\ &= \sum_{l=0}^{L-1} \sum_{m=-l}^{l} f_l^m g_l^m \end{aligned} \tag{67}$$

We can apply the same optimization to equation (62). However, the fact that ρ depends on six angles, two for each direction, will result in a very large amount of coefficients. In order to reduce this amount significantly two assumptions about the BRDF are made.
The first assumption is that we can reduce the complexity of the BRDF to two input directions instead of three $\rho(\vec{\omega}, \vec{n})$ where the direction $\vec{\omega}(\vec{\omega}_i, \vec{\omega}_o)$ depends on $\vec{\omega}_i$ and $\vec{\omega}_o$. As examples of such cases, we have Lambertian reflectance where the transfer function is simply the cosine of the incident angle. Subsequently $\vec{\omega} = \vec{\omega}_i$ and the BRDF is given as

$$\rho(\vec{\omega}_i, \vec{n}) = \max(\vec{\omega}_i \bullet \vec{n}, 0)$$

Another example is the Blinn-Phong specular model, given a specular exponent $s >= 1$ and using the direction $\vec{\omega}_h = \frac{\vec{\omega}_i + \vec{\omega}_o}{\|\vec{\omega}_i + \vec{\omega}_o\|}$, which is also known as the halfway vector. The transfer function is given by the following equation

$$\rho(\vec{\omega}_h, \vec{n}) = \max(\vec{\omega}_h \bullet \vec{n}, 0)^s$$

Unfortunately, this assumption excludes the traditional Phong Specular model since the reflection vector $\vec{R}$ depends on $\vec{\omega}_i$ and the normal $\vec{n}$.

The second assumption made is that the transfer function can be determined from cosine to the angle between the vectors $\vec{n}$ and $\vec{\omega}$. By this assumption, the BRDF is restricted to functions which are radially symmetric about the normal. Because of this, the projection of such a transfer function onto the spherical harmonic basis functions can only lead to nonzero coefficients when $m = 0$, that is the basis functions of the form Y_l^0. By this observation, the amount of coefficients needed has been reduced to L coefficients instead of L^2.

In the remainder, we will only focus on one transfer function which is given by the equation

$$f_s(u) = \max(u, 0)^s \tag{68}$$

Based on equation (68) the Blinn-Phong specular model is given by $\rho(\vec{\omega}_h, \vec{n}) = f_s(\vec{\omega}_h \bullet \vec{n})$ and Lambertian reflection by $\rho(\vec{\omega}_i, \vec{n}) = f_1(\vec{\omega}_i \bullet \vec{n})$.

Let the normal $\vec{n}$ be given by its spherical coordinates (θ_1, φ_1) and similarly the direction $\vec{\omega}$ by (θ_2, φ_2) and as previously mentioned the domain is $[0;\pi] \times [0; 2\pi[$. Given this domain, we use the following parametrization of the unit sphere

$$S(\theta, \varphi) = (\sin(\theta)\cos(\varphi), \sin(\theta)\sin(\varphi), \cos(\theta)) \tag{69}$$

Let the matrix $R_{\vec{n}}$ denote some chosen rotation such that $\vec{n} \to (0, 0, 1)$. Additionally, let (θ_2', φ_2') be the spherical coordinates of $\vec{\omega}$ after rotation by $R_{\vec{n}}$. In this space θ_2' is the angle between $\vec{n}$ and $\vec{\omega}$ so the BRDF can be established entirely from this parameter as

$$\begin{aligned}\rho_s(\vec{\omega}_i, \vec{n}) &= f_s(\cos(\theta_2')) \\ &= \max(\cos(\theta_2'), 0)^s\end{aligned}$$

To establish the spherical harmonics series associated with this transfer function, we need to evaluate the projection of it onto each Y_l^0 using equation (65). Note that $\cos(\theta_2') = \vec{\omega} \bullet \vec{n}$ and additionally that θ_2' is zero at the north pole and π at the south pole, so the poles are reversed. This must be taken into account when we integrate in spherical coordinates. We do this by replacing $\cos(\theta_2')$ with $\cos(\pi - \theta)$ during integration.

$$\begin{aligned}\rho_l &= \int_{S^2} f_s(\vec{\omega} \bullet \vec{n}) Y_l^0(R_{\vec{n}} \cdot \vec{\omega})\, \mathrm{d}\vec{n} \\ &= \int_0^{2\pi}\int_0^{\pi} f_s(\cos(\pi-\theta)) \sqrt{\frac{2l+1}{4\pi}} P_l^0(\cos(\pi-\theta)) \sin\theta\, \mathrm{d}\theta \mathrm{d}\varphi \\ &= \sqrt{\pi(2l+1)} \int_0^{\pi} f_s(-\cos\theta) P_l^0(-\cos\theta) \sin\theta\, \mathrm{d}\theta\end{aligned}$$

Next substitution is applied using $u = -\cos\theta$ and $du = \sin\theta\, \mathrm{d}\theta$.

$$\begin{aligned}\rho_l &= \sqrt{\pi(2l+1)} \int_{-1}^{1} f_s(u) P_l^0(u) du \\ &= \sqrt{\pi(2l+1)} \int_0^1 u^s P_l^0(u) du \end{aligned} \tag{70}$$

The remaining problem is to solve the second term, which is the integral. The following formula is given by [Mac48] in equations 19 and 20 of chapter 5 and is valid for any real value $s \geq 0$.

$$\int_0^1 u^s P_l^0(u) du = \begin{cases} \frac{(s-1)(s-3)\ldots(s-l+2)}{(s+l+1)(s+l-1)\ldots(s+2)} & \text{when } l \text{ is odd} \\ \frac{s(s-2)\ldots(s-l+2)}{(s+l+1)(s+l-1)\ldots(s+1)} & \text{when } l \text{ is even} \end{cases}$$

Thus we now have a closed form which provides an exact evaluation of ρ_l. For a fixed number of coefficients we can precompute these $\{\rho_0, \rho_1, ..., \rho_{L-1}\}$. Alternatively, the following approximation when $s > l - 1$

$$\rho_l \simeq \sqrt{\pi(2l+1)}\frac{e^{-\frac{l^2}{2s}}}{s+1} \tag{71}$$

is given by Ravi Ramamoorthi [RH01]. However, because of the restriction that $s > l-1$, the approximation is unsuitable for the diffuse term, since in this case we have $s = 1$. The approximation is intended for high order specularity, so for the general case we use the exact form instead.

For a user–defined value s and by using equation (66) we are ready to define the partial sums of the given transfer function.

$$\begin{aligned} s_L(\rho)(R_{\vec{n}} \cdot \vec{\omega}) &= \sum_{l=0}^{L-1} \rho_l Y_l^0(R_{\vec{n}} \cdot \vec{\omega}) && (72) \\ s_L(\gamma)(\vec{n}) &= \sum_{l=0}^{L-1}\sum_{m=-l}^{l} \gamma_l^m Y_l^m(\vec{n}) && (73) \end{aligned}$$

Now the idea is to use equation (67), but there is a problem: Equation (72) is a composite function. According to [HSRG07] the result of the integration is

$$\int_{S^2} s_L(\rho)(R_{\vec{n}} \cdot \vec{\omega}) s_L(\gamma)(\vec{n})\,\mathrm{d}\vec{n} = \sum_{l=0}^{L-1}\sum_{m=-l}^{l} \sqrt{\frac{4\pi}{2l+1}}\rho_l \gamma_l^m Y_l^m(\vec{\omega}) \tag{74}$$

However, no proof is given. To remedy this, one will be given here, based on the *Spherical Harmonic Addition Theorem* which, for the real spherical harmonics, states: When ϕ is defined by

$$\cos\phi = \cos\theta_1 \cos\theta_2 + \sin\theta_1 \sin\theta_2 \cos(\varphi_1 - \varphi_2) \tag{75}$$

then the Legendre polynomial of argument ϕ is given by

$$P_l(\cos\phi) = \frac{4\pi}{2l+1}\sum_{m=-l}^{l} Y_l^m(\theta_1, \varphi_1) Y_l^m(\theta_2, \varphi_2) \tag{76}$$

Now as previously mentioned, let (θ_1, φ_1) represent the spherical coordinates for $\vec{n}$ and let (θ_2, φ_2) represent $\vec{\omega}$. Using equation (69), we can express the dot product between the two vectors as

$$\begin{aligned} \vec{n} \bullet \vec{\omega} &= \vec{S}(\theta_1, \varphi_1) \bullet \vec{S}(\theta_2, \varphi_2) \\ &= \cos(\theta_1)\cos(\theta_2) + \sin(\theta_1)\sin(\theta_2)\left(\cos(\varphi_1)\cos(\varphi_2) + \sin(\varphi_1)\sin(\varphi_2)\right) \\ &= \cos(\theta_1)\cos(\theta_2) + \sin(\theta_1)\sin(\theta_2)\left(\cos(\varphi_1)\cos(-\varphi_2) - \sin(\varphi_1)\sin(-\varphi_2)\right) \\ &= \cos(\theta_1)\cos(\theta_2) + \sin(\theta_1)\sin(\theta_2)\cos(\varphi_1 - \varphi_2) \end{aligned}$$

Since this corresponds directly to the initial criteria given in equation (75), we can consider ϕ the angle between $\vec{n}$ and $\vec{\omega}$. As previously mentioned, (θ_2', φ_2') represents the spherical coordinates of $R_{\vec{n}} \cdot \vec{\omega}$ and since $\vec{n}$ in this space is $(0, 0, 1)$ we know that θ_2' is the angle between $\vec{n}$ and $\vec{\omega}$ which tells us that $\theta_2' = \phi$.

$$\begin{aligned} s_L(\rho)(R_{\vec{n}} \cdot \vec{\omega}) &= \sum_{l=0}^{L-1} \rho_l Y_l^0(R_{\vec{n}} \cdot \vec{\omega}) \\ &= \sum_{l=0}^{L-1} \rho_l Y_l^0(\phi, \varphi_2') \\ &= \sum_{l=0}^{L-1} \rho_l \frac{K_l^0}{\sqrt{2}} P_l^0(\cos\phi) \\ &= \sum_{l=0}^{L-1} \rho_l \sqrt{\frac{2l+1}{4\pi}} \frac{4\pi}{2l+1} \sum_{m=-l}^{l} Y_l^m(\theta_1, \varphi_1) Y_l^m(\theta_2, \varphi_2) \\ &= \sum_{l=0}^{L-1} \sum_{m=-l}^{l} \rho_l \sqrt{\frac{4\pi}{2l+1}} Y_l^m(\vec{n}) Y_l^m(\vec{\omega}) \end{aligned} \tag{77}$$

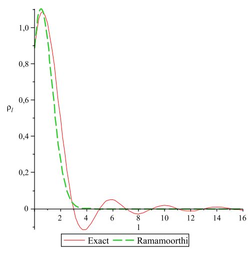
The value $Y_l^m(\vec{\omega})$ is constant under integration over $\vec{n}$ so given what we have just derived, we can consider $\rho_l \sqrt{\frac{4\pi}{2l+1}} Y_l^m(\vec{\omega})$ the coefficients of $s_L(\rho)(R_{\vec{n}} \cdot \vec{\omega})$. Now using this form for $s_L(\rho)(R_{\vec{n}} \cdot \vec{\omega})$ and still using equation (73) for γ, through insertion into equation (67), we obtain the result in equation (74).

