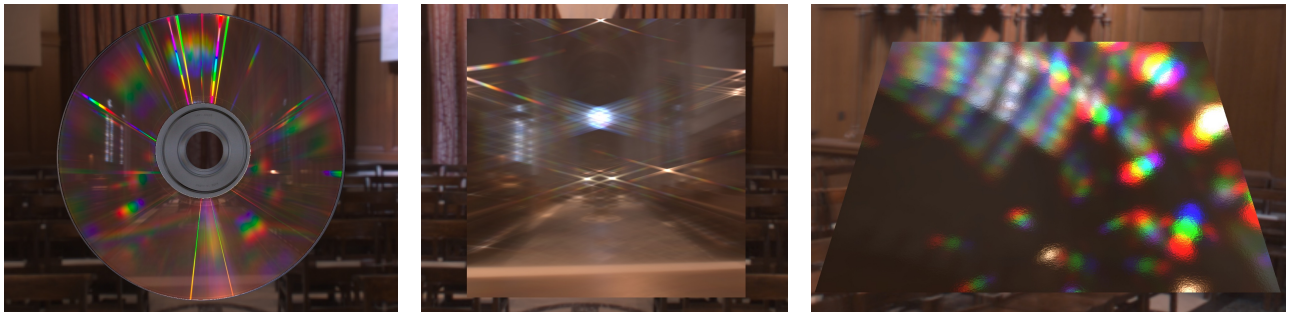
Real-time rendering of realistic surface diffraction with low rank factorisation

Antoine Toisoul
Imperial College London
Department Of Computing
ast13@imperial.ac.uk

Abhijeet Ghosh
Imperial College London
Department Of Computing
ghosh@imperial.ac.uk



(a) (b) (c)

Figure 1: Real-time renderings of diffraction effects in the Grace Cathedral. (a) Compact Disc. (b) Diffraction pattern of a LG 42" Smart TV. (c) Wide diffraction lobes on a rough specular holographic paper.

ABSTRACT

We propose a novel approach for real-time rendering of diffraction effects in surface reflectance in arbitrary environments. Such renderings are usually extremely expensive as they require the computation of a convolution at real-time framerates. In the case of diffraction, the diffraction lobes usually have high frequency details that can only be captured with high resolution convolution kernels which make calculations even more expensive. Our method uses a low rank factorisation of the diffraction lookup table to approximate a 2D convolution kernel by two simpler low rank kernels which allow the computation of the convolution at real-time framerates using two rendering passes. We show realistic renderings in arbitrary environments and achieve a performance from 50 to 100 FPS making possible to use such a technique in real-time applications such as video games and VR.

CCS CONCEPTS

• **Computing methodologies** → **Rendering**;

KEYWORDS

Real-time rendering, diffraction, rank factorisation

ACM Reference format:
Antoine Toisoul and Abhijeet Ghosh. 2017. Real-time rendering of realistic surface diffraction with low rank factorisation. In *Proceedings of 14th European Conference on Visual Media Production (CVMP 2017), London, United Kingdom, December 11–13, 2017 (CVMP 2017),* 7 pages.
https://doi.org/10.1145/3150165.3150167

CVMP 2017, December 11–13, 2017, London, United Kingdom
2017. ACM ISBN 978-1-4503-5329-8/17/12…$15.00
https://doi.org/10.1145/3150165.3150167

1 INTRODUCTION

Research on appearance modelling and rendering is a very active area in computer graphics. Until now most of the effort has focused on rendering effects based on geometric optics. However a few impressive effects such as iridescence cannot be rendered with geometric optics and require wave optics to be understood. Iridescence is a common and appealing effect seen in real life. It causes the rainbow colours seen on surfaces such as compact discs and soap bubbles. In this work we are interested in a particular type of iridescence caused by diffraction of light. When the microgeometry of a surface reaches a size close to the wavelength of light (around the micrometer), light is diffracted by the surface. As a result light waves interfere and produce constructive and destructive interferences depending on the waves being in phase or out of phase [3]. These interferences happen at different angles depending on the wavelength of light, hence creating a separation of colours under white light. These colours always appear from purple to red as smaller wavelengths are reflected at a smaller angle than longer wavelengths. They also repeat angularly in orders of diffraction (see figure 2) which creates a diffraction pattern.

Photorealistic renderings of such effects at real-time framerates have always been challenging even under a point light source. In 1999, Stam [17] proposed the first Bidirectional Reflectance Distribution Function (BRDF) for rendering diffraction effects in surface reflectance. His BRDF cannot be computed in real-time as it requires the computation of a Fourier transform each time the viewing direction or incoming light direction is changed.

