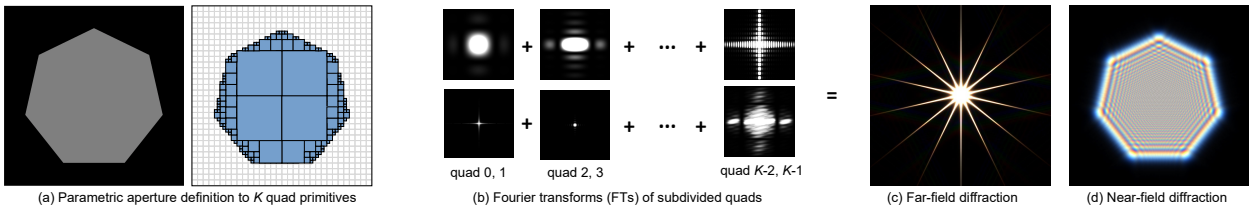
Eurographics Symposium on Rendering 2018
T. Hachisuka and W. Jakob
(Guest Editors)

Volume 37 (2018), Number 4

Quad-Based Fourier Transform for Efficient Diffraction Synthesis

Leonardo Scandolo[†1], Sungkil Lee[‡2] and Elmar Eisemann[§1]

[1]Delft University of Technology, Netherlands [2]Sungkyunkwan University, South Korea



(a) Parametric aperture definition to K quad primitives (b) Fourier transforms (FTs) of subdivided quads (c) Far-field diffraction (d) Near-field diffraction

Figure 1: *Given an aperture image, we subdivide it into a list of parametric quad primitives forming a quadtree (a). The Fourier transform of each quad is efficiently evaluated with a closed-form solution (b), and combined to render the final far-field diffraction (c). We extend this approach to also produce near-field diffraction (d). (c) and (d) are rendered in 0.85 ms and 2.1 ms at* 1024×1024 *resolution, respectively.*

Abstract

Far-field diffraction can be evaluated using the Discrete Fourier Transform (DFT) in image space but it is costly due to its dense sampling. We propose a technique based on a closed-form solution of the continuous Fourier transform for simple vector primitives (quads) and propose a hierarchical and progressive evaluation to achieve real-time performance. Our method is able to simulate diffraction effects in optical systems and can handle varying visibility due to dynamic light sources. Furthermore, it seamlessly extends to near-field diffraction. We show the benefit of our solution in various applications, including realistic real-time glare and bloom rendering.

1. Introduction

Diffraction phenomena can be observed when taking a photo of a strong light source, which typically results in bursting streaks (or glares) [SSZG95, KMN*04, RIF*09, HESL11]. While often used for artistic purposes, these effects also increase the perceived brightness of light sources [SSZG95, YIMS08] and are important elements of realistic rendering beyond typical ray optics.

By definition, diffraction refers to the interference and superposition of spherical waves propagating around obstructions (i.e., apertures) in the path of light. Fresnel's *diffraction integral* [PW15], based on Huygens' principle of wavelet superposition, mathematically models diffraction. In practice, additional geometric approximations (Fresnel and Fraunhofer approximations) can be applied for distant observation planes (at near and far field).

Near/far-field diffraction has typically been computed by formulating the problem as a Fourier Transform (FT) and resorting to optimized FT implementations [KMN*04], usually relying on an image-space Discrete FT (DFT) under the assumption of a periodic function. Given a 2D sampling resolution of $M \times N$, the per-pixel time complexity of the DFT is $O(MN)$. Separable integration and Fast DFT (FFT) are more efficient alternatives but still costly and oriented mostly to offline processing.

While many applications use static aperture patterns, a faster simulation of diffraction can enable applications to capture physical details caused by dynamic aperture changes from area lights, stray lights, and optical elements. However, real-time applications have hardly used dynamic diffraction due to the low FT performance.

Our key observation to accelerate diffraction computations is that the optical aperture image is relatively simple in its color and shape; real light sources are strong (mostly white) and iris apertures are polygon-like. Consequently, we opt for a geometrical representation of the aperture and formulate a closed-form solution for its shape. Similar ideas have been applied in electromagnetics and optics for polygon support and specific patterns [YF74, LM83, MS91] but no generalization for rendering exists. While similar in spirit, our solution is generalizable, flexible, and parallelizable. Specifically, we present an efficient diffraction rendering technique. Instead of

† l.scandolo@tudelft.nl
‡ sungkil@skku.edu (corresponding author)
§ e.eisemann@tudelft.nl

© 2018 The Author(s)
Computer Graphics Forum © 2018 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

DFT, we combine closed-form FT of vector primitives in the continuous domain. At its basis, our approach uses *quads* with constant intensity as such vector primitives and builds upon their closed-form solution to the diffraction integral. We show that quads are an efficient basis even for input images with complex shapes, since we use a hierarchical tessellation. Contrary to previous solutions, the complexity of our transformation relies on the number of quads (tessellating an aperture) instead of the image resolution, which proves typically advantageous. We also introduce a progressive refinement, which exploits spatiotemporal coherence for incremental updates. Our method is suitable for graphics-processing-unit (GPU) execution, enabling interactive and dynamic diffraction rendering. Besides achieving a high-quality far-field approximation, our approach can be extended to a high-quality near-field approximation as well.

Precisely, we make the following major contributions:

- an efficient rendering scheme of a quad-based FT for far-field diffraction;
- a hierarchical technique for diffraction of arbitrary apertures;
- a progressive refinement to exploit spatiotemporal coherence;
- several performance optimizations for real-time rates;
- a seamless extension to near-field diffraction.

2. Related Work

2.1. Diffraction Modeling and Rendering

Diffraction and wave optics in computer graphics are used mostly for glare rendering and reflectance modeling.

Glare rendering relates to the properties of an optical system with an aperture for strong light sources. Early studies to reproduce the patterns used analytic approximations where diffraction gratings are mapped to streaks [NKON90]. It is simple but has limited modeling power for arbitrary apertures. Kakimoto et al. introduced a practical rendering framework and improved accuracy via the FT-based diffraction integral [KMN*04], which was later used for lens flares [HESL11]. Other work attempted to model glares in the human visual system [Rok93, SSZG95, RIF*09]. Spencer et al. intensively discuss physiological glare components and their causes [SSZG95]. Additional modeling (e.g., light scattering) and temporal fluctuation were introduced for higher perceptual accuracy [RIF*09].

The reflectance modeling to spectral interference has focused on bidirectional reflectance functions of surface microstructures. An effective diffraction analysis was proposed (avoiding the evaluation of Kirchhoff integral) for the microstructure of surfaces (e.g., a compact disc) [Sta99]. Further efficiency can be obtained with spherical harmonics and the Chebyshev approximation [LA06, DG16]. A high-quality solution to generate multiple scattering effects was introduced using destructive interference [CHB*12]. Recently, many studies have extended microfacet models to simulate iridescence from scratch [WVJH17], metal surfaces [DWMG15], thin-film coating [BB17], often integrating data-driven acquisition/rendering [TG17] or a two-scale geometry model [HP17].

In the previous work, simple texturing/billboard or iterative filtering are common in interactive applications [Rok93, Oat04], but high-quality modeling relies on FT [Sta99, KMN*04, RIF*09]. The FT results are stored in lookup tables as Point-Spread Functions (PSFs) for online usage due to its cost. In contrast, our work accelerates the FT itself, which enables real-time evaluation and applications.

2.2. Acceleration Techniques for Fourier Transform

A FT of digital images typically relies on numerical integration. FFT [CT65] is typically applied when the sampling interval is uniform over input/output and the signal is periodic. For general cases, FFT is currently the most efficient solution and recent work focuses on optimizations. FFTW is an optimal CPU implementation [FJ98] and parallel versions of FFT have received attention due to the increased throughput on modern hardware [MA03, GLD*08, FPV*09].