The only thing that remains is evaluation of γ_l^m. For the full resolution normal map this is trivial since each texel represents exactly one normal. Let $\vec{n}_0$ denote the normal of some arbitrary texel, then at this texel $\gamma(\vec{n}) = \delta(\vec{n} - \vec{n}_0)$ is the NDF. By insertion of γ into equation (65) we obtain

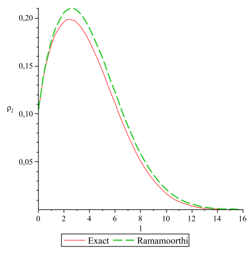
$$\begin{aligned} \gamma_l^m &= \int_{S^2} Y_l^m(\vec{n}) \delta(\vec{n} - \vec{n}_0) \, \mathrm{d}\vec{n} \\ &= Y_l^m(\vec{n}_0) \end{aligned}$$

which is the spherical harmonic basis function sampled at $\vec{n}_0$. A texel of any mip map level (which is not of full resolution) represents 2×2 texels of the mip map level one iteration above. By averaging the coefficients for γ_l^m of these four texels, we obtain the values for γ_l^m in the current iteration. The fact that we can do this is a direct result of equation (74) since filtering the left side of the equation for multiple NDFs is identical to filtering the coefficients γ_l^m on the right side for the given NDFs. So unlike the normals, the coefficients of the NDFs can be filtered. This procedure is intended as an offline process.

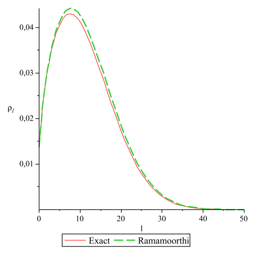
As previously mentioned in this section, ρ_l does not depend on m so only L coefficients are needed. The NDF, on the other hand, depends on both, so it requires L^2 coefficients. For high frequency reflection models, many terms are needed, specifically since the frequency of Y_l^0 is determined by $P_l^0(u)$, and since $P_l^0(u)$ is a polynomial of degree l. This suggests that an approximation of equation (68) requires L to be at least as big as s. In practice however, it turns out that this is not the case. A comparison of the exact and the approximate evaluation of ρ_l for $s = 1$, $s = 16$ and $s = 128$ is shown in figures 35(a)-35(c). Notice in figure 35(c) how the coefficients tend to zero already at $l \geq 35$. Overall,



(a)



(b)



(c)

Figure 35: The exact and the approximate evaluation of ρ_l for $s = 1$, $s = 16$ and $s = 128$ is shown in 35(a)-35(c) respectively. The exact evaluation is given in equation (70) and the approximate is equation (71).

the figures suggest that the approximation by Ramamoorthi works well, and as expected, in particular when s is large. Ramamoorthi uses the exponential part of his approximation to predict when to discard coefficients. Choosing e^{-2} as the threshold leads to

$$\begin{aligned} e^{-\frac{l^2}{2s}} &< e^{-2} \\ &\Leftrightarrow \\ -\frac{l^2}{2s} &< -2 \\ &\Leftrightarrow \\ l &> \sqrt{4s} \end{aligned}$$

which is exactly the cut-off suggested in [HSRG07]. Nevertheless, this still leaves us with an unrealistic footprint of the NDF when used with high frequency reflection models. For instance, for $s = 128$ we get

$$\begin{aligned} L &= \lfloor\sqrt{4 \cdot 128}\rfloor + 1 \\ &= 23 \end{aligned}$$

which results in $L^2 = 529$ coefficients for the NDF. A possible solution to this problem is suggested in [HSRG07] and will be described in section 4.2.

4.2 Spherical Harmonic clustering

Spherical harmonics are a suitable basis for representing low-frequency functions, but are impractical for higher-frequency functions due to the large number of coefficients required. For some NDFs, normals tend to gather in a limited set of clusters. This is typically the case when the normal map represents piecewise approximately planar surfaces. For such cases, we can resort to basis functions which are radially symmetric about some central direction $\vec{\mu}$. For one such cluster, the effective BRDF corresponding to equation (62), is thus rewritten to the current form.

$$\rho^{eff}(\vec{\omega}; \{\gamma, \vec{\mu}\}) = \int_{S^2} \rho(R_{\vec{n}} \cdot \vec{\omega}) \cdot \gamma(R_{\vec{n}} \cdot \vec{\mu}) \, \mathrm{d}\vec{n} \tag{78}$$

Since γ is now defined to be radially symmetric about $\vec{\mu}$, it entirely depends on the angle between $\vec{n}$ and $\vec{\mu}$. Thus, we can use the spherical harmonic addition formula the same way we did in equation (77) to derive the partial sums of γ.

$$s_L(\gamma)(R_{\vec{n}} \cdot \vec{\mu}) = \sum_{l=0}^{L-1} \sum_{m=-l}^{l} \gamma_l \sqrt{\frac{4\pi}{2l+1}} Y_l^m(\vec{n}) Y_l^m(\vec{\mu}) \tag{79}$$

Next, let ϕ denote the angle between $\vec{\mu}$ and $\vec{\omega}$, subsequently, using the partial sums given by equations (77) and (79) we can approximate equation (78) using equation (67) which will give us

$$\begin{aligned} \int_{S^2} s_L(\rho)(R_{\vec{n}} \cdot \vec{\omega}) \cdot s_L(\gamma)(R_{\vec{n}} \cdot \vec{\mu}) \, \mathrm{d}\vec{n} &= \sum_{l=0}^{L-1} \rho_l \gamma_l \frac{4\pi}{2l+1} \sum_{m=-l}^{l} Y_l^m(\vec{\omega}) Y_l^m(\vec{\mu}) \\ &= \sum_{l=0}^{L-1} \rho_l \gamma_l P_l(\cos\phi) \\ &= \sum_{l=0}^{L-1} \rho_l \gamma_l \sqrt{\frac{4\pi}{2l+1}} Y_l^0(\phi, 0) \end{aligned} \tag{80}$$

Here we have used equation (76) to remove the summation over m. The relevant thing to acknowledge here is that the coefficients of γ are no longer double indexed which reduces the footprint significantly. In practice however, it is unrealistic to rely on a single cluster. So let us assume the NDF can be approximated by $J \in \mathbf{N}$ clusters and let $j \in \{1, 2, ..., J\}$ be the cluster index. Subsequently, the approximation for equation (78) becomes.

$$\rho^{eff}(\vec{\omega}; \{\gamma_j, \vec{\mu}_j\}_{j=1}^{J}) \simeq \sum_{j=1}^{J} \sum_{l=0}^{L-1} \rho_l \gamma_{l,j} \sqrt{\frac{4\pi}{2l+1}} Y_l^0(\phi_j, 0) \tag{81}$$

Where ϕ_j denotes the angle between $\vec{\omega}$ and each direction $\vec{\mu}_j$. Notice that the coefficients of γ in equation (81) have once again become double indexed since the set of coefficients are specific to each cluster. The difference however is that unlike m, the range of j does not depend on l and subsequently L, so instead of having L^2 coefficients we now have $J \cdot L$. Another key difference is that J does not depend on the frequency of the reflection model. For NDFs, that can be approximated by a small amount of clusters, this makes a significant reduction in the amount of coefficients needed. Nevertheless, we shall see in the next section how a careful choice of basis function leads to a footprint of only J entries.

4.3 von Mises–Fisher clustering

In order to encapsulate clusters of normals, a Gaussian-like distribution on the unit sphere known as vMF or *von Mises–Fisher* is chosen. Such a distribution is based on the function $e^{\kappa(\vec{n}\bullet\vec{\mu})}$ where κ is the reciprocal width and $\vec{\mu}$ is the central direction. As required by a probability distribution function, it must integrate to 1. Note, the NDF $\gamma(\vec{n})$ integrated over the sphere is also 1. To normalize, the function is divided by

$$\begin{aligned}\int_{S^2} e^{\kappa(\vec{n}\bullet\vec{\mu})}\,\mathrm{d}\vec{n} &= \int_0^{2\pi}\int_0^{\pi} e^{\kappa(\cos(\pi-\theta))}\sin\theta\,\mathrm{d}\theta\mathrm{d}\varphi \\ &= 2\pi\int_0^{\pi} e^{\kappa(-\cos\theta)}\sin\theta\,\mathrm{d}\theta\end{aligned}$$

Next, substitution is applied using $u = -\cos\theta$ and $du = \sin\theta\,\mathrm{d}\theta$.

$$\begin{aligned}2\pi\int_{-1}^{1} e^{\kappa u}du &= \frac{4\pi}{\kappa}\frac{e^{\kappa}-e^{-\kappa}}{2} \\ &= \frac{4\pi}{\kappa}\sinh\kappa\end{aligned}$$

So finally we have

$$\mathcal{N}(\vec{n};\{\vec{\mu},\kappa\}) = \frac{\kappa}{4\pi\sinh\kappa}e^{\kappa(\vec{n}\bullet\vec{\mu})} \tag{82}$$

In order to encapsulate some limited number of clusters of normals (see figure 36) we must support a *mixture* of vMFs, that is some linear combination of vMFs such that the coefficients $\alpha_j > 0$ obey $\sum_{j=1}^{J}\alpha_j = 1$. Let Θ store the parameters $\{\alpha_j,\kappa_j,\mu_j\}_{j=1}^{J}$, the NDF is thus given as

$$\gamma(\vec{n};\Theta) = \sum_{j=1}^{J}\alpha_j\mathcal{N}(\vec{n};\{\vec{\mu}_j,\kappa_j\}) \tag{83}$$

Note that equation (83) still integrates to one given that $\sum_{j=1}^{J}\alpha_j = 1$. In order to use equation (83) with equation (81), the contribution for each j must be projected onto Y_l^0. Given equation (65), projection is achieved by the following equation

$$\gamma_{l,j} = \alpha_j\int_{S^2} Y_l^0(R_{\vec{n}}\cdot\vec{\mu}_j)\frac{\kappa_j}{4\pi\sinh\kappa_j}e^{\kappa_j(\vec{n}\bullet\vec{\mu}_j)}\,\mathrm{d}\vec{n}$$

Since the form is the same for all, we can omit the j for now and focus on the projection of a single vMF. Solving this integral is quite difficult and it is

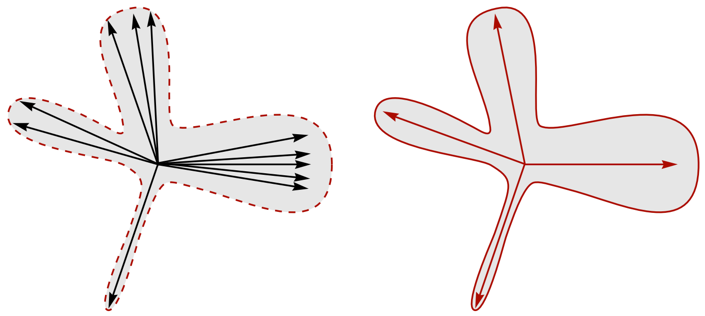


Figure 36: A distribution of normals is shown in 36(a) and a representation of an optimal fitting of a mixture of vMF distributions is shown in 36(b). In this case, four vMF distributions.

only solved in [HSRG07] through a series of approximations. Given a prediction made in [HSRG07] that the vMFs will, generally, have a thickness greater than zero and smaller than half, κ is assumed to be greater than two. From this, the first approximation is applied, that $\sinh\kappa \simeq \frac{e^{\kappa}}{2}$ which leads to

$$\begin{aligned} \gamma(\vec{n};\{\vec{\mu};\kappa\}) &= \frac{\kappa}{4\pi\sinh\kappa} e^{\kappa(\vec{n}\bullet\vec{\mu})} && (84) \\ &\simeq \frac{\kappa}{2\pi} e^{-\kappa(1-\vec{n}\bullet\mu)} && (85) \end{aligned}$$