Recently, Toisoul & Ghosh [19] have proposed a measurement method to render diffraction effects in surface reflectance based a first order approximation of Stam's BRDF. They measure the

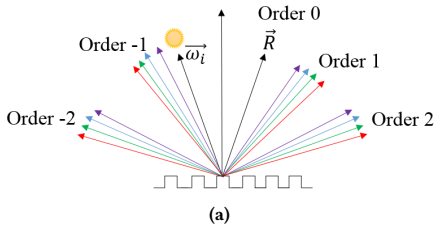


Figure 2: (a) Diffraction of light on a periodic microgeometry. The reflection is wavelength-dependent with the purple colour being reflected at a smaller angle than the red colour [19].

diffraction pattern at a single wavelength with a spectral filter and then compute the diffraction pattern for any light spectrum. The result is stored in a single lookup table, called the diffraction lookup table, that can be used in a fragment shader for real-time rendering. They also present renderings in arbitrary environments by prefiltering (convolving) the environment map (EM) with the diffraction lookup table.

The idea of prefiltering environment maps with specular lobes for real-time rendering of realistic reflections goes back to Kautz et al. [12]. However in the case of diffraction, the reflection of light produces not only a specular lobe but also anisotropic diffraction lobes with high frequency features. As a result prefiltering the environment map by the diffraction lobes requires a bigger convolution kernel which makes computations more expensive. Besides, due to the anisotropy the prefiltering step only works for a single orientation of the object [19]. In this work we present a factorisation approach to approximate a 2D diffraction lookup table with an outter product of two low rank matrices. In particular, we show that the rank 1 factorisation approximates a 2D convolution with two one dimensional convolutions making real time renderings under any rotation possible on current hardware. The paper is organized as follows. Section 3 presents a theoretical formulation of the environment mapping, section 4 deals with the rank factorisation and section 5 talks about the implementation and presents realistic results for several manufactured materials that exhibit diffraction in surface reflectance.

2 RELATED WORK

Diffraction effects and iridescence : In 1999, Stam [17] presented the first diffraction BRDF derived from Kirchoff theory of diffraction. In his work, he describes the microgeometry of the surface that produces diffraction as a height field. The height field causes a change in the phase of the reflected waves. Such a change in phase is described with a phaser function p that links the height field and the viewing and incoming light directions. The final rendering requires the computation of the Fourier transform of the correlation of p at each frame of an animation, making real-time renderings difficult to achieve. This is why, he also derives a few analytic solutions for simple diffraction gratings (e.g. periodic grooves) that work in real-time. However an analytic solution of the BRDF for any height field does not always exist.

In 2014, Dhillon et al. [5] reformulated Stam's BDRF to take a height field measured with an atomic force microscope as an input. Besides they use a Taylor expansion that break the dependency between the height field and the viewing direction and incoming light direction in the BRDF in order to allow real-time renderings under point light illumintation. Indeed, they precompute the Taylor terms in lookup tables that can then used to compute the Taylor approximation in a fragment shader. Although their formulation can be computed in real-time for a single light source they do not provide a way to compute the BRDF under arbitrary illumination at real-time framerates.

Sun et al. [18] rendered iridescence on optical discs by modelling the grooves on a CD by consecutive spheres and by calculating the reflected electric field. Although realistic their model only works for CDs. Besides they do not present any rendering in arbitrary environments. In this work the CD is rendered using the lookup table formulation of Toisoul & Ghosh [19] and not Sun et al. [18] model. In 2002, Agu [1] proposed an approach to render diffraction produced by a multislit diffraction grating (grating made of periodic rectangles). His analytic solution is physically based and works in real-time but cannot be used to render arbitrary diffraction patterns.

In 2006, Lindsay & Agu [13] showed that rendering of diffraction under any object rotation, in arbitrary environment at real-time framerates is possible using spherical harmonics. They precompute spherical harmonics coefficients for the environment map and the BRDF and compute the rendering using the spherical harmonics representation. However their renderings is limited to the diffraction produced by grooves such as Compact Discs but does not work for arbitrary patterns. Besides the spherical harmonics representation only captures the low frequency lighting of an environment and diffraction of light is usually more visible in high frequency lighting environments. Our method works in every environment.

Kang et al. [9] proposed a simple way of rendering diffraction in real-time using an anisotropic microfacet reflection model and without spectral computations. They use different rotation of the half-vector for the three color channels RGB which produces iridescence effects. However their approach is not physically-based and can produce wrong variations of colors in the diffraction orders if the rotation is not set correctly.

Cuypers et al. [4] proposed a diffraction model based on the computation of a wave BSDF and Wigner distributions. Their model can be implemented in a ray-tracer and produce impressive renderings for direct and indirect bounces of light. However their approach is not made for real-time rendering.

Recently, Holzschuch & Pacanowski [7] presented a two-scale BRDF model to model real world surfaces that have both macro and micro scale geometry variations. The model combines a diffraction term to represent the reflection produced the microscale geometry and a specular term (Cook Torrance) to model the reflection produce by the macroscale geometry. They show that their BRDF model provides better fit to measured BRDF data such as the MERL database [15]. However they do not present real-time renderings using their BRDF model.