A substantial amount of literature exists for near-optimal or approximate DFTs, attaining sublinear complexity. We refer the reader to [HIKP12] for a survey. A notable attempt is Sparse FT [HIKP12], which excludes negligible coefficients from the FT evaluation, leading to an approximate but high-quality result.

For simple input functions, the continuous FT can have a closed-form solution. The function can be either piecewise constant [Sor95] or piecewise discontinuous [FL04, LNL08]. Several parametric functions admit closed-form solutions, including sunburst patterns [YF74], polygonal shapes [LM83, MS91], and triangular meshes [HCL91, LLNZ11]. The majority of such closed-form solutions are applied for electromagnetic analyses or the detection of diffraction [LHP17]. Rendering of dispersion and near-field ringing has hardly been explored with this direction.

Our work introduces and extends concepts for accelerated FT computation to rendering for real-time diffraction effects. We exploit that large constant areas in an image are quite common for diffraction rendering; e.g., an area light source or an aperture. Quad primitives approximate such regions well and lead to GPU-friendly light-weight computations, resulting in good performance. We also propose a hierarchical solution and progressive refinement that lead to additional acceleration and enable dynamic visibility integration. These concepts are difficult to couple with general supports, such as triangles [HCL91, LLNZ11], where the performance benefits of closed-form solutions can be lost.

3. Background

We first revisit the standard FT and its formulation for diffraction before introducing our contributions.

3.1. Standard Fourier Transform

Given a function f in the spatial domain, the FT operator $\mathcal{F}$ defines $\mathcal{F}(f)$ as a continuous integration over the spatial domain:

$$\mathcal{F}(f)(u,v) = F(u,v) = \iint_{-\infty}^{\infty} f(x,y)h(ux+vy)dxdy, \qquad (1)$$

where $i = \sqrt{-1}$, $h(x) := e^{-i2\pi x}$, and (u,v) is a frequency-domain position. If f is regularly sampled and periodic, its DFT is:

$$F(u,v) \approx \frac{1}{\sqrt{MN}} \sum_{y=0}^{N-1} \sum_{x=0}^{M-1} f(x,y)h\left(\frac{ux}{M} + \frac{vy}{N}\right), \qquad (2)$$

where M,N are the sampling resolutions over x,y respectively. An effective DFT evaluation relies on FFT, reducing the per-pixel complexity to $O(\log M + \log N)$, but is still costly in a real-time context.

© 2018 The Author(s)
Computer Graphics Forum © 2018 The Eurographics Association and John Wiley & Sons Ltd.

3.2. Diffraction with Fourier Transform

Given the source aperture plane position $\mathbf{p} := (x,y)$ and the destination position (x',y') at the observation plane at distance z, the Huygens-Fresnel equation describes diffraction as:

$$D(x',y',\lambda) = -\frac{i}{\lambda}\iint f(x,y)\frac{e^{ikR}}{R}dxdy, \tag{3}$$

where λ is a wavelength of light, $k = 2\pi/\lambda$, and $R = ((x'-x)^2 + (y'-y)^2 + z^2)^{1/2}$ [PW15]. The power spectrum of $D(x',y',\lambda)$ corresponds to the intensity that we wish to visualize in the final image. Computational efficiency can be increased by using Fresnel's approximation for Near-Field Diffraction (NFD) [PW15]:

$$D(x',y',\lambda) \propto \frac{1}{z\lambda}\iint f(x,y)h(-\frac{x^2+y^2}{2z\lambda})h(\frac{xx'+yy'}{z\lambda})dxdy. \tag{4}$$

Note that we drop a phase term as it is irrelevant to the power spectrum. When the destination plane is assumed to be far (i.e, $(x^2+y^2) \ll z\lambda$), Far-Field Diffraction (FFD) becomes:

$$D(x',y',\lambda) \propto \frac{1}{z\lambda}\iint f(x,y)h(\frac{xx'+yy'}{z\lambda})dxdy, \tag{5}$$

When we let $(u,v) = (x'/z\lambda, y'/z\lambda)$, FFD becomes a scaled FT:

$$D(x',y',\lambda) \propto F(u,v)/(z\lambda), \tag{6}$$

where u and v can be considered spatial frequencies [PW15].

4. Our approach

Our approach uses axis-aligned *quad* primitives with *constant* intensity. Quads allow us to efficiently approximate arbitrary shapes via tessellation into a set of disjoint quads. Given that our input is a digital image, which is by definition composed of small quads/pixels, the constancy assumption does not restrict our solution. While different shapes can be used as primitives, obtaining a tessellation of the input image into more complex shapes becomes a bottleneck. Conversely, quads lend themselves well to a hierarchical representation, which reduces their number, and results in an efficient implementation.

We first introduce our novel approach on a primitive-based FT and specialize it for quad primitives (Sec. 4.1). Then, we present our approach to render FFD effects (Sec. 4.2). In this context, we will also describe the efficient tessellation of the input into a hierarchical representation. We then explain accelerations to our solution (Sec. 4.3) and show how to integrate occlusion and area lights (Sec. 4.4). Finally, we extend our approach to NFD (Sec. 4.5).

4.1. Primitive-based Fourier Transform

Here, we introduce our primitive-based FT, which we use for efficient diffraction synthesis. Fundamental to our method is the reformulation of the image-space FT. The idea is to decompose the input signal into a sum of primitives (e.g., polygons). The primitive-based FT then uses the superposition principle, which states that the FT of each primitive can be computed independently:

$$\mathcal{F}(af(x)+bf(y)) = a\mathcal{F}(f(x))+b\mathcal{F}(f(y)). \tag{7}$$

The superposition principle has also been shown for diffraction in optics (e.g., Babinet's principle) [BW13, PW15].

Initially, we assume all primitives are disjoint, i.e., the original signal is a union of all primitives without intersection. Then, the FT can be computed as a sum:

$$F(\mathbf{q}) = \sum_{k\in\mathcal{K}} W_k(\mathbf{q}), \tag{8}$$

where $\mathbf{q} := (u,v)$, $\mathcal{K}$ is the set of primitives, and W_k the transform of the primitive k. To compute W_k, we assume k is given by a function $f_k(\mathbf{p})$ and a domain Ω_k, typically bounded in $\mathbb{R}^2$. We then obtain:

$$W_k(\mathbf{q}) = \mathcal{F}(f_k)(\mathbf{q}) = \int_{\Omega_k} f_k(\mathbf{p})h(\mathbf{p}\cdot\mathbf{q})d\mathbf{p}. \tag{9}$$

Quad-based Closed-Form Solution Axis-aligned quads with constant intensity are easy-to-integrate, and result in an efficient and concise closed-form solution. Additionally, we show that using the shift property of the FT, we can achieve further acceleration.