Let β be the angle between $\vec{n}$ and $\vec{\mu}$ such that $1-\vec{n}\bullet\vec{\mu} = 1-\cos\beta$. Next, it is argued in [HSRG07] that for moderate κ, β must be small for the exponential to be nonzero, in such cases $1-\cos\beta \simeq \frac{\beta^2}{2}$ and from this it follows

$$\gamma(\vec{n};\{\vec{\mu};\kappa\}) \simeq \frac{\kappa}{2\pi} e^{-\kappa\frac{\beta^2}{2}} \qquad (86)$$

Note that $\frac{\beta^2}{2}$ is really just a second–order Taylor approximation of $1-\cos\beta$ at $\beta = 0$ (see figure 37). An approximation for projection of equation (86) is given in [RH01] and is used by [HSRG07]. Since the derivation of the approximation is rather evolved, it will not be described here but the reader is referred to the appendix of [RH01] for details. The approximation leads to the following coefficients

$$\gamma_l \simeq \sqrt{\frac{2l+1}{4\pi}} e^{-\frac{l^2}{2\kappa}}$$

Let a variant s' of the user–defined value s, used in conjunction with the transfer function, be given as

$$s' = \frac{s\kappa}{\kappa+s}$$

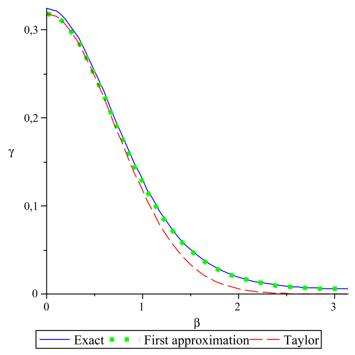


Figure 37: A comparison between the three curves given by: the exact equation (82), the initial approximation (85) and finally using the additional second–order Taylor approximation (86) is shown here. The variable β corresponds to the angle between $\vec{n}$ and $\vec{\mu}$ and κ has been set to its minimum expected value, which is 2.

Next, by using the approximation (71) of Ramamoorthi and thus inserting it and our expression for γ_l into equation (80), we obtain the following simplification

$$\begin{aligned}
\int_{S^2} s_L(\rho)(R_{\vec{n}} \cdot \vec{\omega}) \cdot s_L(\gamma)(R_{\vec{n}} \cdot \vec{\mu}) \, \mathrm{d}\vec{n} &= \sum_{l=0}^{L-1} \rho_l \gamma_l \sqrt{\frac{4\pi}{2l+1}} Y_l^0(\phi, 0) \\
&\simeq \sum_{l=0}^{L-1} \sqrt{\pi(2l+1)} \frac{e^{-\frac{l^2}{2s}}}{s+1} e^{-\frac{l^2}{2\kappa}} Y_l^0(\phi, 0) \\
&= \sum_{l=0}^{L-1} \frac{\sqrt{\pi(2l+1)}}{s+1} e^{-\frac{l^2}{2\left(\frac{s\kappa}{\kappa+s}\right)}} Y_l^0(\phi, 0) \\
&= \frac{s'+1}{s+1} \sum_{l=0}^{L-1} \sqrt{\pi(2l+1)} \frac{e^{-\frac{l^2}{2s'}}}{s'+1} Y_l^0(\phi, 0) \\
&\simeq \frac{s'+1}{s+1} \max(\cos\phi, 0)^{s'}
\end{aligned}$$

As we see here, the summation over l has been optimized away, which leads to a reduction of the footprint as well as the execution time. To support the full vMF mixture, the summation over j, is reintroduced and thus, the following is obtained

$$
\begin{aligned}
s'_j &= \frac{s\kappa_j}{\kappa_j + s} && (87)\\
\rho^{eff}(\vec{\omega};\Theta) &\simeq \sum_{j=1}^{J} \alpha_j \frac{s'_j + 1}{s+1} \max(\vec{\omega} \bullet \vec{\mu}_j, 0)^{s'_j} && (88)
\end{aligned}
$$

An intuitive interpretation of what has been achieved here by [HSRG07], is to apply a form of vector quantization on each NDF where the influence of each resulting vector $\vec{\mu}_j$ is specified by α_j and κ_j. Specifically, if κ_j is small, then s'_j is small which, given equation 88, will flatten the specular high–light. If on the other hand κ_j is large, then s'_j approaches the original specular power s. This makes sense since clusters which are thick have a small κ_j and those which are thin, and thus reflect light in a uniform direction, have a large κ_j.
For this technique to be successful, the normals of the normal map must gather in a small amount of clusters. Additionally, the technique covered in this section implies the presence of a high frequency reflection model. The fact that [HSRG07] uses equation (71) confirms this statement since, as explained in section 4.1, the approximation is based on the presence of a large value for s such that $s > l$. In [HSRG07] the technique is used for $s > 12$.

In section 4.1, it was shown that the coefficients for γ can be filtered. A problem with equation (88) is that these coefficients have been optimized away. Technically, it is not correct to filter $\{\alpha, \vec{\mu}, \kappa\}$ as it is for the coefficients of γ so the most accurate solution is to filter the results of equation (88). It is pointed out in [HSRG07], that the most hardware efficient solution, is to ignore this technicality and simply filter $\{\alpha, \vec{\mu}, \kappa\}$ anyway and then execute equation (88) on the filtered input data. However, doing so requires arranging the J vMFs in a consistent ordering such that filtering is performed between vMFs which are similar in orientation and appearance. The solution suggested in [HSRG07] will be discussed in section 4.4.5.

This section up until now has assumed that the parameters of the mixture stored in Θ are already known. In practice, however, an algorithm is needed to find a suitable choice for Θ, in [HSRG07] a method known as spherical EM or *Expectation Maximization* is used. This technique will be covered in section 4.4.

4.4 Spherical Expectation Maximization

The EM algorithm [DLR77] is commonly used in statistics for fitting a distribution model onto a set of data by finding maximum likelihood estimates. It is an iterative method, with each iteration consisting of two steps known as the E-step and the M-step. The more recent work by [BDGS05] on Spherical EM is an analysis of the EM algorithm applied to a set of directional data using a mixture of vMFs as a distribution model specifically.

The form of the probability distribution function γ is given in equation (83). The NDF is determined from some finite selection of normals $\{\vec{n}_1, \vec{n}_2, ..., \vec{n}_I\}$

where $I \in \mathbf{N}$ and the normals are assumed to be independent. The best choice for Θ is the one that maximizes the likelihood of drawing from the distribution γ, exactly the selection of I normals which we are given. This is also known as *maximum likelihood estimation* and can be written the following way

$$\hat{\Theta} = \arg\max_{\theta} \gamma(\{\vec{n}_i\}_{i=1}^{I} | \Theta)$$

Let X and Y be two random variables, in probability theory, the syntax $p(X, Y)$ denotes the *joint probability* and obeys the symmetry property $p(X, Y) = p(Y, X)$. Furthermore, the following two equations are rules of probability.

$$\begin{aligned} p(X) &= \sum_{Y} p(X, Y) \\ p(X, Y) &= p(Y|X)p(X) \end{aligned}$$

These are known as the *sum rule* and the *product rule* respectively. Additionally, when X and Y are independent, then $p(Y|X) = p(Y)$ and so from the product rule it follows that $p(X, Y) = p(X)p(Y)$. Since the normals are independent, the maximization of Θ can be rewritten

$$\begin{aligned} \hat{\Theta} &= \arg\max_{\theta} \gamma(\{\vec{n}_i\}_{i=1}^{I} | \Theta) \\ &= \arg\max_{\theta} \prod_{i=1}^{I} \gamma(\vec{n}_i | \Theta) \qquad (89) \\ &= \arg\max_{\theta} \sum_{i=1}^{I} \ln \gamma(\vec{n}_i | \Theta) \qquad (90) \end{aligned}$$

However, since a mixture γ, is given in the form of a summation, finding a maximum for equation 89 or even 90 is, generally speaking, not easy. For this reason, the EM algorithm is based on a study of a related problem, that is what if for every observation $\vec{n}_i$ we had a corresponding variable $z_i \in \{1, 2, ..., J\}$ which tells us which cluster $\vec{n}_i$ belongs to.

$$\begin{aligned} p(z = j) &= \alpha_j \\ p(\vec{n}|z = j) &= \mathcal{N}(\vec{n}; \{\vec{\mu}_j, \kappa_j\}) \end{aligned}$$

From this we can write the joint probability the following way

$$\begin{aligned}
p(\{\vec{n}_i\}_{i=1}^{I}, \{\vec{z}_i\}_{i=1}^{I}|\Theta) &= p(\{\vec{n}_1, z_1\}, \{\vec{n}_2, z_2\}, ..., \{\vec{n}_I, z_I\}|\Theta) \\
&= \prod_{i=1}^{I} p(\vec{n}_i, z_i|\Theta) \\
&= \prod_{i=1}^{I} p(\vec{n}_i|\{z_i, \Theta\})p(z_i|\Theta) \\
&= \prod_{i=1}^{I}\prod_{j=1}^{J} p(\vec{n}_i|\{z_i = j, \mu_j, \kappa_j\})^{1_j(z_i)} p(z_i = j|\alpha_j)^{1_j(z_i)} \\
&= \prod_{i=1}^{I}\prod_{j=1}^{J} (\alpha_j \mathcal{N}(\vec{n}_i; \{\vec{\mu}_j, \kappa_j\}))^{1_j(z_i)}
\end{aligned}$$

And if we apply the natural logarithm to both sides, we get the following simpler version.

$$\ln p(\{\vec{n}_i\}_{i=1}^{I}, \{\vec{z}_i\}_{i=1}^{I}|\Theta) = \sum_{i=1}^{I}\sum_{j=1}^{J} 1_j(z_i)\left(\ln \mathcal{N}(\vec{n}_i; \{\vec{\mu}_j, \kappa_j\}) + \ln \alpha_j\right) \qquad (91)$$

This looks promising, α_j has been isolated and can therefore be maximized independently from $\vec{\mu}_j$ and κ_j. Additionally, since every normal $\vec{n}_i$ is tied to a specific cluster, this gives us exactly J disjoint, nonempty, subsets of $\{\vec{n}_i\}_{i=1}^{I}$, one for each vMF. Subsequently, each vMF can be maximized independently of the other $J - 1$ vMFs. There is only one problem and unfortunately it is a deal–breaker: We do not actually have the answers to $1_j(z_i)$. The EM algorithm solves this problem by using the expectation, denoted $\mathbb{E}_z(\ln p(\{\vec{n}_i\}_{i=1}^{I}, \{\vec{z}_i\}_{i=1}^{I}|\Theta))$, which is also the mean with respect to the distribution of z. Additionally, the principal of *conditional expectation* with respect to a conditional distribution is specified by equation (1.37) of [Bis07] and reads

$$\mathbb{E}_x(f|y) = \sum_{x} p(x|y)f(x)$$

for some arbitrary function f of x. Subsequently, the mean is

$$\begin{aligned}
\mathbb{E}_z(\ln p(\{\vec{n}_i\}_{i=1}^I, \{\vec{z}_i\}_{i=1}^I|\Theta)) &= \sum_{i=1}^{I} \mathbb{E}_z(\ln p(\vec{n}_i, z_i|\Theta)) \\
&= \sum_{i=1}^{I}\sum_{j=1}^{J} p(z_i = j|\vec{n}_i) \ln p(\vec{n}_i, z_i = j|\{\alpha_j, \mu_j, \kappa_j\}) \\
&= \sum_{i=1}^{I}\sum_{j=1}^{J} p(z_i = j|\vec{n}_i) \ln \left(p(\vec{n}_i|\{z_i = j, \mu_j, \kappa_j\}) \cdot p(z_i = j|\alpha_j) \right) \\
&= \sum_{i=1}^{I}\sum_{j=1}^{J} p(z_i = j|\vec{n}_i) \left(\ln p(\vec{n}_i|\{z_i = j, \mu_j, \kappa_j\}) + \ln p(z_i = j|\alpha_j) \right) \\
&= \sum_{i=1}^{I}\sum_{j=1}^{J} p(z_i = j|\vec{n}_i) \left(\ln \mathcal{N}(\vec{n}_i; \{\vec{\mu}_j, \kappa_j\}) + \ln \alpha_j \right) \qquad (92)
\end{aligned}$$