Also related is the work of Berlcour & Barla [2] that deals with rendering of iridescence caused by thin film of varying thickness (e.g. soap bubbles). Similarly to diffraction, thin film interference is

also a microscale effect that produces dispersion of light. However thin film interference is not caused by microgeometry variations but by optical path differences through materials with thin layers. Their model adapts regular microfacet BRDFs by replacing the common Fresnel term by an Airy reflectance term that accounts for the phase shifts between waves. They render iridescence in reflection and transmission for dielectrics and reflection for conductors. For perfectly smooth surfaces their model work in real-time under environmental illumination using prefiltered environment maps.

Prefiltering of environment maps : Kautz et al. [12] discuss prefiltering of environment maps (EM) with several methods to render specular reflections in arbitrary environments at real-time framerates. In particular they show that a prefiltered environment map can be indexed by the reflection vector in spherical coordinates instead of the normal and the light direction. This allows a 2D parametrisation instead of 4D. They also propose a hierarchical filtering method to accelerate the filtering process and a method to filter EM filter with anisotropic kernels.

Karis et al. [10] explain how prefiltering of environment maps is computed in the Unreal Engine 4 for rough specular materials. They use an isotropic approximation to precompute a convolution between the specular lobe of the BRDF and the environment map. The result is indexed by the reflection vector and can be used for real-time rendering in an arbitrary environment. The isotropic assumption causes inaccurate reflections at grazing angles but they demonstrate that these are not very noticeable in practice. We use their formulation for the filtering of the environment map with the diffraction kernel.

Kautz & McCool [11] use a separable approximation to render arbitrary BRDFs interactively based on a singular value decomposition or a normalized decomposition. Our paper uses a similar technique but applied to a diffractive BRDF.

3 THEORETICAL FORMULATION

3.1 Data-driven reflectance model

In this work the rendering of diffraction is calculated using the lookup table formulation proposed by Toisoul & Ghosh [19] assuming a first order approximation of Stam's BRDF. The diffractive component of the BRDF in their model is given by equation 1 where F is the Fresnel term and G is Stam's geometric term [17]. $\vec{\omega}_i$ and $\vec{\omega}_o$ are the incident light direction and viewing direction respectively. The vector $\vec{h} = \omega_i + \omega_o$ is the non normalized half-vector.

$$f_{\text{r,diffraction}}(\vec{\omega}_i, \vec{\omega}_o) = 4\pi^2 F^2(\vec{\omega}_i, \vec{\omega}_o) G(\vec{\omega}_i, \vec{\omega}_o) S_d(\vec{h}) \quad (1)$$

The term S_d is the diffraction lookup table. It corresponds to a diffraction pattern for a given light spectrum with a spectral power distibution P. S_d is computed by calculating the integral shown in equation 2 and using a measurement of a diffraction pattern at a single wavelength, denoted I. Besides the look up table is parametrised by the projection of the non normalized half vector $\vec{h}$ onto the tangential coordinate frame $(\vec{t}, \vec{b}, \vec{n})$. λ denotes the wavelength of the light.

$$S_d(\vec{h}) = \int_{\lambda_{min}}^{\lambda_{max}} \frac{1}{\lambda^2} I(\frac{2\pi}{\lambda}\vec{h}.\vec{t}, \frac{2\pi}{\lambda}\vec{h}.\vec{b}) P(\lambda) d\lambda \quad (2)$$

3.2 Environment map rendering

We derive the formulation of the environment map rendering for an anisotropic diffraction pattern as a convolution by adapting the work of Kautz et al. [12] and Karis et al. [10]. We start from the rendering equation [8] :

$$L_i(\vec{\omega}_o, \vec{n}) = \int_\Omega f_r(\vec{\omega}_i, \vec{\omega}_o) L_i(\vec{\omega}_i)(\vec{n}.\vec{\omega}_i) d\omega_i \quad (3)$$

We assume the viewing vector to be constant to $\vec{\omega}_o = (0, 0, 1)$ and uses the same assumption as [10] that the viewing vector is equal to the normal and therefore to its reflection vector : $\vec{n} = \vec{\omega_o} = \vec{r_v}$. As a result, under Schick approximation [16], the Fresnel factor becomes a constant and equal to its value at normal incidence F_0. Hence equation 3 becomes :

$$L_i(\vec{\omega}_o, \vec{n}) = 4\pi^2 F_0^2 \int_\Omega G(\vec{\omega}_i, \vec{\omega}_o) S_d(\vec{h}) L_i(\vec{\omega}_i)(\vec{n}.\vec{\omega}_i) d\omega_i \quad (4)$$

Under the assumption $\vec{n} = \vec{\omega_o} = \vec{r_v}$, Stam's geometric factor becomes :

$$G(\vec{\omega}_i, \vec{\omega}_o) = \frac{(1 + \vec{\omega}_i.\vec{\omega}_o)^2}{(\vec{n}.\vec{\omega}_o)(\vec{n}.\vec{\omega}_i)} = \frac{(1 + \vec{\omega}_i.\vec{\omega}_o)^2}{\vec{n}.\vec{\omega}_i} \quad (5)$$