A quad k with constant intensity I_k is defined as the inside of a bounded rectangular domain $\Omega_k = \{(x,y) \in \mathbb{R}^2 |\ |x-c_x| \le s_x/2,\ |y-c_y| \le s_y/2\}$, which is centered at $\mathbf{c}_k = (c_x,c_y)$ with a size of $\mathbf{s}_k = (s_x,s_y)$. Since the function f_k of k is separable along x and y axes, f_k is defined as a tensor product of boxcar functions:

$$f_k(x,y) = I_k\ (\Pi_{c_x,s_x} \otimes \Pi_{c_y,s_y})(x,y) = I_k\ \Pi_{c_x,s_x}(x)\Pi_{c_y,s_y}(y), \tag{10}$$

where the boxcar function is defined as $\Pi_{c,s}(x) = H(x-c+s/2) - H(x-c-s/2)$, and $H(x)$ is the Heaviside step function. The closed-form FT of the boxcar function $\Pi_{c,s}(x)$ is given as:

$$\Gamma_{c,s}(u) = \mathcal{F}(\Pi_{c,s}(x))(u) = s\ \mathrm{sinc}(\pi us)h(c), \tag{11}$$

where $\mathrm{sinc}(x) = \sin(x)/x$. Then, we can obtain the closed-form solution to $W_k(\mathbf{q})$ as a tensor product of Γ:

$$W_k(\mathbf{q}) = I_k\left(\Gamma_{c_x,s_x} \otimes \Gamma_{c_y,s_y}\right)(\mathbf{q}). \tag{12}$$

While this evaluation is simple for a single quad, having several quads leads to many redundant computations. We can reduce the number of computations using the shift property, which states:

$$W_m(\mathbf{q}) = h(\mathbf{d}\cdot\mathbf{q})W_k(\mathbf{q}). \tag{13}$$

This holds for two quads k and m of the same spatial support size, with a displacement $\mathbf{d}$, such that $f_m(\mathbf{p}) = f_k(\mathbf{p}+\mathbf{d})$. Hence, we can reuse the FT of a single quad for the FTs of all equally-sized quads.

Rendering the object-space FT The rendering is straightforward. For a given list of quads, we evaluate the FT for each pixel using Eqs. (8) and (12), efficiently on the GPU. We employ Eq. (13) to compute the quad FT only once.

4.2. Far-Field Diffraction Rendering

Here, we introduce the application of the previous analysis to achieve efficient FFD effects. We describe the approach for a point light source and how to decompose the aperture into a small set of quads to achieve a fast diffraction computation.

Principle Given a point light source and a camera with an aperture and image plane, we can produce a realistic diffraction pattern in real time. As pointed out in Sec. 3, assuming that the aperture is given in the form of a binary image, its FT will yield the desired diffraction pattern. In principle, we could treat each pixel of the aperture image

© 2018 The Author(s)
Computer Graphics Forum © 2018 The Eurographics Association and John Wiley & Sons Ltd.

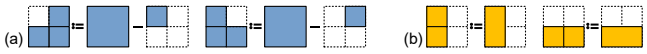


Figure 2: *Examples of the non-uniform quadtree subdivision: subtractive superposition (a) and non-square rectangles (b).*

as a quad, collect them in a list, and apply the solution from the previous section. Nevertheless, as the compute time is governed by the quads, it is beneficial to minimize their number.

Hiearchical Decomposition of Aperture and Rendering To reduce the number of quads, we use a *quadtree*, which hierarchically tessellates the aperture (Figure 1) for a compact representation.

We perform a bottom-up quadtree construction, which is fast to compute. The process is initiated from the binary aperture image and proceeds from mipmap to mipmap level. When four child texels have the same value, we consider them merged, write their common value on the next mipmap level and proceed. Otherwise, we append quads to the list of quads to transform. Contrary to a typical quadtree, we introduce a special non-uniform subdivision procedure (Figure 2). Specifically, if only one quad has a value of one, we attach this single quad to the list and write a zero in the next level. If two adjacent quads have a value of one, a merged quad is appended (Figure 2b). Otherwise, both are appended. If three quads have a value of one, we write a one on the next level and attach a negative quad to the list (Figure 2a), which has the support of the quad with value zero but will be subtracted after transformation.

During the evaluation of the quad list, we employ the shift property. For each quad shape and size, we compute its FT only once, and apply Eq. (13) for congruent quads. Hereby, we accelerate the FT evaluation without reducing quality.

4.3. Accelerations

Culling and Symmetry An important observation is that a significant portion of the FT output contains zeros (in terms of the power spectrum) and skipping these pixels would accelerate computations. Testing for zeros in the power spectrum at full resolution would be too expensive. Instead, we use a lower-resolution image (in practice $1/8^2$) and additionally test if the *analytical* derivatives of the power-spectrum of the FT are zero. While not strictly conservative, the continuity of the derivatives leads to a very good estimate and we found no differences with respect to the actual zero test at full HD resolution. The previous pixel-based FTs lack analytical derivatives and cannot profit from such an acceleration.

Additionally, we exploit mirror symmetry in the FT. We evaluate only the half-plane and mirror it, obtaining the other half for free. When the input image has additional symmetries, which are common for apertures, we can exploit these as well.

Progressive Refinement Another strong benefit of the object-space transformation (Eq. (8)) is the use of *progressive* refinement. We capture *dynamic* spatio-temporal changes, which significantly distinguishes our solution from the pixel-based FTs. When the input apertures show coherence, instead of recreating the FT from scratch, we update the FT by exploiting the superposition principle. To this extent, we derive a quad list that represents the difference between

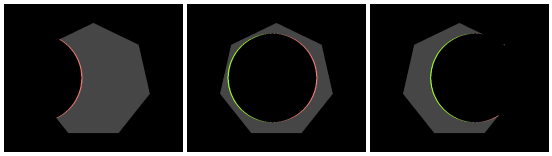


Figure 3: *Example frames of a moving circle occluding a heptagonal aperture. Our progressive method can update the FT for the entire aperture by adding/subtracting only the FTs of newly disoccluded/occluded (green/red) quads to/from the previous calculation.*

the current and previous aperture. We perform a bottom-up procedure on this difference (Figure 3), where we extract all positive and negative quads. This procedure is similar to the original extraction, but we deal with the positive and negative parts independently in different texture channels, which allows us to apply the same merging strategies as before without making the quadtree creation procedure more complex. Progressive refinement proves very beneficial for animation, since typically only a reduced number of quads are needed to update the FT. Additionally, it is highly important when including visibility changes to the aperture shape (Sec. 4.4).

Accelerated Spectral Integration So far, we have described a method for computing the monochromatic diffraction pattern for a single wavelength. For visible diffractions, there is usually a need to cover the whole visible spectra. For an accelerated spectral integration, we extend that introduced in [HESL11].

We start with a reference wavelength λ_r (e.g. 587.6 nm), for which we compute its FT F_{λ_r}. Each different λ gives a relative wavelength scale $\rho(\lambda) = \lambda_r/\lambda$. Given the observation distance z and defining $u_r = x'/z\lambda_r$ and $v_r = y'/z\lambda_r$, the diffraction for λ can be expressed in terms of the diffraction of λ_r as:

$$D(x',y',\lambda) \propto \frac{1}{z\lambda} F_\lambda(\mathbf{q}) = \frac{1}{z\lambda_r}\rho(\lambda) F_{\lambda_r}(\rho(\lambda)\mathbf{q}). \tag{14}$$

The required spectral samples correspond to scaled texture locations ($\rho(\lambda)\mathbf{q}$ in Eq. (14)), lying on the same line passing through $\mathbf{q}$ and the origin. For a fixed z, $1/(z\lambda_r)$ is a constant. Using Eq. (14), we establish a relation between the color pattern S_k created by a quad k:

$$\begin{aligned} S_k(\mathbf{q}) \propto L_k(\mathbf{q}) &= \int_{\lambda\in\Lambda} \mathcal{X}(\lambda)\|F_\lambda(\mathbf{q})\|^2 d\lambda \\ &= \int_{\lambda\in\Lambda} \mathcal{X}(\lambda)\rho(\lambda)^2\|F_{\lambda_r}(\rho(\lambda)\mathbf{q})\|^2 d\lambda, \end{aligned} \tag{15}$$

where $\|F\|^2$ is the power spectrum of the FT, Λ is the spectral domain, $\mathcal{X}(\lambda)$ is a normalized spectral response function for a particular chromaticity channel (e.g., X, Y, or Z in XYZ color space), and the scale relates to the exposure time. In general, $\mathcal{X}(\lambda)$ is empirically defined in terms of piecewise measurements; a recent analytic approximation exists [WSS13] but does not lead to a closed-form integration. Hence, we numerically integrate it via:

$$L_k(\mathbf{q}) \approx \sum_{\lambda\in\Lambda'} \underbrace{\mathcal{X}(\lambda)\rho(\lambda)^2}_{\text{dispersive weights}} \|F_{\lambda_r}(\rho(\lambda)\mathbf{q})\|^2 d\lambda, \tag{16}$$

where Λ' is a discrete subset of Λ in a finite visible spectral range, and $\sum_{\lambda\in\Lambda'} \mathcal{X}(\lambda)d\lambda = 1$.