Notice how equation (92) is very similar to equation (91). The term $1_j(z_i)$ has now been replaced by $p(z_i = j|\vec{n}_i)$. Unfortunately, we do not have the values for this term either:

$$c_{ij} = p(z_i = j|\vec{n}_i) \qquad (93)$$

The EM method is based on two steps, the first is the E-step and its purpose is to provide an estimation for the coefficients c_{ij}. The second is the M-step and its purpose is to maximize with respect to Θ the following rewrite of equation (92)

$$\mathbb{E}_z(\ln p(\{\vec{n}_i\}_{i=1}^I, \{\vec{z}_i\}_{i=1}^I|\Theta)) \simeq \sum_{i=1}^{I}\sum_{j=1}^{J} c_{ij} \ln \mathcal{N}(\vec{n}_i; \{\vec{\mu}_j, \kappa_j\}) + \sum_{i=1}^{I}\sum_{j=1}^{J} c_{ij} \ln \alpha_j \qquad (94)$$

where the coefficients c_{ij} are interpreted as constants during maximization. After the M-step, the coefficients c_{ij} are reevaluated again, and thus refined, in a subsequent E-step. Next, the M-step follows and so on. This process forms an iterative approach which is to be terminated once equation (94) no longer increases during maximization or if the amount of iterations which have taken place exceeds some user–defined maximum. Intuitively, the estimates for c_{ij} and for $\hat{\Theta}$ are improved for every iteration and eventually convergence takes place at some local maximum. For a proof of convergence, the reader is referred to section (9.4) of [Bis07].

4.4.1 The expectation step

To explain how the E-step provides an estimate of the coefficients c_{ij}, based on the current iteration of $\hat{\Theta}$, consider once again the random variables X and Y. From the symmetry property and the product rule it follows that

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}$$

which is also known as *Bayes' theorem.* By using the sum rule, the product rule and the symmetry property on the denominator of Bayes' theorem we get

$$p(Y|X) = \frac{p(X|Y)p(Y)}{\sum_Y p(X|Y)p(Y)} \tag{95}$$

We can think of the denominator in equation (95) as being the normalization constant required to ensure that the sum of the conditional property on the left-hand side over all values of Y equals one. Using equation (95) on equation (93) specifically yields

$$\begin{aligned} c_{ij} &= p(z=j|\vec{n}_i) \\ &= \frac{p(z=j)p(\vec{n}_i|z=j)}{\sum_{k=1}^{J} p(z=k)p(\vec{n}_i|z=k)} \\ &= \frac{\alpha_j \mathcal{N}(\vec{n}_i;\{\vec{\mu}_j,\kappa_j\})}{\sum_{k=1}^{J} \alpha_k \mathcal{N}(\vec{n}_i;\{\vec{\mu}_k,\kappa_k\})} \end{aligned} \tag{96}$$

The most recent iteration of $\hat{\Theta}$, which is refined in every M-step, is used to evaluate equation (96) which completes the E-step. As mentioned, in regards to equation (95), summation over all Y equals one. Similarly for equation (96) we have

$$\sum_{j=1}^{J} c_{ij} = 1 \tag{97}$$

Maximization of equation (94) is covered in section 4.4.2.

4.4.2 The maximization step

Initially, the parameter set on the right side of equation (94) is maximized, that is $\{\alpha_j\}_{j=1}^{J}$. Let

$$f(\{\alpha_j\}_{j=1}^{J}) = \sum_{i=1}^{I}\sum_{j=1}^{J} c_{ij} \ln \alpha_j$$

The function f is maximized by evaluating the derivative of f with respect to every α_j and thus solving $\frac{df}{d\alpha_j} = 0$ for all J equations. Note however that the constraint in equation (97) needs to be taken into account. This is done by introducing the following function

$$\begin{aligned} g(\{\alpha_j\}_{j=1}^{J}) &= -1 + \sum_{j=1}^{J} \alpha_j \\ &= 0 \end{aligned}$$

and the Lagrange multiplier λ. Instead, differentiation is performed on $f + \lambda g$, with respect to α_h for $h \in \{1, 2, ..., J\}$.

$$\begin{aligned}\frac{d(f+\lambda g)(\{\alpha_j\}_{j=1}^{J})}{d\alpha_h} &= \frac{d}{d\alpha_h}\sum_{i=1}^{I}\sum_{j=1}^{J} c_{ij}\ln\alpha_j + \lambda\left(-1+\sum_{j=1}^{J}\alpha_j\right) \\ &= \lambda + \frac{1}{\alpha_h}\sum_{i=1}^{I} c_{ih}\end{aligned}$$

Next, $\frac{d(f+\lambda g)}{d\alpha_h} = 0$ is solved for all J equations. For additional detail on the concept of maximizing a function given a constraint, the reader is referred to pp. 721–728 in [Ada94].

$$\begin{aligned}\frac{d(f+\lambda g)(\{\alpha_j\}_{j=1}^{J})}{d\alpha_h} &= 0 \\ &\Leftrightarrow \\ \lambda + \frac{1}{\alpha_h}\sum_{i=1}^{I} c_{ih} &= 0 \qquad (98)\end{aligned}$$

Next both sides of equation (98) are multiplied by α_h, which leads to the equation

$$\sum_{i=1}^{I} c_{ih} = -\lambda\alpha_h \qquad (99)$$

Since this equation applies to any chosen $h \in \{1, 2, ..., J\}$ summation over j can be applied on both sides.

$$\begin{aligned}\sum_{j=1}^{J}\sum_{i=1}^{I} c_{ij} &= \sum_{j=1}^{J} -\lambda\alpha_j \\ &\Leftrightarrow \\ \sum_{i=1}^{I}\sum_{j=1}^{J} c_{ij} &= -\lambda\sum_{j=1}^{J}\alpha_j \\ &\Leftrightarrow \\ \sum_{i=1}^{I} 1 &= -\lambda \\ &\Leftrightarrow \\ \lambda &= -I\end{aligned}$$

Finally, substitution of $\lambda = -I$ back into (99) provides the result,

$$\alpha_h = \frac{1}{I}\sum_{i=1}^{I} c_{ih}$$

which completes maximization of the set $\{\alpha_j\}_{j=1}^{J}$, given the estimate for c_{ij} provided by the E-step. Next, the other half of equation (94) is maximized in a similar fashion.

$$\begin{aligned} f(\{\vec{n}_i\}_{i=1}^{I}, \{\mu_j, \kappa_j, \lambda_j\}_{j=1}^{J}) &= \sum_{i=1}^{I}\sum_{j=1}^{J} c_{ij} \ln \mathcal{N}(\vec{n}_i; \{\vec{\mu}_j, \kappa_j\}) \\ &= \sum_{i=1}^{I}\sum_{j=1}^{J} c_{ij} \left(\kappa_j \cdot (\vec{n}_i \bullet \vec{\mu}_j) + \ln \kappa_j - \ln(4\pi) - \ln(\sinh \kappa_j)\right) \\ g(\{\vec{\mu}_j, \lambda_j\}_{j=1}^{J}) &= \sum_{j=1}^{J} \lambda_j \left(1 - \vec{\mu}_j \bullet \vec{\mu}_j\right) \\ &= 0 \end{aligned}$$

Again, f is maximized given the constraint $g = 0$, initially the derivatives are evaluated.

$$\frac{d(f+g)(\{\vec{n}_i\}_{i=1}^{I}, \{\mu_j, \kappa_j, \lambda_j\}_{j=1}^{J})}{d\mu_h} = -2\lambda_h \vec{\mu}_h + \kappa_h \sum_{i=1}^{I} c_{ih} \cdot \vec{n}_i \tag{100}$$

$$\frac{d(f+g)(\{\vec{n}_i\}_{i=1}^{I}, \{\mu_j, \kappa_j, \lambda_j\}_{j=1}^{J})}{d\lambda_h} = 1 - \vec{\mu}_h \bullet \vec{\mu}_h$$

$$\frac{d(f+g)(\{\vec{n}_i\}_{i=1}^{I}, \{\mu_j, \kappa_j, \lambda_j\}_{j=1}^{J})}{d\kappa_h} = \sum_{i=1}^{I} c_{ih} \cdot \left(\vec{n}_i \bullet \vec{\mu}_h + \frac{1}{\kappa_h} - \coth \kappa_h\right)$$

By setting these derivatives equal to zero the following three equations are obtained.

$$\vec{\mu}_h = \frac{\kappa_h}{2\lambda_h} \sum_{i=1}^{I} c_{ih} \cdot \vec{n}_i \tag{101}$$

$$\vec{\mu}_h \bullet \vec{\mu}_h = 1 \tag{102}$$

$$\left(\coth \kappa_h - \frac{1}{\kappa_h}\right) \sum_{i=1}^{I} c_{ih} = \vec{\mu}_h \bullet \sum_{i=1}^{I} c_{ih} \vec{n}_i \tag{103}$$

Now given equation (102), if $\vec{\mu}_h$ is applied as a dot product on both sides of equation (101), this leads to

$$\lambda_h = \frac{\kappa_h}{2} \sum_{i=1}^{I} c_{ih} \cdot (\vec{\mu}_h \bullet \vec{n}_i)$$

Since $\kappa_h > 0$ and $c_{ih} \geq 0$, the sign of λ_h will depend on the outcome of the sum on the right. However, the objective is to maximize the probability of obtaining $\{\vec{n}_1, \vec{n}_2, ..., \vec{n}_I\}$ from the distribution and since c_{ih} increases (see eq. (82) and (96)) as the angle between $\vec{\mu}_h$ and $\vec{n}_i$ is overall diminished, which maximizes

$\vec{\mu}_h \bullet \vec{n}_i$, the value of λ_h is considered greater than zero. From this and equation (101), it follows after normalization that

$$\vec{\mu}_h = \frac{\sum_{i=1}^{I} c_{ih} \cdot \vec{n}_i}{\| \sum_{i=1}^{I} c_{ih} \cdot \vec{n}_i \|} \tag{104}$$

By substitution of this into equation (103) we get

$$\left(\coth \kappa_h - \frac{1}{\kappa_h} \right) \sum_{i=1}^{I} c_{ih} = \| \sum_{i=1}^{I} c_{ih} \vec{n}_i \| \tag{105}$$

Next, let $A(x) = \coth x - \frac{1}{x}$, it follows that the objective is to find the inverse such that

$$\kappa_h = A^{-1} \left(\frac{\| \sum_{i=1}^{I} c_{ih} \vec{n}_i \|}{\sum_{i=1}^{I} c_{ih}} \right)$$

Unfortunately, no closed form exists for A^{-1}, in [BDGS05] the following approximation

$$\begin{aligned} \vec{r}_h &= \frac{\sum_{i=1}^{I} c_{ih} \vec{n}_i}{\sum_{i=1}^{I} c_{ih}} \\ \kappa_h &\simeq \frac{3\|\vec{r}_h\| - \|\vec{r}_h\|^3}{1 - \|\vec{r}_h\|^2} \end{aligned}$$

is empirically found to provide good results and is also used in [HSRG07]. In conclusion, the M-step is achieved by the following equations:

$$\alpha_h = \frac{1}{I} \sum_{i=1}^{I} c_{ih} \tag{106}$$

$$\vec{r}_h = \frac{\sum_{i=1}^{I} c_{ih} \vec{n}_i}{\sum_{i=1}^{I} c_{ih}} \tag{107}$$

$$\vec{\mu}_h = \frac{r_h}{\|r_h\|} \tag{108}$$

$$\kappa_h \simeq \frac{3\|\vec{r}_h\| - \|\vec{r}_h\|^3}{1 - \|\vec{r}_h\|^2} \tag{109}$$

Given the mutual dependence between the E–step and the M–step, an initial rough estimate for $\hat{\Theta}$ must be provided before any iterations take place. However, since there is a risk of tracking down some arbitrary local extrema, it is important that this be a good estimate for the initial value. This will be discussed in section 4.4.3.