Hence under environmental illumination represented as a finite number of light sources N, we have :

$$L_i(\vec{\omega}_o, \vec{n}) = 4\pi^2 F_0^2 \sum_{k=1}^{N} (1 + \vec{\omega}_{i,k}.\vec{\omega}_o)^2 S_d(\vec{h}) L_i(\vec{\omega}_{i,k}) \quad (6)$$

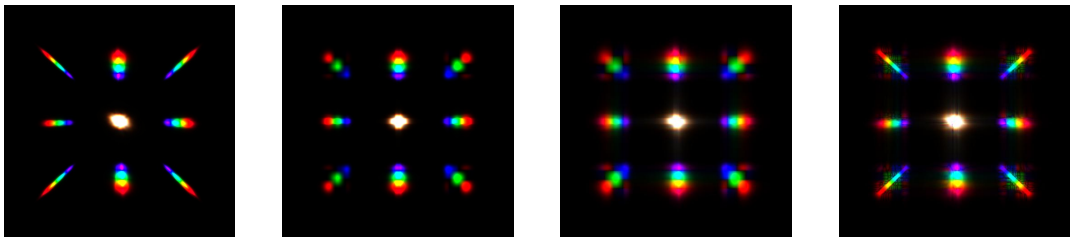
The diffraction pattern only covers a finite set of non normalized half vectors $\vec{h}$. Hence the sum becomes a sum on the values of $\vec{h}$ covered by the diffraction lookup table :

$$L_i(\vec{r}_v) = 4\pi^2 F_0^2 \sum_{k=1}^{height} \sum_{l=1}^{width} (1 + \vec{\omega}_i.\vec{\omega}_o)^2 S_d(h_k, h_l) L_i(\vec{\omega}_i) \quad (7)$$

where $width$ and $height$ are the width and height of the diffraction lookup table. The values h_l and h_k are respectively the x and y component of the non normalized half vector in the tangential coordinate frame $(\vec{t}, \vec{b}, \vec{n})$ as the diffraction lookup table is parametrised by the projection of $\vec{h}$ onto $\vec{t}$ and $\vec{b}$. Note that for every half-vector $\vec{h}$ corresponds a unique light direction $\vec{\omega}_i$ as the viewing direction is supposed fixed to $(0, 0, 1)$. Besides the environment map is parametrised and indexed with the reflection vector of the viewing direction $\vec{r}_v$ to be stored as a two dimensional texture [12]. As shown by equation 7, the environment map rendering is a convolution between the diffraction lookup table S_d and the environment map L_i. Such a convolution cannot be computed in real-time, even for small diffraction tables. Hence the next section discusses how it can be factorised into a outter product of two low rank matrices to allow a computation at real-time framerates.

4 RANK FACTORISATION

This section explains how the 2D diffraction lookup table of several manufactured materials that exhibit diffraction effects can be factorised into an outter product of two low rank matrices (see figure 3). We first present the factorisation step for axis aligned diffraction patterns using a Singular Value Decomposition. We next show how



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

Figure 3: Diffraction table of a holographic paper. (a) Original table. (b-d) Reconstruction using SVD. (b) Rank 1. (c) Rank 2. (d) Rank 10.

non axis aligned diffraction patterns can also be factorised with SVD by applying a preprocessing step.

4.1 Axis aligned diffraction patterns

Given a RGB diffraction lookup table S_d, we want to find three low rank matrices (one for each color channel) that best approximate S_d i.e solve the optimisation problem shown in equation 8 where $\|.\|_F$ is the Frobenius norm on matrices. The problem is only presented for the red colour channel of the lookup table but is applied to each channel separately.

$$\begin{aligned} \arg \min_{\tilde{S}_{d,R}} \quad & \|S_{d,R} - \tilde{S}_{d,R}\|_F \\ \text{subject to} \quad & rank(\tilde{S}_{d,R}) = r \end{aligned} \tag{8}$$

The solution to this problem can be found using a singular value decomposition of the matrix $S_{d,R} = U_R \Sigma_R V_R^T$ where U_R and V_R are orthogonal matrices and Σ_R is a diagonal matrix containing the singular values of $S_{d,R}$. The final rank r matrix that best approximate $S_{d,R}$ is given by equation 4.1 where $U_{R,r}$ and $V_{R,r}$ are the r first column of U_R and V_R and $\Sigma_{R,r}$ is the diagonal matrix containing the r largest singular values of $S_{d,R}$ [14].

$$\tilde{S}_{d,R} = U_{R,r} \Sigma_{R,r} V_{R,r}^T$$

Instead of computing a convolution between the environment map and the lookup table S_d, this factorisation allows us to store two low dimensional convolution filters, one vertical and one horizontal, and compute a separable convolution. The red channel of the vertical filter is given by $f_{v,R} = \Sigma_{R,r}{}^{\frac{1}{2}} U_{R,r}$ and the red channel of the horizontal filter is $f_{h,R} = \Sigma_{R,r}{}^{\frac{1}{2}} V_{R,r}^T$. At the end of the process we obtain two RGB filters and their product gives the matrix of rank r that best approximate of the diffraction lookup table. An example of a diffraction lookup table and its factorisation into matrices of rank 1, 2 and 10 is presented in figure 3. In particular, we use the rank 1 factorisation of S_d to compute the convolution in real-time as two one dimensional convolutions.

