Pacific Graphics 2015
N. J. Mitra, J. Stam, and K. Xu
(Guest Editors)

Volume 34 (2015), Number 7

Realtime Rendering Glossy to Glossy Reflections in Screen Space

Chao Xu, Rui Wang†, Hujun Bao

State Key Lab of CAD&CG, Zhejiang University

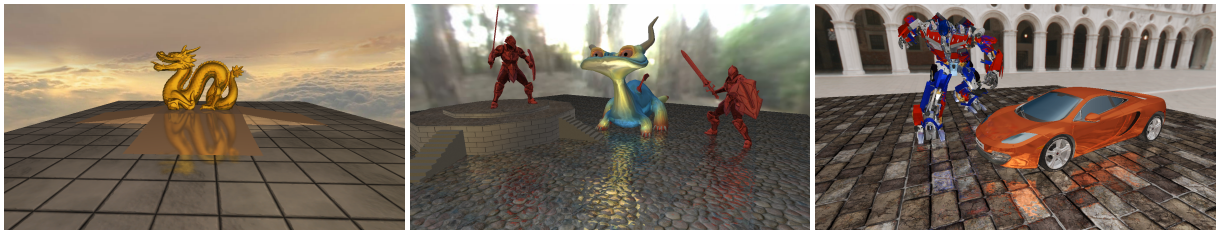


Figure 1: *We proposed a new method to render screen-space glossy to glossy reflections in realtime. Three scenes are all with glossy materials and rendered at* 1280×720 *resolution. The FPS of these three snapshots are 65.1, 64.1 and 61.7 respectively.*

Abstract

Glossy to glossy reflections are lights bounced between glossy surfaces. Such directional light transports are important for humans to perceive glossy materials, but difficult to simulate. This paper proposes a new method for rendering screen-space glossy to glossy reflections in realtime. We use spherical von Mises-Fisher (vMF) distributions to model glossy BRDFs at surfaces, and employ screen space directional occlusion (SSDO) rendering framework to trace indirect light transports bounced in the screen space. As our main contributions, we derive a new parameterization of vMF distribution so as to convert the non-linear fit of multiple vMF distributions into a linear sum in the new space. Then, we present a new linear filtering technique to build MIP-maps on glossy BRDFs, which allows us to create filtered radiance transfer functions at runtime, and efficiently estimate indirect glossy to glossy reflections. We demonstrate our method in a realtime application for rendering scenes with dynamic glossy objects. Compared with screen space directional occlusion, our approach only requires one extra texture and has a negligible overhead, $3\% \sim 6\%$ *loss at frame rate, but enables glossy to glossy reflections.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1. Introduction

Real-time global illumination for complex and dynamic scenes is a remarkable milestone in rendering but has not been accomplished yet. The computation to simulate multi-bounce light transports is still beyond the computational power of desktop PC even with the latest hardware. Many methods have been proposed for different kinds of approximations on light transports to tradeoff for sufficient frame rates. Screen-space global illumination rendering technique is one class of methods with certain approximations. It uses graphic hardware to rasterize scene geometry into screen-space buffers, and only simulates light transports bounced in geometry stored in buffers, instead of computing over all surfaces. Due to the simple implementation and high performance, screen-space methods [MKB*05, SA07, Mit07, BSD08, RS09, HF14] have been successfully applied to sim-

† Corresponding authors: Rui Wang, rwang@cad.zju.edu.cn; Hujun Bao, bao@cad.zju.edu.cn.

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

ulate several global illumination effects in many realtime applications, e.g. games. However, almost all of these methods reuse the screen-space illumination generated by camera to compute next bounces of light transports. This approximation works well if the scene is diffuse or next bounces of light transport are insensitive to directions. But, for glossy to glossy reflections, such an approximation is no longer correct.