4.4.3 Initial estimate

The objective of the initialization step is to choose for every texel J vMFs such that the mixture is close to the best fitting onto the NDF of the texel. The

full resolution normal map has only one normal per texel so initialization is done, simply by setting $\vec{\mu} = \vec{n}$, $\alpha = 1$ and κ to an unspecified large value which corresponds to a thin vMF (as in [HSRG07]).
A texel of a mip map level represents 2×2 texels of the mip map level one iteration above. Initialization, in [HSRG07], is done by choosing J vMFs from the $4 \cdot J$ vMFs assigned to these 2×2 texels. The choice is made such that the shortest angular distance between any two vMF central directions is maximized. This is done using a heuristic known as Hochbaum Shmoys clustering [HS85]. Let the initial $4 \cdot J$ central directions be known as $V = \{\vec{\mu}_i\}_{i=1}^{I}$ where $I = 4 \cdot J$. The work in [HS85] operates on a complete graph $G = (V, E)$ where $E = \{e_1, e_2, ..., e_m\}$ with $m = |E|$. For every edge e_j and $j \in \{1, 2, ..., m\}$ there is an associated weight $w_{e_j} \geq 0$ which in [HSRG07] is the angular distance. The edge list E is assumed to be given ordered such that $w_{e_1} \leq w_{e_2} \leq ...w_{e_m}$. Let $G_k = (V, E_k)$ denote the subgraph of G where $E_k = \{e_1, e_2, ..., e_k\}$. The subgraph G_k thus only contains links between vertices with a weight less than or equal to w_{e_k}. Removing vertices in V connected to an entry $\vec{\mu}_i \in V$, through edges in E_k, results in $S \subseteq V$. We can consider S a trimmed version of V using the threshold w_{e_k}. Thus by iteratively performing trimming for entries of $\vec{\mu}_i \in V$, the result S only contains entries connected by edges with an associated weight, greater than or equal to w_{e_k}. Unfortunately, this alone does not provide $|S| = J$, so bisection over k is applied using 1 and m as the initial lower and upper index. This locates the k that brings $|S|$ as close to J as possible.
Let the syntax $\text{ADJ}_k(\vec{\mu}_i)$ denote the set of entries in V connected to $\vec{\mu}_i$ through edges in E_k. The algorithm in pseudo code is given below.

```
low ← 1
high ← $|E|$
**while** high > low + 1
    mid ← $\lfloor \frac{\text{low}+\text{high}}{2} \rfloor$
    $S \leftarrow \emptyset$
    $T \leftarrow V$
    **while** $\exists \vec{\mu} \in T$
        $S \leftarrow S \cup \{\vec{\mu}\}$
        **for all** $\vec{v} \in \text{ADJ}_{\text{mid}}(\vec{\mu})$
            $T \leftarrow T - \text{ADJ}_{\text{mid}}(\vec{v}) - \{\vec{v}\}$
    **if** $|S| \leq J$
        high ← mid
        $S' \leftarrow S$
    **else** low ← mid
**output** $S'$
```

Ideally, better initialization could be done by choosing J central directions based on the full coverage of a texel. However, since the algorithm operates on the complete graph associated with V, the edge list is $|E| = O(|V|^2)$ entries long. As an example, the smallest mip map level is only a single texel and so the coverage corresponds to every normal in the full resolution normal map. To minimize execution time significantly, the choice in [HSRG07] is made based on the $4 \cdot J$ vMFs assigned to the 2×2 texels previously described. The subsequent Spherical EM algorithm, however, operates on the full coverage

4.4.4 The algorithm

Preprocessing of the normal map is performed as follows: Every texel of each mip map corresponds to some axis–aligned coverage in the full resolution normal map. Let J be a user–defined integer constant. For each texel, a mixture of J vMFs is fitted to the distribution of normals in the coverage. This fitting is done using Spherical EM as described in the previous sections. First initialization is performed as explained in section 4.4.3. Next, the E–step and the M–step are performed in turn until convergence is reached. Convergence is reached once the probability of drawing the normals, in the coverage, from the mixture of vMFs is no longer significantly maximized or once a maximum number of iterations have been performed. The algorithm is given in pseudo–code below where I defines the amount of normals in the coverage of the texel currently being processed.

```
repeat
    % The E–step
    for all samples $\vec{n}_i$
        for $j = 1$ to $J$
            $c_{ij} \leftarrow \frac{\alpha_j \mathcal{N}(\vec{n}_i; \{\vec{\mu}_j, \kappa_j\})}{\sum_{k=1}^{J} \alpha_k \mathcal{N}(\vec{n}_i; \{\vec{\mu}_k, \kappa_k\})}$
    % The M–step
    for $j = 1$ to $J$
        $\alpha_j \leftarrow \frac{\sum_{i=1}^{I} c_{ij}}{I}$
        $\vec{r}_j \leftarrow \frac{\sum_{i=1}^{I} c_{ij} \vec{n}_i}{\sum_{i=1}^{I} c_{ij}}$
        $\kappa_j \leftarrow \frac{3\|\vec{r}_j\| - \|\vec{r}_j\|^3}{1 - \|\vec{r}_j\|^2}$
        $\vec{\mu}_j \leftarrow \frac{\vec{r}_j}{\|\vec{r}_j\|}$
until convergence
```

The assignment made to c_{ij} in the E–step is given by equation (96) and the four assignments in the M–step are given by equations (106), (107), (108) and (109).

It should be mentioned that in [HSRG07] the assignment in the E–step is given (without multiplication by α) as

$$c_{ij} \leftarrow \frac{\mathcal{N}(\vec{n}_i; \{\vec{\mu}_j, \kappa_j\})}{\sum_{k=1}^{J} \mathcal{N}(\vec{n}_i; \{\vec{\mu}_k, \kappa_k\})}$$

Since this does not correspond to equation (96), I have asked Charles Han regarding the matter and he has confirmed that it is an error in his paper.

4.4.5 Filtering the clusters

It was pointed out in section 4.3 that there are no coefficients to filter when performing lighting calculations using equation (88). The correct approach is to collect the mixtures of the texels sampled, use equation (88) on each mixture separately and finally filter the results. For trilinear filtering, this results in eight individual mixtures sampled and thus equation (88) is evaluated eight times before filtering takes place.

In order to make his work applicable to real-time rendering, though it is technically incorrect, Han proposes an alternative where mixtures are filtered instead of the results. The problem is that the order of the vMFs in the mixture is arbitrary. To filter the vMFs $(\vec{\mu}_j, \kappa_j, \alpha_j)$ there has to be a certain coherency and similarity between those assigned to neighboring texels at the same entry j. Han solves this problem by changing the probability which is maximized such that it takes results of the previously processed mip map level into account. This is explained in the following. Let the sets X and Y be given as

$$\begin{aligned} X &= \{\vec{n}_i\}_{i=1}^{I} \\ Z &= \{\vec{z}_i\}_{i=1}^{I} \end{aligned}$$

In section 4.4.2, fitting was done by maximizing equation (92) which is the maximization of $\mathbb{E}_z(\ln P(X, Z|\Theta))$. In order to achieve better coherency for filtering, Han replaces maximization of this probability with $P(\Theta, X, Z|N(\Theta))$ where $N(\Theta)$ is the cluster arguments of the 2×2 texels from the processed previous mip map level at the corresponding location. Let these four texels be referenced by $k \in \{1, 2, 3, 4\}$, the elements of $N(\Theta)$ are known and given by μ_{jk}, κ_{jk} and α_{jk}. Using the product rule, we can use the following expansion

$$P(\Theta, X, Z|N(\Theta)) = P(\Theta|N(\Theta)) \cdot P(X, Z|\Theta)$$

where the left term is chosen in [HSRG07] as a product of vMF distributions

$$P(\Theta|N(\Theta)) = \prod_{j=1}^{J} \prod_{k=1}^{K} \frac{\alpha_{jk} C'}{4\pi \sinh(\alpha_{jk} C')} e^{\alpha_{jk} C' (\vec{\mu}_j \bullet \vec{\mu}_{jk})} \tag{110}$$

Thus the reciprocal thickness of each vMF is given by $\alpha_{jk} C'$ where C' is a user-defined scale used to tweak the distribution and $K = 4$. The probability

$P(\Theta|N(\Theta))$ is independent of Z which leads to the expectation

$$\begin{aligned}\mathbb{E}_z(\ln P(\Theta, X, Z|N(\Theta))) &= \mathbb{E}_z(\ln P(\Theta|N(\Theta))) + \mathbb{E}_z(\ln P(X, Z|\Theta)) \\ &= \ln P(\Theta|N(\Theta)) + \mathbb{E}_z(\ln P(X, Z|\Theta)) \qquad (111)\end{aligned}$$

Maximization is applied to equation (111) instead of equation (92). Note that the second term in equation (111) is identical to equation (92). In section 4.4.2, maximization was done by evaluating the first–order derivatives with respect to μ_h, κ_h and α_h. The first term in equation (111) does not depend on κ_h or α_h so only the derivative with respect to μ_h needs to be reevaluated. The derivative of the first term with respect to μ_h, given equation (110), is

$$\frac{d \ln P(\Theta|N(\Theta))}{d\mu_h} = C' \sum_{k=1}^{K} \alpha_{hk} \vec{\mu}_{hk}$$

This is added to the derivative of the second term which is given by equation (100) and the result is subsequently set to zero to find the local extrema. This yields the following

$$\begin{aligned}0 &= -2\lambda_h \vec{\mu}_h + \kappa_h \sum_{i=1}^{I} c_{ih} \cdot \vec{n}_i + C' \sum_{k=1}^{K} \alpha_{hk} \vec{\mu}_{hk} \\ &\Leftrightarrow \\ 2\lambda_h \vec{\mu}_h &= \kappa_h \sum_{i=1}^{I} c_{ih} \cdot \vec{n}_i + C' \sum_{k=1}^{K} \alpha_{hk} \vec{\mu}_{hk} \qquad (112)\end{aligned}$$

As in section 4.4.2 $\lambda_h > 0$ so it follows

$$\vec{\mu}_h = \frac{\kappa_h \sum_{i=1}^{I} c_{ih} \cdot \vec{n}_i + C' \sum_{k=1}^{K} \alpha_{hk} \vec{\mu}_{hk}}{\|\kappa_h \sum_{i=1}^{I} c_{ih} \cdot \vec{n}_i + C' \sum_{k=1}^{K} \alpha_{hk} \vec{\mu}_{hk}\|} \qquad (113)$$