4.2 Non axis aligned diffraction patterns

The previous method gives a good approximation of axis aligned diffraction patterns. However the reconstruction of non axis aligned diffraction patterns is not optimal as higher order lobes of diffraction disappear (see figure 4). As a result we apply a preprocessing step to automatically straighten a diffraction pattern and make it axis aligned before computing the SVD decomposition.

Aligning the diffraction pattern with the X and Y axis requires an affine transformation. The affine transformation is entirely defined by the transformation of 3 points on the pattern. Therefore we first find the main directions of the pattern using an exhaustive search to find the two rays (starting from the specular lobe and parametrised with an angle) with maximum intensity in the pattern. For the intensity along the ray, we found that summing the green channel gives the best result for the main directions as the green colour always appear in the middle of the diffractions orders. Note that we further impose a constraint as we are looking for two rays with angle greater than a given value (e.g. 60 degrees in the case of the pattern in Fig. 4). This avoids finding two collinear rays. Once we have these directions we choose the center of the specular lobe and two points on the two main directions of the pattern to be transformed to points aligned with the vectors $(1, 0)$ and $(0, 1)$. We then apply the transformation matrix using OpenCV *getAffineTransform* and *warpAffine* functions. Note that we also save the original main directions of the diffraction pattern to rotate the convolution kernel in the rendering process, i.e to apply the inverse of the affine transformation for realistic renderings (see section 5). We then apply the SVD explained in section 4.1 to get our convolution kernels. Figure 4 shows a comparison of the reconstruction with and without an affine transformation.

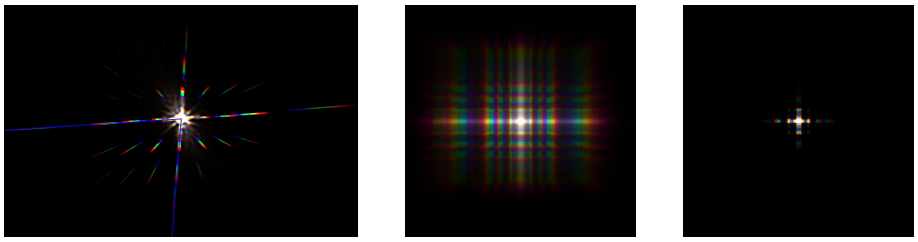
4.3 Rendering formulation

Equation 7 showed that the rendering is a convolution between the environment map and the diffraction table. In the case where the table is factorised into a vertical and a horizontal filter we have $S_d(\vec{h}) = f_v(\vec{h}) f_h(\vec{h})$ and the rendering equation becomes :

$$L_i(\vec{r}_v) = 4\pi^2 F_0^2 \sum_{\vec{h}} (1 + \vec{\omega}_i.\vec{\omega}_o)^2 f_v(\vec{h}) f_h(\vec{h}) L_i(\vec{\omega}_i) \tag{9}$$

where the sum spans all possible non normalized half-vectors covered by the filters f_v and f_h. As the convolution kernel is now separable, this can be reformulated using the rank r of the matrix :

$$L_i(\vec{r}_v) = 4\pi^2 F_0^2 \sum_{k=1}^{height} \sum_{m=1}^{r} f_v(h_k, h_m) L_{int}(\vec{\omega}_i) \tag{10}$$



(a) (b) (c)

Figure 4: Diffraction pattern of the HTC 8X smartphone. (a) Diffraction table with identified main direction. (b) Rank 1 reconstruction using an affine transformation and a SVD. (c) Direct reconstruction with SVD only.

with L_{int} being the intermediate calculation :

$$L_{int}(\vec{r}_v) = \sum_{l=1}^{width} \sum_{n=1}^{r} (1 + \vec{\omega}_i.\vec{\omega}_o)^2 f_h(h_n, h_l) L_i(\vec{\omega}_i) \qquad (11)$$

Usually $r << width$ and $r << height$ which makes the computation of two low rank convolutions faster than a 2D convolution.

5 RENDERING

5.1 Implementation

The diffraction lookup tables considered in this work are the one given by Toisoul & Ghosh [19]. The lookup table is parametrised by the vector $\vec{h} = \omega_i + \omega_o$ (non normalized half vector). We compute the convolution between the EM and the diffraction look up table using equation 11. The convolution assumes $\vec{\omega}_o = \vec{n} = \vec{r}_v$ and is therefore an approximation that is not valid at grazing angles. In practice the approximation also leads to curved diffraction pattern around the poles of the environment map which is corrected using a rendering in four passes (see section 5.2).