While the scaling-based integration (Eq. (16)) avoids brute-force

© 2018 The Author(s)
Computer Graphics Forum © 2018 The Eurographics Association and John Wiley & Sons Ltd.

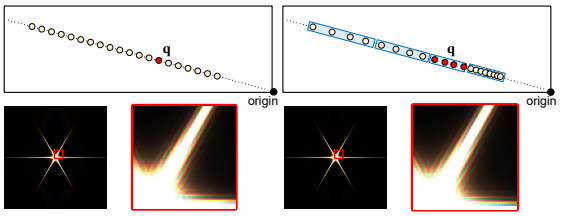


Figure 4: *Uniform spectral scaling (left) and 4-channel batch spectral scaling (right), with examples for the diffraction of an hexagonal aperture. Our batch scaling saves the amount of lookups from the base pattern by roughly a factor of four.*

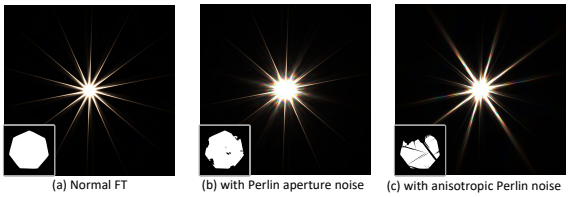


Figure 5: *Examples of diffraction patterns without noise (a) and with the addition of Perlin noise (b) and anisotropic noise (c).*

spectral sampling, many samples (e.g., > 60) are needed to avoid spectral aliasing. Therefore, we propose *batch-scaling* using an intermediate four-channel texture. In this texture, we store four nearby samples of $\|F_{\lambda_r}\|^2$ along the scaling line $\rho(\lambda)\mathbf{q}$ in a RGBA quadruplet. In a second pass, we sparsely sample this texture along the same line recovering four samples at a time, which are then scaled by the dispersive weights (Eq. (16)). In this way, we only need a quarter of the original texture lookups. As reference wavelength λ_r, we use the geometric mean of the extrema of the visible wavelength range (e.g., 529 nm for $[400, 700]$ nm). Using batch-sampling leads to non-uniform spacing towards the extrema but the difference to uniform sampling is marginal, as seen in Figure 4.

4.4. Occlusion and Area Lights

Up to now, we have ignored any kinds of occlusions additional to the aperture shape. Given the efficiency of our approach, we can integrate dynamic visibility changes in the scene.

One simple type of occlusion results from imperfections such as floating dust particles on lenses. These occlusions could be approximately handled by integrating perturbation of the input aperture pattern. We can use arbitrary noise patterns but Perlin noise offers spatiotemporal coherence for animation (Figure 5).

For more structural occlusions due to a blocker, our approach can show major advantages. In this case, we can render the aperture as seen from the light source and project the scene geometry on top of it to derive the unblocked part of the aperture. The resulting aperture image can then be transformed with our approach. In this scenario, the progressive refinement proves particularly useful.

So far, we have assumed a point light source and obtained only the Point-Spread Function (PSF) of the resulting aperture image. In reality, light sources cover an area. Consequently, we convolve diffraction patterns over this area by using them as PSFs [RIF*09].

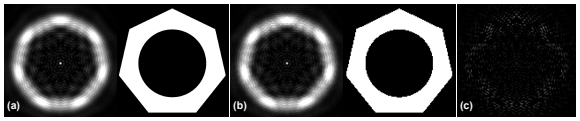


Figure 6: *Near-field diffraction pattern of a partially occluded aperture using a quadtree resolution of 1024^2 (a) and 128^2 (b), and a $5\times$ difference visualization (c). Both results are very similar but the lower quadtree resolution results in a $\sim 7\times$ speedup.*

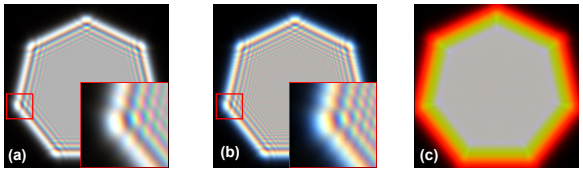


Figure 7: *Spectral integration (over 60 wavelengths) of the NFD pattern (a) cannot be approximated as done for the FFD (c). However, a heuristically chosen scaling factor (here, $1+0.05(1-\rho)$) still can produce a plausible approximation (b).*

However, when integrating visibility, each point in the area light source may exhibit different viewing conditions, impacting the diffraction pattern. Here, for each point sample, we need to render the scene towards the aperture. For large area lights, this step remains costly. Although we did not investigate this direction, image-warping approximations and solutions such as [HREB11] can be used. As a cheaper alternative, we could also consider a small neighborhood in the scene image around each light sample and intersect the light-source pixels with the aperture image. In this way, we approximate a projection of the aperture from the image plane into the scene and onto the light. Our approximate projection is meaningful under the assumption that all scene geometry is closer to the aperture than the light source. When the geometry is fairly far from the aperture, the quality of the resulting approximation is reduced.

4.5. Near-Field Diffraction

Having derived FFD patterns (Eq. (5)), we also seamlessly extend our work to NFD, where the far-field assumption does not hold. NFD is often related to *ringing* at the edges of bokeh patterns or ghosting apertures in lens-flare rendering [Kin75, HESL11].

It is known that the superposition principle also holds for near-field diffraction [PW15], as evidenced by Eq. (4). Consequently, we can compute it as a sum in terms of the NFD of a set of primitives:

$$G(u,v) = \sum_{k\in\mathcal{K}} V_k(u,v), \tag{17}$$

where V_k is the contribution of quad k in the set $\mathcal{K}$. Using Eq. (4) and the constancy assumption, we obtain:

$$V_k(u,v) = (I_k/z\lambda)\left(g_{c_x,s_x}(u)\right)\left(g_{c_y,s_y}(v)\right), \tag{18}$$

where

$$g_{c,s}(t) = \int_{c-s/2}^{c+s/2} h(tx - \frac{1}{2z\lambda}x^2)dx. \tag{19}$$

Unlike the FFD, the integral we need to solve involves square

© 2018 The Author(s)
Computer Graphics Forum © 2018 The Eurographics Association and John Wiley & Sons Ltd.

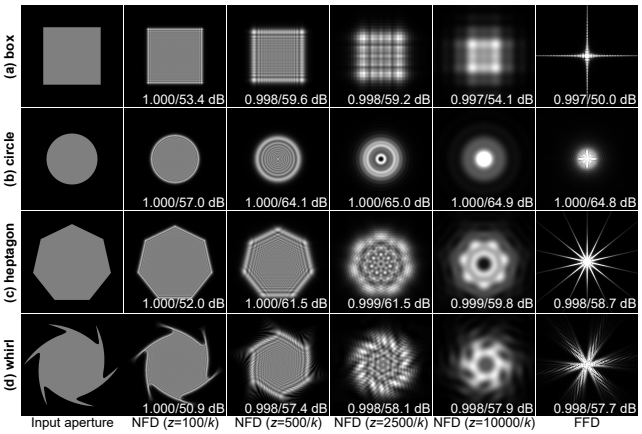


Figure 8: *FFD and NFD outputs (scaled for illustration) for the input aperture images (the first column) used for the experiment. The numbers at the bottom indicate SSIM/PSNR values measured against the references (generated by discrete NFD/FFD solutions).*

terms of x, requiring evaluating Fresnel integrals (see Appendix A). Unfortunately, Fresnel integrals have no analytical solution. Therefore, we use a numerical approximation [Mie98], accelerated by a 1D look-up table of the precomputed Fresnel integrals.