Equation (113) is also given in the appendix of [HSRG07]. At the end of the appendix, Han says that by introducing C where $C' = C \cdot \kappa_h$, equation (113) can be replaced by the following

$$\vec{\mu}_h = \frac{\vec{r}_h + C \sum_{k=1}^{K} \alpha_{hk} \vec{\mu}_{hk}}{\|\vec{r}_h + C \sum_{k=1}^{K} \alpha_{hk} \vec{\mu}_{hk}\|} \qquad (114)$$

However, given equation (107), substitution of $\sum_{i=1}^{I} c_{ih} \cdot \vec{n}_i$ with $\vec{r}_h$ implies a division by $\sum_{i=1}^{I} c_{ih}$ which given equation (106) is the same as $I \cdot \alpha_h$. Since division must be applied to both terms, this substitution is only valid if C is defined such that

$$C' = C \cdot I \cdot \kappa_h \cdot \alpha_h \qquad (115)$$

I have contacted Han and he agrees that $C' = C \cdot \kappa_h$ is a mistake in his paper. During this correspondence I also asked which constant value he uses for C' in his implementation. Surprizingly, Han told me, he sets C directly to the constant 2 in equation (114). Given equation (115), interpreting C as constant means C' is no longer a constant and makes it depend on I, α_h and κ_h. This changes $P(\Theta|N(\Theta))$ given in equation (110) and subsequently invalidates the result, obtained by maximization, given in the appendix of [HSRG07] (and equation (113) of this thesis). Han agrees this is an error in his implementation but suspects it might not make a significant difference in terms of results in practice. Intuitively there is some sense to what Han is saying since equations (114) and (113) are both identical to the initial equation (108) when $C' = 0$. When $C' > 0$ the central directions $\alpha_{hk} \cdot \vec{\mu}_{hk}$ in $N(\Theta)$ at entry h are added to $\vec{r}_h$ which, as intended, makes it probable that convergence of $\vec{\mu}_h$ will be similar to the central direction of the most dominant vMF in $N(\Theta)$. However, the scale applied to this adjustment is different in the two cases. Even in the unlikely event that κ_h and α_h stay the same as we transcend through the mip map levels, the integer I which is the amount of normals in the coverage of a texel quadruples for each subsequent mip map level.

In section 7.3 in [HSRG07], it is said that the only thing that needs to be changed in the Spherical EM process is to use equation (114) to replace the assignment made to $\vec{\mu}_j$ in the M–step. Whether you use equation (114) or (113), this claim is wrong. The problem is that equation (103) which is used to maximize κ_h depends on $\vec{\mu}_h$ which was previously given by equation (104) in section 4.4.2. Equation (104) does not depend on κ_h and insertion into equation (103) gives a relatively nice result (105). Nevertheless, it is pointed out in [BDGS05] that κ_h cannot be isolated in this equation and it is the purpose of their paper to provide an approximation for κ_h as given by equation (109). Since, in [HSRG07], the central direction $\vec{\mu}_h$ is replaced by equation (113) this appears to invalidate use of the approximation for κ_h given in [BDGS05]. The correct thing to do would be to follow in the foot–steps of [BDGS05] and insert equation (113) into (103) and try to isolate κ_h or, as in [BDGS05], provide a new approximation. The resulting equation, however, is very complex and since it was a difficult problem to solve the first time, it would most likely be very hard given the new and more complex form. I will not attempt to solve the problem in this thesis, but it might be interesting to pursue this in a future project. I have informed Han of the problem and he has confirmed this was an oversight in his paper.

Though it is technically wrong, it could be argued that continuing to use equation (109) to evaluate κ_h can be considered an approximation. Unfortunately, equation (109) does not take $N(\Theta)$ into account, but since the E-step evaluates c_{ij} based on the chosen central directions of the M–step, κ_h is in fact influenced by $N(\Theta)$ in each subsequent iteration. This is because equation (109) is evaluated based on the coefficients c_{ij}.

4.5 Results

In addition to normal mapping with standard filtering I have implemented the method based on spherical harmonics described in section 4.1, the method based on clustering as described in sections 4.2, 4.3 and 4.4 and finally a method which will henceforth be known as *ground-truth*. This method calculates the exact

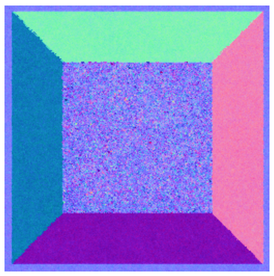
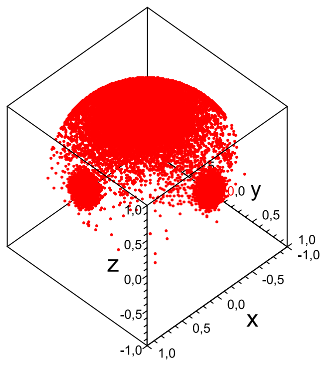


Figure 38: A normal map is shown in 38 with five surface orientations and noise.

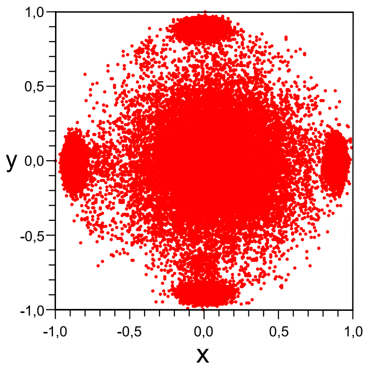
coverage, as shown in figure 34(b), and accumulates the light intensity based on the normal of every texel inside the coverage. Texels which are only partially inside are clipped against the outline of the coverage and the area of the resulting region is applied as a scale to the light intensity before accumulation. After all contributions have been accumulated, the total is divided by the area of the coverage. This process filters the light intensities as opposed to the normals. The method will, in the following, serve as a reference model of the correct result. Since it has a very long execution time, it is not considered a realistic alternative in practice.

The normal map which will be used for the following tests is shown in figure 38. It consists of five primary surface orientations with noise distributed across the map. Figure 39 shows every normal represented by a point on the unit–sphere. There appears a cluster for each of the five orientations where the thickest cluster represents the middle surface of the normal map. This surface has more noise in it which is why the cluster is fuller compared to the remaining four.

Figure 40 shows the first test which is a tea pot with the normal map tiled 60 times in the horizontal direction, 36 times in the vertical and the specular power s is set to 64. The result generated using ground–truth is shown in figure 40(b). Specular high–lights appear for each of the five surfaces. Specifically, a high–light formed as a cross appears as a result of the four sides. Standard filtering of normals was used in figure 40(a) and the result is different compared to figure 40(b). In particular, the cross shape is not as explicit as that of ground–truth. To create a large amount of over–sampling the same normal map is now tiled 480 times in the horizontal direction and 280 times in the vertical. In figure 41(c), the result obtained by ground–truth is shown and in contrast the result obtained by standard filtering of normals in figure 41(a) appears not only aliased

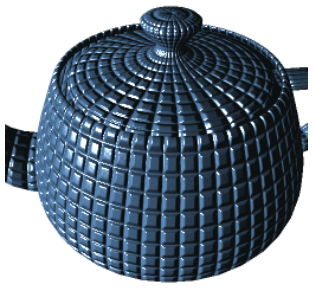


(a)

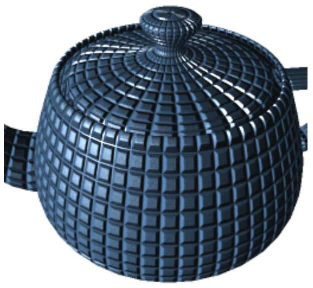


(b)

Figure 39: A normal map is shown in figure 38 with five surface orientations. The distribution of normals is shown in 39(a) and with a top–down view in 39(b). Each normal is visualized as a point on the unit–sphere.



(a)

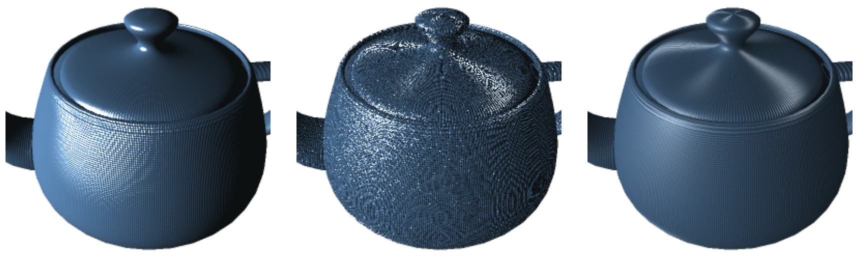


(b)

Figure 40: The result seen in 40(a) was generated using standard filtering of normals. A more accurate result is shown in 40(b) and was generated using ground–truth. The specular high–light appears as a cross on the surface due to the four sides of the normal map in figure 38.

but also completely different from the reference model. An alternative way to prevent normals across a large coverage from being averaged is to disable mip mapping. This is shown in figure 41(b) and though the specular high–lights do appear in the same locations as for the reference model, an excessive amount of noise is now visible in the result.

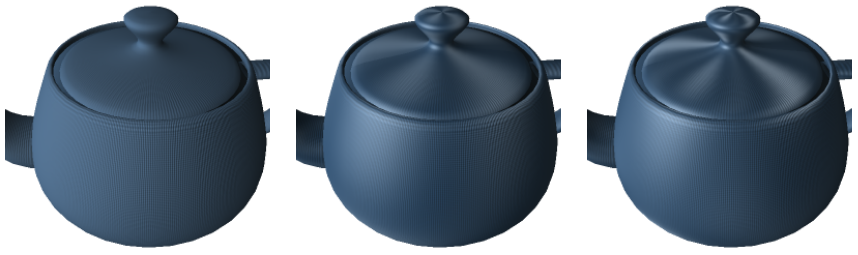
Figure 39 represents the true NDF of the last mip map level which consists



(a) (b) (c)

Figure 41: Standard filtering of normals is shown in 41(a) and appears different from ground–truth shown in 41(c). Disabling mip mapping, which is shown in 41(b), confirms the result in 41(c) but has noise.

of a single texel. The coverage of this single texel is the entire normal map. For the next test, the NDF of every texel of any mip map level is approximated using spherical harmonics as explained in section 4.1 and the light intensity is evaluated using equation (74). In figure 42(a), the result is shown using $L = 4$



(a) (b) (c)

Figure 42: The results obtained by using spherical harmonics approximation of the NDF is shown in figures 42(a)-42(c) and using 16, 64 and 144 coefficients respectively per texel.

which is a total of 16 spherical harmonic coefficients. No specular high–lights appear in the figure. Next, using $L = 8$, which is 64 coefficients, is shown in figure 42(b) and now specular high–lights appear, but they look blurred and not as bright as they need to be. Finally, in figure 42(c), we see the result using $L = 12$ which is 144 coefficients. The specular high–lights now appear much brighter and not as blurred as for $L = 8$. Nevertheless, the result does still not correspond exactly to ground–truth. The specular high–lights still appear blurred and they are also slightly dimmer than they should be. In conclusion, the spherical harmonics based method achieves better results than standard filtering of normals, but it is also confirmed that, due to the large memory foot–print, the solution is not suitable for higher values of the specular power.