In our case, the EM filtering is computed in real-time in two passes : a vertical pass and a horizontal pass corresponding to the factorisation presented in section 4. In each pass we first calculate the vector $\vec{h}$ in the (x,y) plane and rotate $\vec{h}$ depending on the tangential coordinate frame of the object $(\vec{t}, n \vec{\times} t, \vec{n})$ to take into account the rotation of the object. We also apply the inverse of the 2D affine transformation (obtained in section 4.2) if the pattern is not axis aligned. Finally the z component of $\vec{h}$ is then given by : $1.0 + \sqrt{1.0 - h_x^2 - h_y^2}$ as the vector is not normalized and the normal $\vec{n}$ is assumed to be equal to $\vec{\omega}_o$. The full algorithm that is applied in the fragment shader is described in algorithms 1 and 2. *EM* denotes the environment map in latitude longitude format. *HFilter* denotes the horizontal filter and *currentPixel* denotes the current fragment that the fragment shader is computing. Note that the pseudo code only covers the horizontal pass. The vertical pass is identical except that it is not required to compute and apply the geometric factor G and the indices of the filter have to be switched.

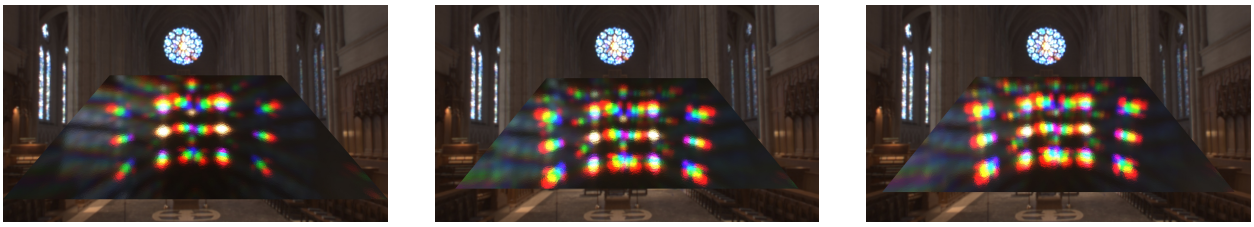
5.2 Fixing bent diffraction lines

A direct convolution with an environment map parametrised in a latitude-longitude format produces diffraction lines that appear correct in the directions close to the view vector and more and more circularly bent as we reach the poles of the sphere (see figure 6) This is a consequence of the assumption $\vec{\omega}_o = \vec{n} = \vec{r}_v$ under spherical illumination.

In order to solve this problem, we compute the convolution in four passes instead of two. We first rotate the environment map by applying the model-view normal matrix (without scaling) to each reflection vector of the environment map, to take into account the rotation of the model and the camera. This way the vector at the center of the environment map is aligned with the surface normal of the object (camera space). We then convolve the environment map in two passes as described in section 5.1. Finally we rotate the result back to the original orientation of the environment map by applying the inverse of model-view normal matrix. The result

Data: EM, Hfilter(rank, width), vec2 currentPixel, Normal in camera space vec3 $\vec{n}$, Tangent in camera space vec3 $\vec{t}$, Bitangent in camera space vec3 $\vec{b}$
Result: EM convolved with the horizontal diffraction filter
1 - Get the reflection vector $\vec{r}_v$ in spherical coordinates at the current pixel of the EM (latitude longiture format).
2 - Convert it to cartesian coordinates and normalize it.
3 - use the approximation $\vec{\omega}_o = \vec{r}_v$ to get the viewing vector in world space.
4 Compute the convolution.
sum = 0.0
for *r in range(0,rank)* **do**
  for *j in range(0,width)* **do**
    4:1 - Get the filter value at position (r,j) : $filterColor$
    4:2 - Convert position (r,j) of the filter to a light direction $\vec{\omega}_i$ using algorithm 2
    4:3 - Convert the normalized light direction to spherical coordinates (angles (θ_i, ϕ_i))
    4:4 - Get the EM color at position (θ_i, ϕ_i)
    4:5 - Compute G with equation 5
    4:6 - Compute : $sum \mathrel{+}= G * filterColor * EMcolor$
    4:7 - Compute : $totalWeight \mathrel{+}= filterColor$
  end
end
sum /= max(totalWeight.R, totalWeight.G, totalWeight.B)
Output: *sum*

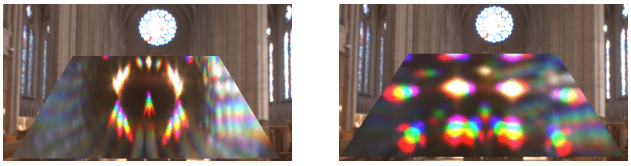
Algorithm 1: Pseudo code to compute the horizontal convolution pass in a fragment shader



(a) (b) (c)

Figure 5: Comparison of the rendering using filters of different ranks. Each rendering was done at the same exposure. (b) Rank 1. 60 FPS. (c) Rank 11. 10 FPS (d) Rank 21. 5 FPS.