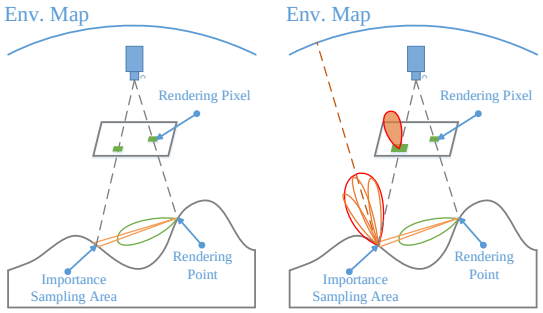
Glossy to glossy reflections are lights bounced between surfaces with glossy materials. With high directional diversity, glossy to glossy light transports are yet difficult to simulate. Several methods with approximations have been proposed to accelerate the estimation of glossy to glossy light transports [LWDB10, LWLD11, XCM*14, YWY08]. The main idea of these approaches is to approximate the glossy transfers, i.e. the distribution of glossy BRDFs to simplify computation. More specifically, it treats the glossy BRDFs at surface points as spherical von Mises-Fisher (vMF) distributions, and merges multiple vMFs of points into one vMF. This new vMF has several advantages. First, it still keeps the directional property even with a broader lobe, which makes it be able to preserve directional reflections. Second, this new distribution depicts the reflections from a surface region, therefore reduces the number of samples. However, the merging of multiple vMFs is an iterative optimization process [HSRG07], which has to be computed in the precomputation stage.

Thus, these methods are impractical for realtime applications with dynamic scenes.

In this paper, we present a new screen-space method for rendering glossy to glossy reflections with dynamic scenes at real-time frame rates. Our key idea is to compute and store glossy transfers in screen space instead of only the color buffer used in previous screen-space methods. In Fig. 2, we illustrate the difference between our method and SSDO. The basic strategy is to enable linear operators on these glossy transfers, such as linear filtering and MIP-mapping, so that we can integrate the rendering of glossy to glossy reflections into screen-space rendering framework. To achieve it, First, we derive a new parameterization of vMF distribution and simplify the non-linear fit of multiple vMF distributions into a linear sum in the new space. Then, we present a new linear filtering technique to build MIP-maps on glossy BRDFs in screen space. In this way, we are able to approximate the glossy transfer from different sizes of patches by reconstructing the vMF distribution from different levels of MIP-maps. This allows us to efficiently estimate indirect glossy to glossy reflections under local or environmental lighting.

We demonstrate our method in a realtime application for rendering scenes with glossy objects. Compared with SSDO, our approach only requires one extra texture and has a negligible overhead, 3% $\sim$ 6% loss at framerate in our examples, but enables new glossy to glossy reflections. We conclude our main contributions as:

- A new realtime rendering technique to simulate glossy to glossy reflections in screen space.
- A new parameterization of vMF distribution to linearly sum multiple vMFs instead of non-linear fit.
- A new linear filtering technique to decouple outgoing directions from the filtering of glossy BRDFs so that MIP-maps on vMFs of glossy transfers can be built in screen space and used to create filtered radiance transfer function at runtime.



(a) SSDO (b) Our method

Figure 2: *Illustration of SSDO and our method. Compared with SSDO, we store glossy transfers in the screen space so as to correctly compute the glossy to glossy light transport bounced in the scene.*

2. Related Work

Even with decades of development, global illumination is still a big challenge in graphics research. Certain classes of light paths have traditionally been a source of difficulty in fast and efficient simulations. An example is the light paths bounced between many glossy surfaces, known as glossy to glossy reflections.

Rendering Glossy to Glossy Reflections. Monte Carlo path tracing [CPC84] and some followups, the bidirectional path tracing [LW93, Vea98] and the Metropolis Light Transport [VG97], are all powerful tools to physically simulate various light transports, including the glossy to glossy reflections. However, due to the highly directional varies and the high dimensions of glossy to glossy reflections, rays usually diverge upon each reflection, even with proper importance sampling. Therefore, a large number of sample paths is required to eliminate noise in the image. Yu et al. [YWY08] presented an interactive GPU-based algorithm for accurately rendering glossy reflection effects using GPU-based BRDF importance sampling, Jakob and Marschner [JM12] explored the low dimensional embedding of specular paths in the path space, and proposed a Markov Chain Monte Carlo technique to efficiently trace paths in scenes with difficult specular or highly glossy light transports. However, due

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.

to the high computation costs, physically path tracing methods are still far from realtime rendering applications.