In the following, testing is done using clustering and fitting of a mixture of von Mises–Fisher (vMF) distributions. Given figure 39, we know that there are five primary clusters in the distribution of normals. As previously mentioned, the last mip map level is a single texel and the corresponding coverage is the entire normal map and subsequently all five clusters need to be represented. With this in mind, it seems intuitive that a single vMF ($J = 1$) will not provide adequate results. The result $(\vec{\mu}_1, \kappa_1, \alpha_1)$ after spherical EM applied on the full coverage for $J = 1$ is given in table 1. The central direction is given in

Index j	θ	φ	κ	α
1	2.278°	334.4°	2.922	1.0

Table 1: This table shows the result of spherical EM applied to the complete distribution of normals into a single vMF. The central direction is given by (θ, φ).

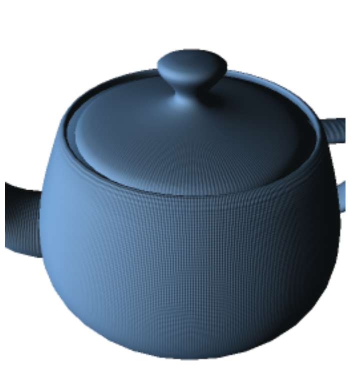
degrees as $\vec{\mu}_1 = (\theta, \varphi)$, and since θ is close to zero, which is up, the central direction approximately coincides with the Z-axis. Furthermore, since all five clusters are represented with a single vMF, this becomes a very thick vMF which corresponds to a small κ. This is confirmed by the result $\kappa_1 = 2.922$ listed in the table. Finally, $\alpha_1 = 1.0$ since there is only one vMF in the mixture. This mixture is illustrated in figure 43(b). Note that origo is at the bottom and not the center of the volume and additionally the ranges are not the same along each axis. The result obtained by using equations (87) and (88) for lighting is shown in 43(a). At the end of section 4.3 it was explained that a small κ will flatten the frequency of the specular high–light and additionally scale down the intensity of it. This is verified here.

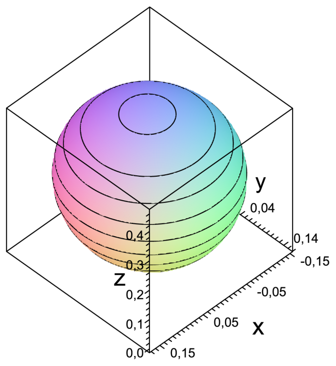
The result $(\vec{\mu}_j, \kappa_j, \alpha_j)$ after spherical EM applied on the full coverage for $J = 3$ is given in table 2. The first entry $j = 1$ has the smallest θ and thus

Index j	θ	φ	κ	α
1	20.981°	318.204°	4.869	0.717
2	64.217°	90.424°	233.027	0.140
3	62.492°	181.298°	228.74	0.143

Table 2: This table shows the result of spherical EM applied to the complete distribution of normals into three vMFs. The central direction is given by (θ, φ).

represents the vMF with the central direction closest to the Z-axis. This entry also has the smallest κ of the three and thus corresponds to the thick middle cluster of figure 39. The two last entries both have approximately $\theta = 63°$ and

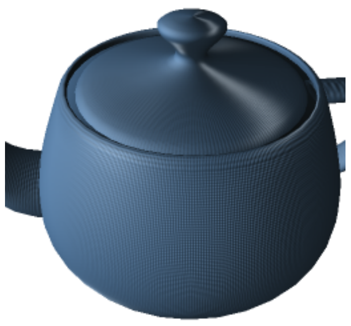

(a)

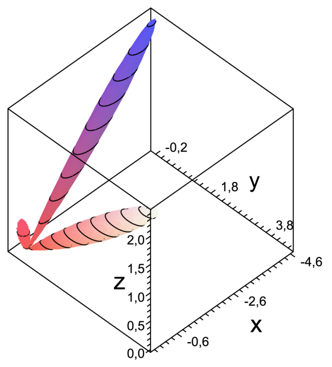


(b)

Figure 43: The result, using a single vMF in every texel, is shown in 43(a). The vMF obtained from the full coverage is shown in 43(b). Each vMF is fitted using spherical EM.

φ as 90° and 181° respectively. These correspond well to the left and upper sides of the normal map. They also have a much larger κ which agrees with the fact that the clusters of the sides are thinner. An illustration of the mixture is shown in figure 44(b). The final lit tea pot is shown in figure 44(a) and now the


(a)



(b)

Figure 44: The result, using a mixture of three vMFs in every texel, is shown in 44(a). The mixture obtained from the full coverage is shown in 44(b). Each mixture is fitted using spherical EM.

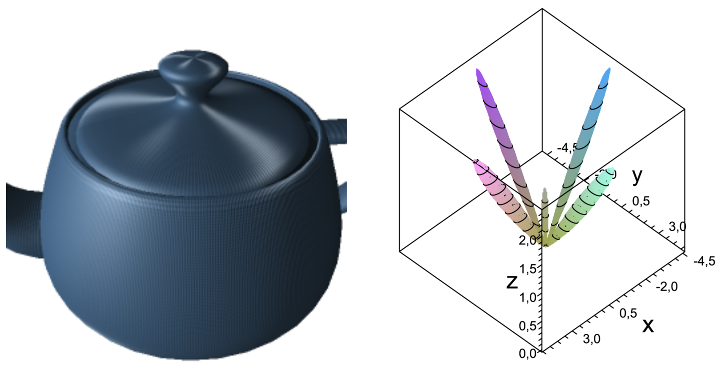
specular high–lights which, according to ground–truth, should occur from the left and upper sides of the normal map appear in the result.

The result $(\vec{\mu}_j, \kappa_j, \alpha_j)$ after spherical EM applied on the full coverage for $J = 5$ is given in table 3. Again, the first entry $j = 1$ has the smallest θ and

Index j	θ	φ	κ	α
1	2.910°	340.587°	15.529	0.426
2	62.435°	181.314°	220.148	0.144
3	63.897°	270.384°	229.626	0.142
4	64.191°	90.418°	226.349	0.141
5	63.777°	359.5°	232.69	0.147

Table 3: This table shows the result of spherical EM applied to the complete distribution of normals into five vMFs. The central direction is given by (θ, φ).

thus represents the vMF with the central direction closest to the Z-axis. This entry also has the smallest κ of the five and thus corresponds to the thick middle cluster of figure 39. The four remaining entries all have $\theta \simeq 63°$ and φ as 181°, 270°, 90° and 359° respectively. These four entries correspond to the clusters of the four sides and an illustration of the full mixture is shown in figure 45(b). The final lit tea pot is shown in figure 45(a) and now all specular high–lights



(a) (b)

Figure 45: The result, using a mixture of five vMFs in every texel, is shown in 45(a). The mixture obtained from the full coverage is shown in 45(b). Each mixture is fitted using spherical EM.

appear in the result. In fact the result is almost identical to ground–truth.

For the next test a different normal map will be used where the distribution does not gather in a small collection of clusters, see figures 46(a) and 46(b).

The result after spherical EM applied on the full coverage for $J = 5$ and $J = 16$ is shown in figures 46(c) and 46(d) respectively and they appear the same. However, this is specifically when applied to the full coverage, for texels of mip map levels with bigger dimensions than 1×1 there will be differences. To reduce

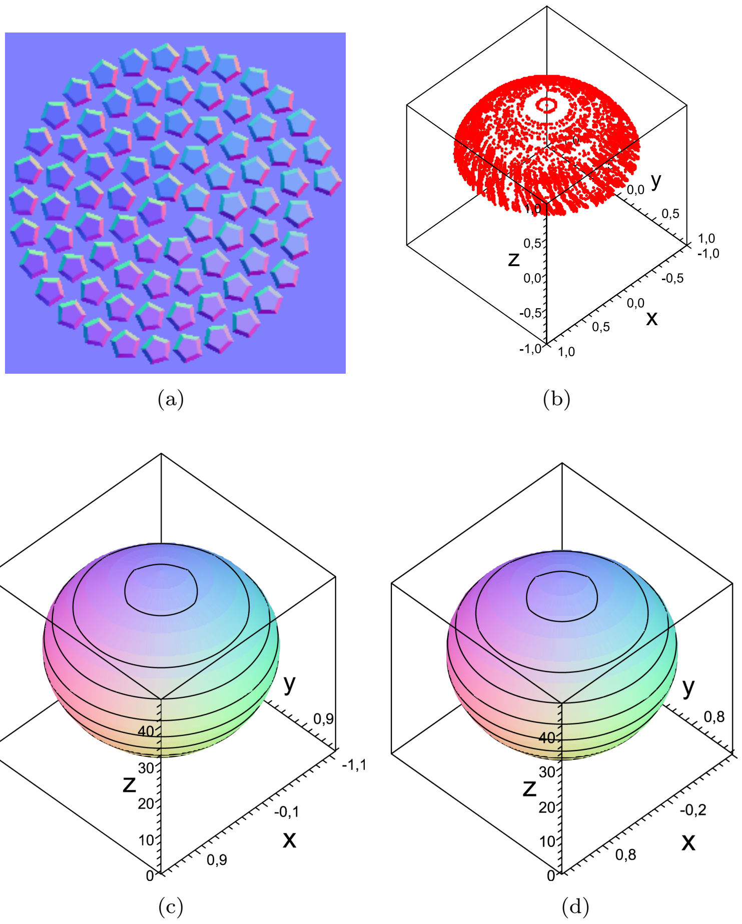


Figure 46: A normal map is shown in 46(a) with the corresponding distribution of normals shown in 46(b). Each normal is represented as a point on the unit–sphere. The mixture obtained from the full coverage is shown in 46(c). The mixture was fitted using spherical EM with five vMFs. The same principle applied using sixteen vMFs in the mixture is shown in 46(d).

the over–sampling, the tiling is changed to 120 times in the horizontal direction and 72 times in the vertical. The normal map in figure 46(a) is applied to the tea pot and a close–up using $J = 5$ is shown in figure 47(a). In contrast the result using ground–truth is shown in figure 47(c) and the specular high–lights appear a lot less blurred. The result in figure 47(b) was generated using $J = 16$ and though it appears less blurred than the result obtained using $J = 5$, the specular high–lights are not as high–frequency as that of ground–truth.

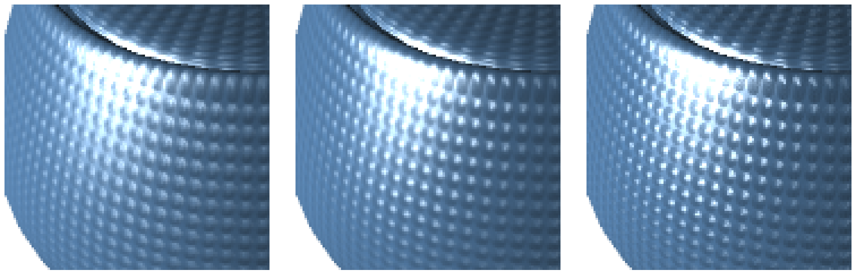


Figure 47: In 47(a) the lit tea pot, using a mixture of five vMFs per texel, is shown. Similarly in 47(b) the result is shown for sixteen vMFs per texel. Finally, in 47(c) ground–truth is shown as a reference and the specular high–lights are less blurred.

5 Conclusion

A thorough analysis of Blinn's approximation has been given, for the first time, in this thesis. Furthermore, a derivation of an equation for the exact perturbed normal has been given. According to Blinn, his approximation holds well under the assumption that the bumps are considered small, relative to the extent of the surface. The exact equation presented here reveals that a more correct interpretation is that the approximation holds well when the product between the bump value and each principal curvature is close to zero. As initially suspected, the visual results showed no significant difference in terms of lighting. This indicates that the approximation is sensible, and at least for rendering, we can continue to use it. Nevertheless, the presented analysis on the subject is significant because it provides a fundamental understanding of the approximation made and confirms its application. Furthermore, examples have been given here showing a very detectable difference between the exact and the approximate perturbed normal as the principal curvatures and the bump value increase which also confirms the analysis. It is likely that the exact equation will prove useful in other contexts of 3D graphics, such as physics, or possibly even within other areas of computer science.