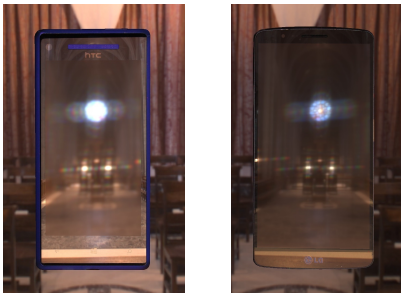
Data: Diffraction filter position (r,j), float scaleH, float scaleV, tangent $\vec{t}$, bitangent $\vec{b}$
Result: Light direction corresponding to the position (r,j) of the diffraction filter in world space
1 - Given the position (r,j) in the filter computes the projection of the half vector in the xy plane (world space). It should be taken into account that the table spans an angular range of $[-scaleH; scaleH] \times [-scaleV; scaleV]$
2 - If the diffraction pattern is not axis aligned apply the inverse affine transform as explained in section 4.2. These correspond to independent rotations applied to the x and y axis.
3 - Rotate the half-vector into the tangential coordinate frame $(\vec{t}, \vec{b})$.
4 - Compute the z component of the half-vector : $h.z = 1.0 + \sqrt{1.0 - (h.x^2 + h.y^2)}$
5 - Normalize $\vec{h}$ and compute the light direction using : $\vec{\omega}_i = \vec{\omega}_o - 2(\vec{\omega}_o.\vec{h})\vec{h}$
Output: Light direction $\vec{\omega}_i$ in world space
Algorithm 2: Pseudo code to compute the light direction corresponding to a given position in the diffraction filter



(a) (b)

Figure 6: Rendering of diffraction at the pole of the sphere. (a) Without the rotations passes. Diffraction lines are bent in circles. (b) With the rotation passes.

can directly be used to render diffraction on the object. Note that the frames per seconds are almost not affected as the two rotation passes are very cheap to compute (a few vector calculations) with current GPUs.



(a) (b)

Figure 7: Rendering of the diffraction pattern of two phones in the Grace Cathedral. (a) HTC 8X. (b) LG G3.

5.3 Results

We present results in the Grace Cathedral environment in figures 1 and 7. In this environment, different reflections can be observed due to point light sources as well as area light sources (windows). The convolutions were computed in a fragment shader for an environment map of size of 3070x1535 for the CD and TV renderings and half of this size for the other renderings. For rough specular materials such as the holographic paper, the convolution can even be computed at a quarter of the 3070x1535 size as they do not require as much details. Choosing the correct environment size is important and is a tradeoff between FPS and rendering quality. The rendering window was set to a size of 1920x1080. We computed the results on a desktop computer with an Intel Xeon 3.70 GHz CPU and a NVIDIA GTX 1080 GPU. The reflectance maps of the planar phones and CD were acquired using second order gradient illumination projected from an LCD panel [6] and the normal map of the holographic paper was captured using Wang et al. [20] step edge illumination method. We also compare our renderings to Toisoul&Ghosh [19] prefiltering method in figure 8. Our method produces correct rotations of the diffraction pattern for rotations of the sample about the view vector.

Object	FPS	Convolution Type	kernel size
HTC 8X	56	Composition	256
LG G3	59	Composition	256
Holographic paper	50	Composition	256
TV	88	Sum	256
CD	100	1D	512

(a) (b)

Figure 8: Comparison of an in plane rotation of the LG phone (a) Our method. (b) Prefiltered method of Toisoul & Ghosh [19] (incorrect result).

Table 1: Summary of the average performance obtained for the rendering in the Grace Cathedral environment.

Depending on the diffraction pattern, the convolution was computed differently. The filtering of the EM for the phone and holographic paper is calculated in two passes (a vertical followed by a horizontal pass), denoted as 'Composition' in table 1, as they have diffraction lobes outside the main directions of the pattern. The filtering for the TV is modelled with the sum of two directional blurs : one in the top right diagonal and another in the bottom right diagonal. Finally the filtering for the CD is a one dimensional convolution in the direction of the local tangents of the grooves of the CD (using Toisoul & Ghosh [19] lookup table). As a result the TV and CD shaders can be computed in a single pass which leads to higher framerates. Rendering the diffraction on a CD was not possible using Toisoul & Ghosh [19] prefiltering method as it would require prefiltering the environment map for every possible rotation of the local tangent coordinate frame. The overall performance for each rendering is summarized in table 1.

The rank and size of the diffraction kernel also has an impact on the performance. We found that using a size of 256 pixel and a rank 1 factorisation is a good compromise to get renderings without artefacts while keeping a high framerate. The frame per seconds goes from 60 to 30 to 10 and 5 for convolutions with ranks 1, 3 10 and 21 respectively but the rendering does not improve much (see figure 5). This is partially due to the fact that the artefacts that are visible in a low rank factorisation are blurred out when convolving with the environment map.

6 CONCLUSION

We presented a method to compute realistic real-time renderings of diffraction effects in static and distant arbitrary environments using a low rank factorisation of the diffraction lookup table. The method works for a wide range of diffraction patterns of commonly manufactured materials. A rank 1 factorisation is the best compromise between realism and high framerate on current GPUs, but future GPUs will very likely be able to filter an environment map with higher rank matrices at high framerates for improved realism.

7 ACKNOWLEDGEMENTS

This work was supported by an EPSRC Early Career Fellowship EP/N006259/1 and a Royal Society Wolfson Research Merit Award.