To fast estimate the glossy to glossy reflections, Laurijssen et al. [LWDB10] proposed a method that treats the glossy BRDFs at surface points as vMF distributions, and merges multiple vMFs into one vMF. Therefore, instead of tracing more rays, the glossy to glossy reflections can be approximately computed by the merged vMF with local incident lighting. The underlying idea of fitting or merging vMFs into one vMF was first proposed by Han et al. [HSRG07] to filter normal maps. An EM algorithm is employed to take the approximation. Compared with their methods, we find a new parameterization of vMF distribution and simplify the non-linear iterative optimization into linear filtering. This greatly reduces the computation time of merging multiple vMFs while using hardware MIP-mapping, thereby can be applied in the realtime rendering.

The idea of gathering of multi-bounce glossy reflections proposed in [LWDB10] was then extended in the Precomputed Radiance Transfer (PRT) framework to enable realtime relighting of scenes [LWLD11]. In the PRT framework, several methods have been proposed to consider interactive rendering or editing of glossy BRDFs with global illumination [BAOR06, SZC*07, BAEDR08, NKLN10]. However, the main drawback of these PRT methods is the restriction on static scenes. By contrast, our goal is to handle dynamic scenes.

Recently, Xu et al. [XCM*14] presented an analytical derivation on the one-bounce interreflection equation from a triangle to a shading point, allowing efficient computation of one-bounce interreflection, including glossy to glossy reflections. However, even with a GPU-based implementation, it is still impractical for realtime rendering of dynamic scenes.

Screen Space-based Rendering. To enable realtime rendering, screen space-based rendering attracts much attention due to the simple implementation and high performance. Usually, it first uses graphic hardware to rasterize scene geometry into screen-space buffers, and then simulates light transports bounced using geometry stored in buffers as a post-process. Instead of computing over all surfaces, screen space-based rendering is output-sensitive and scalable for complex scenes.

Several global illumination effects have been successfully simulated in screen space. Mertens et al. [MKB*05] proposed a screen-space subsurface scattering method (SSSS). Ambient Occlusion (AO) is a coarse approximation to general light transport. It can be efficiently approximated by nearby geometry in neighboring pixels. This leads to screen-space ambient occlusion (SSAO) method with many variations [SA07, Mit07, BSD08]. A standard SSAO algorithm illuminates a pixel by first computing an average visibility value from a set of neighboring pixels, and then multiplying this occlusion value with the unoccluded illumination from all incoming directions. Ritschel et al. [RGS09] extended SSAO to screen-space directional occlusions (SSDO) that it does not compute the illumination from all incoming directions, but only from the visible directions. Therefore, it enables more types of effects than just occlusion, such as shadows and indirect color bleeding.

To take glossy reflections into consideration, Robison and Shirley [RS09] developed an image space gathering method, and Hermanns et al. [HF14] presented a screen-space cone tracing method. However, these methods only handle diffuse to glossy reflections and can not simulate the glossy to glossy reflections.

Instead of directly using screen buffer as color buffer to compute multiple bounces of reflections, Wang et al. [WWZ*09] proposed a parallel screen-space irradiance caching scheme to enable interactive global illumination rendering. McGuire et al. [ML09] presented a screen-space photon mapping method that recasts the initial and final photon bounces as image-space operations amenable to GPU acceleration.

Screen-space method was also used in the simulation of subsurface scattering. Jimenez et al. [JSG09] presented a screen-space diffusion approximation to simulate skin in real-time performance.

In summary, the goal of our method is to simulate glossy to glossy reflections in screen space. As we know, this problem has not been addressed before.

3. Approximating Glossy to Glossy Light Transport

In this section we first describe the rendering equation to compute glossy to glossy reflections, and then introduce the approximation we made to compute them in screen space. An illustration of how we approximate glossy to glossy light transport is shown in Fig. 3

3.1. Glossy to Glossy Light Transport

According to the rendering equation [Kaj86], the outgoing radiance L_o at a shading point $\mathbf{p}$ along direction $\mathbf{s}_o$ is computed by the following integral:

$$L_o(\mathbf{p},\mathbf{s}_o) = \int_H \rho(\mathbf{p},\mathbf{s}_o,\mathbf{s}) L_i(\mathbf{p},\mathbf{s}) \cos(\mathbf{n}_p,\mathbf{s}) \mathrm{d}\mathbf{s} \qquad (1)$$

where L_i is the incident radiance, $\mathbf{s}$ is an incident direction, $\rho(\mathbf{p},\mathbf{s}_o,\mathbf{s})$ is the BRDF at surface point $\mathbf{p}$, $\cos(\mathbf{n}_p,\mathbf{s})$ is the clamped cosine between surface normal $\mathbf{n}_p$ and direction $\mathbf{s}$, and H is the hemisphere centered around the surface normal $\mathbf{n}_p$.