Efficient Rendering Our approach for NFD is more expensive than our FFD solution given that we need to compute the complex-valued Fresnel integrals four times for each quad. Yet, we can observe that the product in Eq. (18) is again separable. To avoid redundant computations, we propose an effective two-pass optimization, such that unique combinations of (k,u) and (k,v) are evaluated only once. In the first pass, for each quad, we precompute $g_{c_x,s_x}(u)$ and $g_{c_y,s_y}(v)$ for all u and v, respectively, and store these results in a texture. Then, in the next pass, for every pixel, $V_k(u,v)$ can be obtained by combining these partial results according to Eq. (18), which reduces the number of evaluations drastically.

For the FFD, we speed up computations by performing a culling pass at a lower resolution, and enabling progressive refinement for dynamic scenes. Our experience shows that the NFD result is not sensitive to high-frequency content for low z values and a coarse quadtree can be used without significant quality loss (Figure 6).

With respect to the spectral integration, the previously-used simple scaling for λ_r cannot be applied (Figure 7c). For correct results, we need to repeat Eq. (18) for dispersive samples (Figure 7a). However, we empirically found that scaling with smaller amounts (e.g., $1+0.05(1-\rho)$) leads to a plausible approximation but the result will no longer be physically accurate (Figure 7b).

5. Results

In this section, we assess performance and quality of our solutions for FFD and NFD and provide several examples of our results.

5.1. Far-Field Diffraction

The experiments used four input aperture shapes; the monochromatic images produced for FFD (and NFD) are shown in Figure 8,

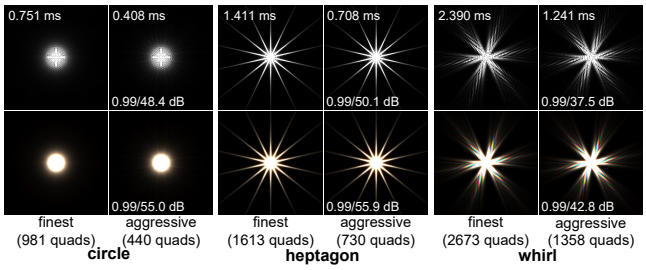


Figure 9: *Comparison of quality in terms of quadtree resolution for monochromatic and dispersive FFDs. The one-step further downsampling from the finest quadtree resolution results in a negligible visual difference (SSIM/PSNR at the bottom), while increasing speed by a factor of two (timings given at the top).*

Table 1: *Number of quads produced at the quadtree tessellation.*

aperture	256^2	512^2	1024^2	2048^2	4096^2
box	40	40	40	617	617
circle	109	227	440	981	1976
heptagon	176	389	730	1613	2996
whirl	303	647	1358	2673	5278

along with quality comparison with respect to the brute-force DFT-based reference solutions. Their resolution scales from 256^2 to 4096^2. Unlike standard DFT techniques, our technique scales with the number of quads, where the degree of tessellation depends on the image content (Table 1); rectangle and circle would already have optimal analytical solutions, but we showcase them as general cases. We initiate the finest mipmap level of the quadtree construction to the half size of the input image to avoid redundancy from anti-aliased boundaries (rasterized from vector images); note that this does not reduce the output resolution, unlike a downsampling for the pixel-based FT techniques. We also use a coarser quarter-size mipmap for a more aggressive quadtree approximation. Their visual differences are marginal; Figure 9 shows the comparison of quality and performance in terms of the quadtree tessellation resolution.

We implemented our solution in OpenGL 4.5 on an Intel i7-5820K with 3.3 GHz and an NVIDIA GeForce GTX 1080 Ti graphics card. We first compare our FFD techniques with different implementations of the standard FT: the well-known FFTW [FJ98] (CPU-FFT), and our GPU implementation of FFT (GPU-FFT); we note that our GPU-FFT is equivalent to those used in [KMN*04, HESL11]. Our techniques use four versions in terms of aggressive tessellation and progressive evaluation; the symmetry and culling optimizations are used in all cases. The progressive evaluation uses animated sequences, as shown in Figure 3; we overlaid a circular shadow on each aperture, which moves at 5 pixels per frame. Then, we report the individual effects of our optimization techniques.

Figure 10 shows the performance comparison of our techniques against CPU-FFT and GPU-FFT for monochromatic FFD. Our solutions are faster than CPU-FFT and GPU-FFT, because our methods evaluate far fewer quads than pixels present in the image. The exception being the Whirl aperture for high resolutions at the finest tessellation level. Using a coarser input quadtree, our solution outperforms the competing methods while producing visually indistinguishable

© 2018 The Author(s)
Computer Graphics Forum © 2018 The Eurographics Association and John Wiley & Sons Ltd.

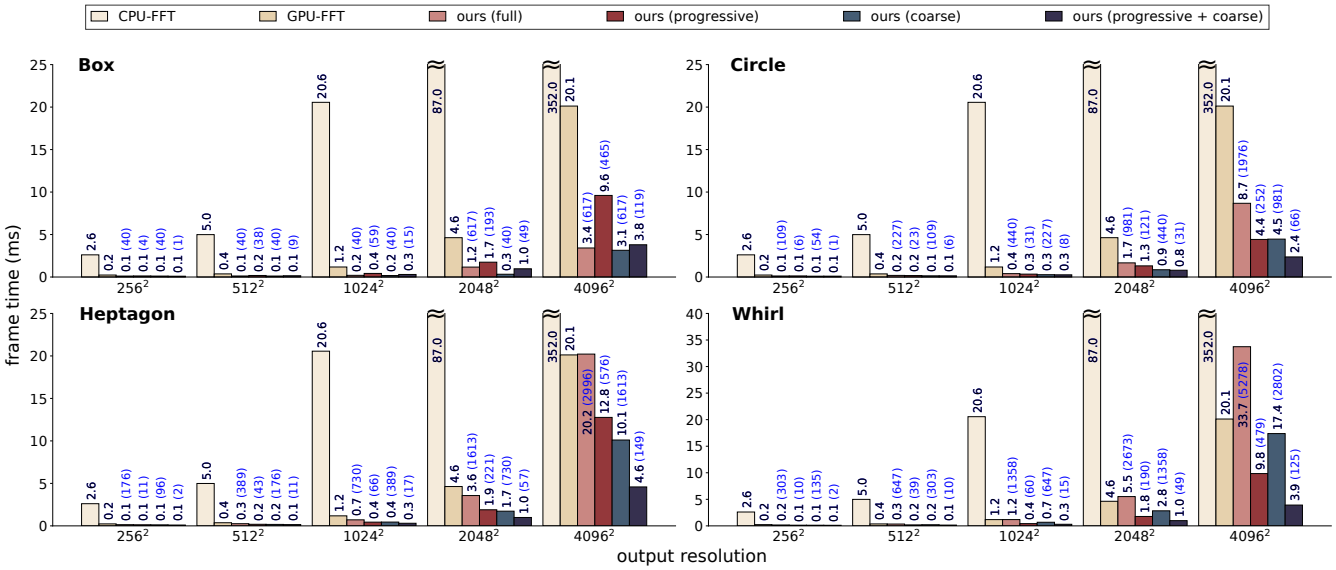


Figure 10: *Timing comparison of our object-based FT methods to the pixel-based FT techniques. Ours include the four versions and the frame times are shown on each bar. For ours, the average numbers of evaluated quads are shown with blue color in parentheses.*