We also gratefully acknowledge the support of EPSRC Centre for Doctoral Training in High Performance Embedded and Distributed Systems (HiPEDS, Grant Reference EP/L016796/1).

REFERENCES

[1] Emmanuel Agu and Francis S. Hill Jr. 2002. Diffraction Shading Models For Iridescent Surfaces. In *Proc. IASTED VIIP*.

[2] Laurent Belcour and Pascal Barla. 2017. A Practical Extension to Microfacet Theory for the Modeling of Varying Iridescence. *ACM Transactions on Graphics* 36, 4 (2017), 65.

[3] J.M. Cowley. 1995. *Diffraction Physics*. Elsevier Science. https://books.google.co.uk/books?id=S3ekKcs2QoIC

[4] Tom Cuypers, Tom Haber, Philippe Bekaert, Se Baek Oh, and Ramesh Raskar. 2012. Reflectance Model for Diffraction. *ACM Trans. Graph.* 31, 5, Article 122 (Sept. 2012), 11 pages. https://doi.org/10.1145/2231816.2231820

[5] D.S. Dhillon, J. Teyssier, M. Single, I. Gaponenko, M.C. Milinkovitch, and M. Zwicker. 2014. Interactive Diffraction from Biological Nanostructures. *Comput. Graph. Forum* 33, 8 (Dec. 2014), 177–188. https://doi.org/10.1111/cgf.12425

[6] Abhijeet Ghosh, Tongbo Chen, Pieter Peers, Cyrus A. Wilson, and Paul Debevec. 2009. Estimating Specular Roughness and Anisotropy from Second Order Spherical Gradient Illumination. *Computer Graphics Forum* 28, 4 (2009), 1161–1170. https://doi.org/10.1111/j.1467-8659.2009.01493.x

[7] Nicolas Holzschuch and Romain Pacanowski. 2017. A two-scale microfacet reflectance model combining reflection and diffraction. *ACM Transactions on Graphics* 36, 4 (2017), 12.

[8] James T. Kajiya. 1986. The Rendering Equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '86)*. ACM, New York, NY, USA, 143–150. https://doi.org/10.1145/15922.15902

[9] Young-Min Kang, Do-Hoon Lee, and Hwan-Gue Cho. 2015. Multipeak Anisotropic Microfacet Model for Iridescent Surfaces. *Multimedia Tools Appl.* 74, 16 (Aug. 2015), 6229–6242. https://doi.org/10.1007/s11042-014-2092-1

[10] Brian Karis and Epic Games. 2013. Real Shading in Unreal Engine 4. *part of "Physically Based Shading in Theory and Practice," SIGGRAPH* (2013).

[11] Jan Kautz and Michael D. McCool. 1999. Interactive Rendering with Arbitrary BRDFs Using Separable Approximations. In *Proceedings of the 10th Eurographics Conference on Rendering (EGWR'99)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 247–260. https://doi.org/10.2312/EGWR/EGWR99/247-260

[12] Jan Kautz, Pere-Pau Vázquez, Wolfgang Heidrich, and Hans-Peter Seidel. 2000. Unified Approach to Prefiltered Environment Maps. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*. Springer-Verlag, London, UK, UK, 185–196. http://dl.acm.org/citation.cfm?id=647652.732274

[13] Clifford Lindsay and Emmanuel Agu. 2006. Physically-based Real-time Diffraction Using Spherical Harmonics. In *Proceedings of the Second International Conference on Advances in Visual Computing - Volume Part I (ISVC'06)*. Springer-Verlag, Berlin, Heidelberg, 505–517. https://doi.org/10.1007/11919476_51

[14] Ivan Markovsky. 2008. Structured low-rank approximation and its applications. *Automatica* 44, 4 (2008), 891–909.

[15] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. 2003. A data-driven reflectance model. In *ACM TOG*. 759–769.

[16] Christophe Schlick. 1994. An Inexpensive BRDF Model for Physically-based Rendering. In *Computer graphics forum*, Vol. 13. Wiley Online Library, 233–246.

[17] Jos Stam. 1999. Diffraction Shaders. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 101–110. https://doi.org/10.1145/311535.311546

[18] Yinlong Sun, F. David Fracchia, Mark S. Drew, and Thomas W. Calvert. 2000. Rendering Iridescent Colors of Optical Disks. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*. Springer-Verlag, London, UK, UK, 341–352. http://dl.acm.org/citation.cfm?id=647652.732138

[19] A Toisoul and A Ghosh. 2017. Practical acquisition and rendering of diffraction effects in surface reflectance. *ACM Transactions on Graphics* 36, 5 (2017). https://doi.org/10.1145/3012001

[20] Chun-Po Wang, Noah Snavely, and Steve Marschner. 2011. Estimating Dual-scale Properties of Glossy Surfaces from Step-edge Lighting. *ACM Trans. Graph.* 30, 6, Article 172 (Dec. 2011), 12 pages. https://doi.org/10.1145/2070781.2024206