We use Monte Carlo path tracing to sample the incident light at point $\mathbf{p}$ as:

$$L_o(\mathbf{p},\mathbf{s}_o) = \frac{1}{N}\sum_{k=1}^{N} \frac{\rho(\mathbf{p},\mathbf{s}_o,\mathbf{s}_k) L_i(\mathbf{p},\mathbf{s}_k) \cos(\mathbf{n}_p,\mathbf{s})}{f(\mathbf{s}_k,\mathbf{s}_o)} \qquad (2)$$

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.

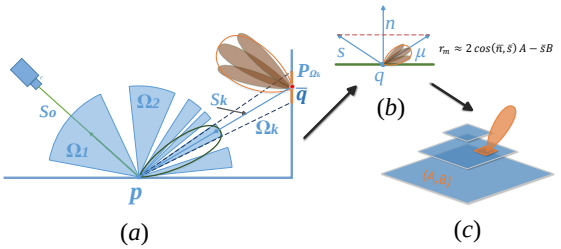


Figure 3: *Illustration of approximating glossy to glossy light transport. (a) We use cone tracing to trace paths from surface point* $\mathbf{p}$*. The cone hit region is* S_{Ω_k}*. The vMF lobes at region* S_{Ω_k} *is approximated by one vMF lobe, shown in orange. (b) The new lobe is computed by merging multiple reflected lobes. The center direction of the new lobe is the reflected direction of outgoing direction* $\mathbf{s}$*. (c) We decouple the outgoing direction from the computation of merging vMF lobes and MIP-map A and B terms that are independent of outgoing directions.*

where N is the number of samples, and $f(\mathbf{s}_k,\mathbf{s}_o)$ is the PDF to generate the sample direction $\mathbf{s}_k$.

Then, we adapt the cone tracing proposed in [CK07, CNS*11] to trace paths in screen space. Specifically, the inverse of the product between the PDF and the total number of samples, $\frac{1}{Nf(\mathbf{s}_k,\mathbf{s}_o)}$, is approximated by a cone of solid angle Ω_k, and the incidence radiance over solid angle Ω_k is approximated by the average incident radiance $\overline{L}_{\Omega_k}(\mathbf{p},\mathbf{s}_k)$, i.e.:

$$\Omega_k = \frac{1}{Nf(\mathbf{s}_k,\mathbf{s}_o)},\quad \overline{L}_{\Omega_k}(\mathbf{p},\mathbf{s}_k) = \int_{\Omega_k} L_i(\mathbf{p},\mathbf{s})\mathrm{d}\mathbf{s} \tag{3}$$

Thus, the rendering equation is reformulated as:

$$L_o(\mathbf{p},\mathbf{s}_o) \approx \sum_{k=1}^{N} \rho(\mathbf{p},\mathbf{s}_k,\mathbf{s}_o)\overline{L}_{\Omega_k}(\mathbf{p},\mathbf{s}_k)\cos(\mathbf{n}_p,\mathbf{s}) \tag{4}$$

Note that the incident radiance $L_i(\mathbf{p},\mathbf{s})$ is the outgoing radiance from surface point $\mathbf{q}_s$ along direction $\mathbf{s}$, i.e. $L_i(\mathbf{p},\mathbf{s}) = L_o(\mathbf{q}_s,\mathbf{s})$, Fig. 3(a). The outgoing radiance from surface point $\mathbf{q}_s$ in the region S_{Ω_k} can be computed by one more bounce of incident radiance. Thus, the average incident radiance $\overline{L}_{\Omega_i}(\mathbf{p},\mathbf{s}_k)$ under the second bounce of light transport can be computed as:

$$\overline{L}_{\Omega_k}(\mathbf{p},\mathbf{s}_k) = \int_{\Omega_k}\int_{H} \rho(\mathbf{q}_s,\mathbf{s}',\mathbf{s})L_i(\mathbf{q}_s,\mathbf{s}')\cos(\mathbf{n}_s,\mathbf{s}')\mathrm{d}\mathbf{s}'\mathrm{d}\mathbf{s} \tag{5}$$