Table 2: *Impact of optimizations on the timing (measured in ms) of the FFD for the Heptagon aperture.*

output resolution	256^2	512^2	1024^2	2048^2	4096^2
no optimizations (ms)	0.13	0.40	2.25	19.46	139.35
culling (ms)	0.21	0.32	1.06	5.65	31.63
symmetry (ms)	0.14	0.25	1.20	9.76	74.70
culling+symmetry (ms)	0.14	0.26	0.71	3.57	20.22

Table 3: *Cost comparison (measured in ms) between the full spectral scaling and our four-sample batch spectral scaling.*

output resolution	256^2	512^2	1024^2	2048^2	4096^2
full (ms)	0.040	0.127	0.530	1.960	8.180
batch (ms)	0.019	0.038	0.114	0.430	1.710

results (Figure 9). Similarly to FFTs, our solutions also scale with the output resolution, but depend more on the input shape. Our progressive refinement significantly reduces the effective number of quads to evaluate without any quality loss; the Box aperture is an exception due to its simple shape. The number of quads are reduced roughly down to 10–20% (see the figure for the average number of the effective quads), and the speedup factors about 2. The aggressive approximation with the coarser quadtree representation gains an additional speedup with marginal quality loss. When combining the progressive evaluation and aggressive approximation, our solution achieves a great speedup in comparison to GPU-FFT (roughly 5× faster). Hence, in practice, we suggest using a coarser resolution for plausible fast rendering. With our object-based approach, it is also easier to decouple the input and output resolutions than it is when using pixel-based FTs.

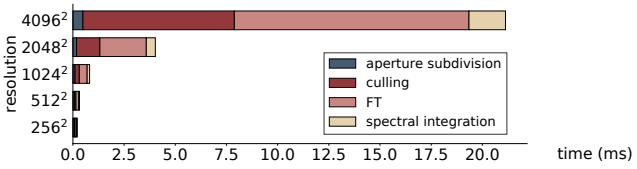


Figure 11: *Per-stage performance breakdown of our FFD rendering for the Heptagon aperture.*

Table 2 shows the performance gain using culling and symmetry. Culling improves performance by up to 4× and symmetry up to 2×. Combining both, we achieve up to roughly a 8× speedup.

Table 3 shows the performance gain of the batch spectral scaling against the full spectral processing. The full processing with scaling requires 60 spectral samples while our batch scaling requires 4+15 spectral samples and indeed performs about 4 times faster.

Figure 11 shows the performance breakdown of the entire rendering pipeline of FFD, which includes the aperture subdivision, culling, monochromatic FT, and spectral integration. We note the noise patterns are optional, for realistic imperfections. Their addition to the main pattern takes less than 0.5 ms at a 4096^2 resolution.

5.2. Near-Field Diffraction

For the same data and the same machine used in Sec. 5.1, we compare our NFD technique against our GPU-based implementation of the reference discrete NFD (GPU-NFD) and a recent CPU-based discrete fractional FT [TLZ10] (CPU-FrFT; written in Matlab); in many cases, the NFD evaluation relies on an optimized FrFT implementation [OZK01]. Unlike our closed-form solution of the *continuous* integrals, the *discrete* techniques may exhibit severe

© 2018 The Author(s)
Computer Graphics Forum © 2018 The Eurographics Association and John Wiley & Sons Ltd.

Table 4: *Performance (measured in ms) comparison of CPU-FrFT [TLZ10], GPU-NFD, and our NFD.*

output resolution	CPU-FrFT	GPU-NFD	Our NFD			
			Box	Circle	Heptagon	Whirl
256^2	15.98	4.87	0.12	0.11	0.14	0.16
512^2	33.01	20.94	0.14	0.23	0.36	0.49
1024^2	128.72	94.73	0.28	0.84	1.71	2.93
2048^2	533.85	—	7.88	5.43	15.78	23.42
4096^2	2133.19	—	30.88	41.30	137.16	255.72

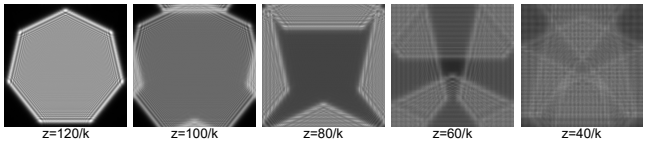


Figure 12: *As z decreases, the discrete NFD without enough zero padding leads to interference patterns. Our NFD solution does not suffer from these artifacts.*

aliasing for near-source distances (see Figure 12), which requires significant zero padding to avoid it. For GPU-NFD, we had to use $8\times$ larger resolutions than the input to completely remove aliasing in our experiments. Thus, in our experiments, GPU-NFD cannot be produced at input resolutions higher than 1024^2 (requiring 8192^2 for the evaluation) due to the hardware limitation. This indicates important benefit of our solution against the discrete techniques, where ours accelerates at a lower resolution and uses less memory.

Our NFD solution uses the two-pass optimization (separable computation of the tensor product), which is roughly $5\times$ faster than those without the optimization. We used culling, but no symmetry. The culling used function-only sampling at a lower resolution, since it is impossible to find the analytic gradient of the NFD power spectrum. Given that the NFD depends less on high frequencies when compared to the FFD, this culling produces no artifacts.

Table 4 shows the comparison of the monochromatic NFD generated with our solution compared to CPU-FrFT and GPU-NFD. Similarly to the FFD techniques, CPU-FrFT and GPU-NFD scale only with the image resolutions, while ours depends on the aperture shapes as well as the image resolutions. For all resolutions, ours are significantly faster than CPU-FrFT and GPU-NFD; speedup factors are 34–$182\times$ for the most complex Whirl aperture. GPU-NFD is also slow due to its significant zero padding, which results in 30–$42\times$ slower performance than our NFD (for the Whirl aperture). Our approach does not suffer from such problems and can be used at any resolution, which also provides more fine-grained control over performance. As expected, CPU-FrFT is the slowest variant.

As shown in Figure 6, the results of the coarse quadtree are less sensitive than those of FFDs. Thus, in practice, a more aggressive downsampling can be used.

Our spectral approximation using linear scaling for NFD is much faster than a precise evaluation with speedup factors in the range of $93.1/244.7$ for a resolution of $256^2/512^2$, Given the marginal quality loss, it is a practical alternative to the full evaluation. The progressive approach has a similar effect as for FFD, which is why we left this evaluation out.

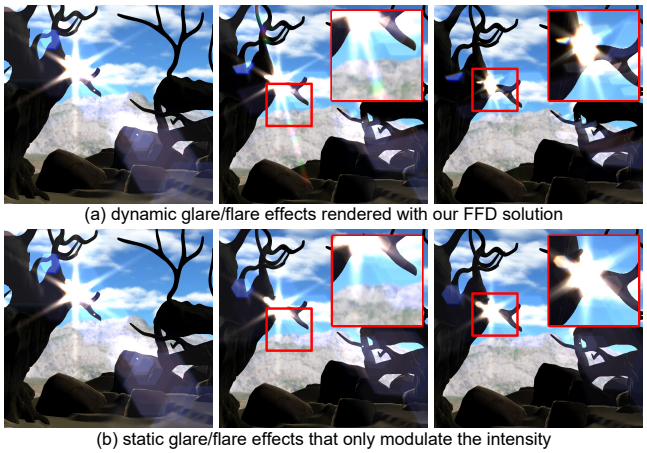


Figure 13: *Comparison of the glare/flare rendering [HESL11] with the static and our dynamic solutions. Our FFD solution (a) for the area-light integration well reflects the varying visibility of the dynamic light source, unlike the static glare effects (b).*

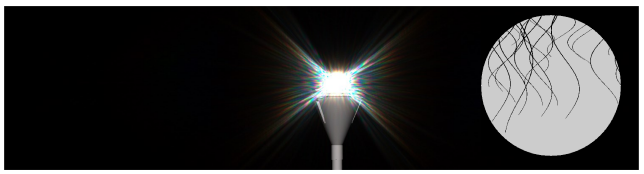


Figure 14: *An example glare rendering with an eyelash aperture.*

6. Applications

6.1. Glare Rendering

Glare rendering is a major application for our FFD and NFD techniques, which can be used to achieve more realistic results when compared to previous techniques that rely on static diffraction patterns [HESL11]. To render dynamic glare, the diffraction pattern (i.e., PSF) is applied only to the relevant parts of the scene. For an area light source, we extract the scene image around the light source, integrate the PSFs of individual light samples with varying occlusions, and blend the result with the scene image.