Furthermore, we have managed to extend Blinn's work to function under reparametrizations of the surface which preserve the orientation as well as those which do not. This includes a valid expression for the exact normal as well as the approximate one. The use of reparametrizations of the surface enables us to perform manipulations of the texture coordinate prior to texture sampling. Such manipulations were not covered by Blinn, but in fact any diffeomorphism on the input coordinate to the surface is a valid manipulation. A special case of such a diffeomorphism is an affine map with an inverse. This means that graphic artists can, for example, rotate, scale, shear and translate texture sampling coordinates. Such an operation simply reorients the bump map on the surface. Bump mapping under reparametrizations is a common practice but a complete proof and derivation of the perturbation equation has, to the author's knowledge, not been given before this thesis. Additionally, the analysis on the subject provided here has proven useful in terms of studying and perfecting bump mapping applied to triangular meshes.

Bump mapping as defined by Jim Blinn requires the presence of a known surface parametrization to evaluate the first–order derivatives of the surface. The theory for extending bump mapping to work for triangular meshes has, until now, been poorly documented. It was briefly suggested in a paper by Nelson L. Max that one could approximate the derivatives at vertex level by averaging the first–order derivatives of the surrounding triangles. The accumulated derivatives are subsequently interpolated across each triangle as rasterization is performed. Today, this strategy has been adopted by most, if not all, commercial products, because it is very hardware efficient. Nevertheless, it has been shown in this thesis that it is a nontrivial solution which, if not carefully planned, easily leads to visual errors. In particular, several examples have been given showing leading commercial graphics tools exhibiting such errors in their results.

Based on a thorough analysis and by observing flaws in methods used by predecessors and current commercial products, a new algorithm for vertex level tangent space evaluation is presented in this thesis. The technique is based on a set of coherency rules, defined in this thesis, which are used to determine which triangles surrounding a vertex should share tangent space. These rules are to a large extent the result of studying reparametrized surfaces in general and the proof made in this thesis that texture coordinates assigned at the vertices of a triangle is in fact an instance of reparametrization using an affine map. Another important aspect of the algorithm is that it has been defined to provide results independently of the order in which triangles of the mesh are given. This is important to provide consistency but as shown in this thesis also important to preserve mesh symmetries. Failing to do so can lead to discontinuities in the lighting between adjacent triangles.

As shown in this thesis, bump maps can be converted into tangent space normal maps, and similarly normals sampled from a 3D surface/mesh can be transformed into tangent space and are then also stored as a tangent space normal map. This enables a uniform processing of the two and today the concept is an industry standard. An important observation made in this thesis is that, although converted bump maps can be applied to any surface, the same does not fully apply to sampled normal maps. Since each sampled normal is transformed by a matrix into tangent space, it is important to use the exact inverse during rasterization to obtain the original sampled normal to be used in the lighting calculation. Not doing so will cause a deviation between the sampled normal and the final normal passed to the lighting model. Such a deviation alone is not a problem. However, for adjacent triangles which share vertex normals but not averaged vertex level derivatives, and subsequently do not share tangent space, it is a problem. This is because the discontinuity in tangent space, at the edge between the two triangles, will lead to a discontinuity in the deviation which leads to a discontinuity in the final normal passed to the lighting model. Finally, this creates what is known as a shading seam, i.e., an unwanted discontinuity in the lighting at an edge between two triangles. If, on the other hand, tangent spaces are shared at the edge, then the deviation (and the final normal) will transition continuously across the edge and thus preserve continuity of the lighting calculations. A problematic, yet typical, test–case provided by IO-Interactive tested with several leading graphics tools resulted in such shading seams. Since in each case the sampler and the viewer was provided by the same middle–ware company, this is a clear indication that their developers failed to acknowledge the dependency in terms of chosen transformation between the sampler and the viewer. In contrast, respecting the dependency and using the technique presented in this thesis to assign vertex level tangent spaces provided very nice and completely artifact–free results.

It has been pointed out in this thesis that there exists no single right method to calculate tangent space at a point within a triangle based on the barycentric weights and the three tangent spaces assigned at vertex level. Additionally, there exists no single right way to determine the tangent spaces assigned at vertex level. There does not even exist a generally accepted standard for an approximation. This, combined with the aforementioned dependency between the normal map sampler and the application using the normal map for rendering, means there are compatibility problems between different tools and renderers. Though such a problem might be solved by having a final standard introduced, it seems unlikely that one single standard might be generally accepted. In this thesis alone, four different strategies for interpolated tangent space were suggested with different strengths and weaknesses. To make matters worse, existing middleware tools for sampling normal maps generally do not reveal their strategies for interpolated tangent space nor those assigned at vertex level. This means that even if a developer is willing to adopt the concept of tangent space used by the sampling tool in use, he/she does not have access to the information needed and thus cannot reconstruct the accurate inverse in the shader as required.

In conclusion, there are two sensible options for developers who use sampled normal maps. One is to have every such developer write his/her own tool for sampling which will allow him/her to control the notion of tangent space used in their application. The second option is to develop a customizable sampling tool which will allow developers to overwrite tangent space if they are not satisfied with the default implementation. Furthermore, full documentation and source–code must be provided for the default implementation of every tool. In terms of future work, it would be interesting to pursue this idea and more specifically analyze how such an interface to the sampling tool should be provided.

In this thesis, the work on filtering normal maps by Charles Han et al. has been thoroughly described, closely studied and additionally implemented and tested. A large portion of the work involved in this thesis has been spent on interpreting the comprehensive math used by Han and also on finding derivations to equations given, but not shown, in his paper. Additionally, getting well acquainted with the underlying theoretical principles of spherical EM has also been very time–consuming.

The work describes two primary techniques, the first uses spherical harmonics to approximate the distribution of normals and also to approximate the material reflectance. The results were compared to an accurate brute–force implementation, referred to as ground–truth. This comparison showed that using spherical harmonics does approach the results achieved by ground–truth as the amount of coefficients used for the approximation is increased. However, the test also confirmed the comment by Han that a very large amount of coefficients is required for high–order specularity.

The second technique described by Han approximates the distribution of normals using a mixture of a small user–defined amount of Gaussian–like distributions on the spherical domain known as von Mises–Fisher. This mixture is fitted to the distribution of normals using a recent technique known as spherical EM. The analysis and tests performed in this thesis confirmed that the method is remarkably successful when the distribution of normals gather in a small amount of thin clusters. Under different circumstances, the results were visually appealing but appeared as if the specular high–lights had been heavily blurred compared to ground–truth. An additional test showed that this blurring is reduced as the amount of von Mises–Fisher distributions used in the mixture is increased. However, doing so also increases (linearly) the foot–print and the execution time.

Finally, three errors in Han's paper were discovered during the development of this thesis. The first error is in the E–step (see section 4.4.4 for details) and the second is equation (23), in his paper, which has an error due to an incorrect substitution (see section 4.4.5). Fortunately, both errors are easy to solve. The third error, on the other hand, is difficult to solve. By changing the distribution in use, Han is no longer compliant with the form used in spherical EM. Han does reevaluate the equation in the M–step for the central direction, but overlooks that in spherical EM the maximized thickness κ depends on the central direction and subsequently it is, at least from a technical standpoint, no longer correct to evaluate κ the same way. However, as pointed out in section 4.4.5, since EM is about iterative refinement, it is possible that it is a usable approximation. For future work, it would be interesting to analyze this specific issue in greater detail and possibly try to determine a new and more accurate approximation for κ in the M–step given the modified distribution. Another possible course of future work might be to try to extend the method such that the amount of von Mises–Fisher distributions used in the mixture is automatically chosen by the algorithm as opposed to being user–defined. Such features are already available in existing standard EM algorithms.

References

[Ada94] Robert A. Adams. *Calculus: A Complete Course.* Addison–Wesley Pub., third edition, 1994.

[BDGS05] Arindam Banerjee, Inderjit S. Dhillon, Joydeep Ghosh, and Suvrit Sra. Clustering on the unit hypersphere using von mises-fisher distributions. *J. Mach. Learn. Res.*, 6:1345–1382, 2005.

[Ben06] Carsten Benthin. *Realtime Ray Tracing on current CPU Architectures.* PhD thesis, Saarland University, Saarbrucken, Germany, January 2006.

[Bis07] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer, August 2007.

[Bli78] J.F. Blinn. Simulation of wrinkled surfaces. In *Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, pages 286–292, 1978.

[Coh98] Jonathan David Cohen. *Appearance-preserving simplification of polygonal models.* PhD thesis, The University of North Carolina at Chapel Hill, 1998. Adviser-Dinesh Manocha.

[Coo84] Robert L. Cook. Shade trees. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 223–231, 1984.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[EWWL98] Jon P. Ewins, Marcus D. Waller, Martin White, and Paul F. Lister. Mip-map level selection for texture mapping. *IEEE Transactions on Visualization and Computer Graphics*, 4(4):317–329, 1998.

[FvDFH95] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer graphics: principles and practice.* Addison-Wesley Professional, Boston, MA, USA, second edition, 1995.

[Gla94] Andrew S. Glassner. *Principles of Digital Image Synthesis.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994.

[HS85] Dorit S. Hochbaum and David B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, May 1985.

[HSRG07] Charles Han, Bo Sun, Ravi Ramamoorthi, and Eitan Grinspun. Frequency domain normal map filtering. *ACM Trans. Graph.*, 26(3):28, 2007.

[Kil00] Mark J. Kilgard. A practical and robust bump-mapping technique for today's gpu's. Technical report, NVIDIA Corporation, February 2000. Available at http://www.nvidia.com/.

[Mac48] T.M. MacRobert. *Spherical Harmonics; an elementary treatise on harmonic functions, with applications.* Dover Publications., 1948.

[Max88] Nelson L. Max. Horizon mapping: shadows for bump-mapped surfaces. *The Visual Computer*, 4(2):109–117, 1988.

[Mes97] Robert Messer. *Linear Algebra; Gateway to Mathematics.* Addison–Wesley, 1997.

[MPFJ99] Joel McCormack, Ronald Perry, Keith I. Farkas, and Norman P. Jouppi. Feline: fast elliptical lines for anisotropic texture mapping. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 243–250, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[ON97] M. Olano and M. North. Normal distribution mapping. Technical Report 97-041, The University of North Carolina at Chapel Hill, 1997. url: http://www.cs.unc.edu/ olano/papers/ndm/.

[Pre01] Andrew Pressley. *Elementary Differential Geometry.* Springer, 2001.

[RH01] Ravi Ramamoorthi and Pat Hanrahan. A signal-processing framework for inverse rendering. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 117–128, New York, NY, USA, 2001. ACM.

[Sch97] Andreas Schilling. Towards real-time photorealistic rendering: challenges and solutions. In *HWWS '97: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 7–15, New York, NY, USA, 1997. ACM.

[Sch06] Christian Schuler. Normal mapping without precomputed tangents. In Wolfgang Engel, editor, *ShaderX5: Advanced Rendering Techniques*, pages 131–140. Charles River Media, Cambridge, MA, 2006.

[TLQ$^+$05] Ping Tan, Stephen Lin, Long Quan, Baining Guo, and Heung-Yeung Shum. Multiresolution reflectance filtering. In *Proceedings of the Eurographics Symposium on Rendering Techniques*, pages 111–116. Eurographics Association, 2005.