Figure 13 shows examples rendered for an area-light source with and without dynamic diffraction. The aggressive approximations still result in plausible outcomes, which suits real-time applications well. A variation of glare rendering simulates the effect of eyelashes (Figure 14), which simplifies the previous models of the human visual system [KMN*04, RIF*09]. More extensive modeling involves additional components (e.g., iris, cornea, pupil, vitreous humor, and retina [SSZG95, RIF*09]).

The performance of our area-light integration is roughly $2\times$ faster than that of FFT. In our experiments (81 light point samples at 1024^2 resolution), the timings of ours and FFT for monochromatic outputs are 31.9 ms and 63.4 ms, respectively. In addition, the aperture

© 2018 The Author(s)
Computer Graphics Forum © 2018 The Eurographics Association and John Wiley & Sons Ltd.

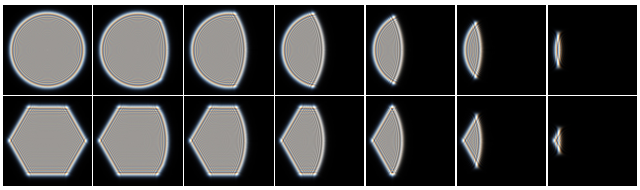


Figure 15: *The dynamic ringing of the aperture cropped by the lens housing, which can be used for realistic lens-flare rendering.*

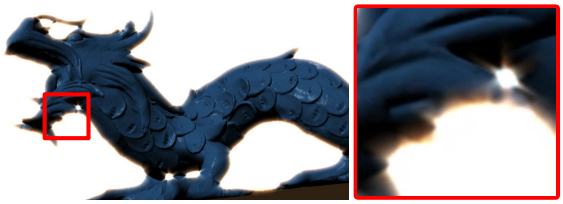


Figure 16: *The examples of the realistic bloom rendering generated with our FFD solution.*

creation (9.42 ms) and spectral integration (13.5 ms) are required, which are shared with the FFT-based solution.

6.2. Ringing at Dynamic Aperture Edges

Our NFD enables the simulation of an aperture pattern cropped by the housing, as is typical for a composite lens. Here, some light rays from the entrance pupil are consumed by the housing. Hence, light reaching the iris aperture is already partially culled (often dubbed as the cat's eye effect). The NFD of such patterns becomes visible in lens-flare ghosts [HESL11]. In contrast to previous work [HESL11, LE13], our solution can handle these dynamic changes. Figure 15 shows examples of dynamic diffraction with cropped apertures.

6.3. Bloom/Glow Rendering

Bloom rendering is another application of our FFD technique, which shows the halos of light sources or their reflections. Real-time rendering typically uses simple postfiltering [JO04], destroying significant details. In our implementation, we create a glare pattern that incorporates dynamic lens effects (e.g., noise) and convolve it with the scene image at a low resolution using pixel intensities as weights. Thereby, we can capture the physical impact of source shapes in the diffraction patterns (Figure 16).

7. Discussion and Limitations

We have shown the utility of our primitive-based FT and NFD in terms of performance, quality, and flexibility in parametrization. While standard pixel-based FT techniques have a fixed cost, our primitive-based FT solution is able to employ many optimization techniques. This results in a higher performance without quality reduction in all proposed application scenarios.

Our experiments showed that reducing the number of quads is crucial for higher performance. At present, we exploited only quads, but integrating more primitives is an interesting direction. For instance, a triangular mesh [HCL91, LLNZ11] can spawn less primitives, but the hierarchical construction can be difficult. Investigating a full vectorial representation is an interesting direction.

In addition to our solution of introducing negative quads when extracting the list from the quadtree, we also described the merging of neighboring quads into rectangles, which reduced the number of primitives for FFD and NFD. Merging across several levels of the quadtree is an interesting direction for future work. While promising, the application of the shift property becomes more difficult.

Our approach is most efficient if larger homogeneous areas appear in the input image. Natural images might lead to a larger quad set, which eventually becomes equivalent to processing every pixel independently, making our approach unpractical. On current hardware, around 1500 quads can be handled very efficiently per frame. This amount is largely sufficient for our application scenarios, especially when using progressive refinement.

Acknowledgments

We would like to thank Dr. Timothy Balint for his assistance in proof-reading this article. Monster Tree and Dragon models are provided through the courtesy of GalaxyArt (turbosquid.com) and the Stanford 3D Scanning Repository, respectively. This work was supported in part by the Global Frontier R&D program (NRF grant no. 2018M3A6A3058649) and the ITRC program (IITP- 2018-2016-0-00312), funded by the Korea Government, and VIDI NextView, funded by NWO Vernieuwingsimpuls. Correspondence on this article can be addressed to Sungkil Lee.

Appendix A: Numerical solution for NFD integral

The NFD computation requires computing the following integral:

$$\int_0^t e^{i(ax+bx^2)}dx = \int_0^t \cos(ax+bx^2)dx + i\int_0^t \sin(ax+bx^2)dx. \tag{20}$$

The Fresnel integrals $C(x)$ and $S(x)$ can express these terms as:

$$\int_0^t \cos(ax+bx^2)dx = \frac{\sqrt{\pi}}{\sqrt{2b}}(\cos\frac{a^2}{4b}C(\phi_t) + \sin\frac{a^2}{4b}S(\phi_t)) \tag{21}$$

$$\int_0^t \sin(ax+bx^2)dx = \frac{\sqrt{\pi}}{\sqrt{2b}}(\cos\frac{a^2}{4b}S(\phi_t) - \sin\frac{a^2}{4b}C(\phi_t)), \tag{22}$$

where $\phi_t = \frac{a+2bt}{\sqrt{2\pi b}}$ and $C(t)$ and $S(t)$ are defined as:

$$C(t) = \int_0^t \cos(\pi x^2/2)dx \quad \text{and} \quad S(t) = \int_0^t \sin(\pi x^2/2)dx. \tag{23}$$

In general, $C(t)$ and $S(t)$ do not have analytic solutions, and thus, we use their numerical approximations [Mie98].

References

[BB17] BELCOUR L., BARLA P.: A practical extension to microfacet theory for the modeling of varying iridescence. *ACM Trans. Graphics 36*, 4 (2017), 65. 2

[BW13] BORN M., WOLF E.: *Principles of optics: electromagnetic theory of propagation, interference and diffraction of light*. Elsevier, 2013. 3

© 2018 The Author(s)
Computer Graphics Forum © 2018 The Eurographics Association and John Wiley & Sons Ltd.

[CHB*12] CUYPERS T., HABER T., BEKAERT P., OH S. B., RASKAR R.: Reflectance model for diffraction. *ACM Trans. Graphics 31*, 5 (2012), 122. 2

[CT65] COOLEY J. W., TUKEY J. W.: An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation 19*, 90 (1965), 297–301. 2

[DG16] DHILLON D. S. J., GHOSH A.: Efficient surface diffraction renderings with Chebyshev approximations. In *SIGGRAPH ASIA Technical Briefs* (2016), ACM, p. 7. 2

[DWMG15] DONG Z., WALTER B., MARSCHNER S., GREENBERG D. P.: Predicting appearance from measured microgeometry of metal surfaces. *ACM Trans. Graphics 35*, 1 (2015), 9. 2

[FJ98] FRIGO M., JOHNSON S. G.: FFTW: An adaptive software architecture for the FFT. In *Proc. Acoustics, Speech and Signal Processing* (1998), vol. 3, pp. 1381–1384. 2, 6

[FL04] FAN G.-X., LIU Q. H.: Fast Fourier transform for discontinuous functions. *IEEE Trans. Ant. and Prop. 52*, 2 (2004), 461–465. 2

[FPV*09] FRANCHETTI F., PUSCHEL M., VORONENKO Y., CHELLAPPA S., MOURA J. M.: Discrete Fourier transform on multicore. *IEEE Signal Processing Magazine 26*, 6 (2009). 2

[GLD*08] GOVINDARAJU N. K., LLOYD B., DOTSENKO Y., SMITH B., MANFERDELLI J.: High performance discrete Fourier transforms on graphics processors. In *Proc. Supercomputing* (2008), p. 2. 2

[HCL91] HOUSHMAND B., CHEW W. C., LEE S.-W.: Fourier transform of a linear distribution with triangular support and its applications in electromagnetics. *IEEE Trans. Ant. and Prop. 39*, 2 (1991), 252–254. 2, 9

[HESL11] HULLIN M., EISEMANN E., SEIDEL H.-P., LEE S.: Physically-based real-time lens flare rendering. *ACM Trans. Graph. 30*, 4 (2011), 108:1–108:9. 1, 2, 4, 5, 6, 8, 9

[HIKP12] HASSANIEH H., INDYK P., KATABI D., PRICE E.: Simple and practical algorithm for sparse Fourier transform. In *Proc. ACM-SIAM Symp. Discrete Algorithms* (2012), pp. 1183–1194. 2

[HP17] HOLZSCHUCH N., PACANOWSKI R.: A two-scale microfacet reflectance model combining reflection and diffraction. *ACM Trans. Graphics 36*, 4 (2017), 66. 2

[HREB11] HOLLANDER M., RITSCHEL T., EISEMANN E., BOUBEKEUR T.: ManyLoDs: Parallel many-view level-of-detail selection for real-time global illumination. *Computer Graphics Forum 30*, 4 (2011), 1233–1240. 5

[JO04] JAMES G., O'RORKE J.: Real-time glow. In *GPU Gems*, Fernando R., (Ed.). Addison Wesley Professional, 2004, pp. 343–362. 9

[Kin75] KINTNER E. C.: Edge-ringing and Fresnel diffraction. *Optica Acta: International Journal of Optics 22*, 3 (1975), 235–241. 5

[KMN*04] KAKIMOTO M., MATSUOKA K., NISHITA T., NAEMURA T., HARASHIMA H.: Glare generation based on wave optics. In *Proc. Pacific Graphics* (2004), pp. 133–140. 1, 2, 6, 8

[LA06] LINDSAY C., AGU E.: Physically-based real-time diffraction using spherical harmonics. *Advances in Vis. Comp.* (2006), 505–517. 2

[LE13] LEE S., EISEMANN E.: Practical real-time lens-flare rendering. *Computer Graphics Forum 32*, 4 (2013), 1–6. 9

[LHP17] LUCAT A., HEGEDUS R., PACANOWSKI R.: Diffraction prediction in HDR measurements. In *Proc. Eurographics Workshop on Material Appearance Modeling* (2017). 2

[LLNZ11] LIU Y.-H., LIU Q., NIE Z.-P., ZHAO Z.-Q.: Discontinuous fast Fourier transform with triangle mesh for two-dimensional discontinuous functions. *J. Electromag. Waves and Appl. 25*, 7 (2011), 1045–1057. 2, 9

[LM83] LEE S.-W., MITTRA R.: Fourier transform of a polygonal shape function and its application in electromagnetics. *IEEE Trans. Ant. and Prop. 31*, 1 (1983), 99–103. 1, 2

[LNL08] LIU Y., NIE Z., LIU Q. H.: DIFFT: A fast and accurate algorithm for Fourier transform integrals of discontinuous functions. *IEEE Microwave and Wireless Components Letters 18*, 11 (2008), 716–718. 2

[MA03] MORELAND K., ANGEL E.: The FFT on a GPU. In *Proc. Graphics Hardware* (2003), Eurographics Association, pp. 112–119. 2

[Mie98] MIELENZ K. D.: Algorithms for Fresnel diffraction at rectangular and circular apertures. *J. Research of the National Institute of Standards and Technology 103*, 5 (1998), 497. 6, 9

[MS91] MCINTURFF K., SIMON P. S.: The Fourier transform of linearly varying functions with polygonal support. *IEEE Trans. Ant. and Prop. 39*, 9 (1991), 1441–1443. 1, 2

[NKON90] NAKAMAE E., KANEDA K., OKAMOTO T., NISHITA T.: A lighting model aiming at drive simulators. *ACM Trans. Graphics 24*, 4 (1990), 395–404. 2

[Oat04] OAT C.: A steerable streak filter. In *Shader X3*, Engel W., (Ed.). Charles River Media, 2004, pp. 341–348. 2

[OZK01] OZAKTAS H. M., ZALEVSKY Z., KUTAY M. A.: *The fractional Fourier transform with applications in optics and signal processing*. Wiley, 2001. 7

[PW15] PEATROSS J., WARE M.: *Physics of Light and Optics*. Bringham Young University, 2015. 1, 3, 5

[RIF*09] RITSCHEL T., IHRKE M., FRISVAD J. R., COPPENS J., MYSZKOWSKI K., SEIDEL H.-P.: Temporal glare: Real-time dynamic simulation of the scattering in the human eye. *Computer Graphics Forum 28*, 2 (2009), 183–192. 1, 2, 5, 8

[Rok93] ROKITA P.: A model for rendering high intensity lights. *Computers & graphics 17*, 4 (1993), 431–437. 2

[Sor95] SORETS E.: Fast Fourier transforms of piecewise constant functions. *Journal of Computational Physics 116*, 2 (1995), 369–379. 2

[SSZG95] SPENCER G., SHIRLEY P., ZIMMERMAN K., GREENBERG D. P.: Physically-based glare effects for digital images. In *Proc. ACM SIGGRAPH* (1995), ACM, pp. 325–334. 1, 2, 8

[Sta99] STAM J.: Diffraction shaders. In *Proc. ACM SIGGRAPH* (1999), ACM, pp. 101–110. 2

[TG17] TOISOUL A., GHOSH A.: Practical acquisition and rendering of diffraction effects in surface reflectance. *ACM Trans. Graphics 36*, 5 (2017), 166. 2

[TLZ10] TAO R., LIANG G., ZHAO X.-H.: An efficient FPGA-based implementation of fractional Fourier transform algorithm. *J. Signal Proc. Systems 60*, 1 (2010), 47–58. 7, 8

[WSS13] WYMAN C., SLOAN P.-P., SHIRLEY P.: Simple analytic approximations to the CIE XYZ color matching functions. *Journal of Computer Graphics Techniques 2*, 2 (2013), 1–11. 4

[WVJH17] WERNER S., VELINOV Z., JAKOB W., HULLIN M. B.: Scratch iridescence: Wave-optical rendering of diffractive surface structure. *ACM Trans. Graphics 36*, 6 (2017), 220:1–11. 2

[YF74] YAP B. K., FANTONE S. D.: Application of a sunburst aperture to diffraction suppression. *JOSA 64*, 7 (1974), 978–982. 1, 2

[YIMS08] YOSHIDA A., IHRKE M., MANTIUK R., SEIDEL H.-P.: Brightness of the glare illusion. In *Proc. Symp. Applied Perception in Graphics and Visualization* (2008), pp. 83–90. 1

© 2018 The Author(s)
Computer Graphics Forum © 2018 The Eurographics Association and John Wiley & Sons Ltd.