Then, we assume that the incident radiance to $\mathbf{q}_s$ is from distant lights, thus it does not change much for points in the region S_{Ω_k}. Therefore, we use the cosine-weighed incident radiance towards a mean point in the region, $L_i(\overline{\mathbf{q}},\mathbf{s}')\cos(\mathbf{n}_{\overline{q}},\mathbf{s}')$, to replace the incident radiance to other points. Under this assumption, Eq. 5 can now be reordered as follows:

$$\begin{aligned}\overline{L}_{\Omega_k}(\mathbf{p},\mathbf{s}_k) &\approx \int_S L_i(\overline{\mathbf{q}},\mathbf{s}')\cos(\mathbf{n}_{\overline{q}},\mathbf{s}')\int_{\Omega_k}\rho(\mathbf{q}_s,\mathbf{s}',\mathbf{s})\mathrm{d}\mathbf{s}\mathrm{d}\mathbf{s}' &(6)\\ &= \int_S L_i(\overline{\mathbf{q}},\mathbf{s}')\cos(\mathbf{n}_{\overline{q}},\mathbf{s}')\overline{\rho}(\overline{\mathbf{q}},\mathbf{s}',\mathbf{s}_k)\mathrm{d}\mathbf{s}' &(7)\end{aligned}$$

where $\overline{\rho}(\overline{\mathbf{q}},\mathbf{s}',\mathbf{s}_k)$ is the average glossy BRDFs over region S_{Ω_k}. It can be further approximated by merging glossy BRDFs from a set of sample points in the region into one glossy BRDF as:

$$\overline{\rho}(\overline{\mathbf{q}},\mathbf{s}',\mathbf{s}_k) = \int_{\Omega_k}\rho(\mathbf{q}_s,\mathbf{s}',\mathbf{s})\mathrm{d}\mathbf{s} \approx \sum_{j=1}^{M}\rho(\mathbf{q}_j,\mathbf{s}',\mathbf{s}_k). \tag{8}$$

The average glossy BRDF, $\overline{\rho}(\overline{\mathbf{q}},\mathbf{s}',\mathbf{s}_i)$, can be regarded as the glossy transfer from the region S_{Ω_k} to point $\mathbf{p}$. It helps us to fast estimate the glossy reflection from one region to one shading point, so as to accelerate the rendering. Previous methods [HSRG07,LWDB10] proposed an iterative EM algorithm to optimize Eq. 8, which is time-consuming and impractical for realtime rendering. We need to develop our own optimization approach.

3.2. Approximating Glossy BRDFs

Inspired by previous work [HSRG07, LWDB10], we use von Mises-Fisher (vMF) distributions to represent specular BRDFs. In the following, we first propose a new parameterization space to linearly merge multiple vMFs, and then extend it to compute the average glossy BRDFs.

3.2.1. Summing vMF Distributions

The von Mises-Fisher Distribution. vMF distributions have been used in many methods to approximate specular BRDFs. Mathematically, the vMF distribution, $\gamma(\mathbf{s})$, is a probability density function that describes the distribution of directions $\mathbf{s}$ centered around a mean direction μ as:

$$\gamma(\mathbf{s}) = c(\kappa)e^{\kappa(\mu\cdot\mathbf{s})} \tag{9}$$

where κ is the inverse angular width: higher values of κ correspond to more concentrated distributions, $c(\kappa) = \frac{\kappa}{4\pi\sinh(\kappa)}$ is a normalization factor. Assuming $\kappa \gg 1$, the distribution can be approximated as:

$$\gamma(\mathbf{s}) \approx \frac{\kappa}{2\pi}e^{\kappa(\mu\cdot\mathbf{s})-1} \tag{10}$$

Let us take another view on the vMF distribution. One vMF distribution can be regarded as an approximation on a set of directions resulting from a stochastic process [BDGS05]. Given a set of directions, $\mathbf{s}_i, i\in[1,M]$, one vMF distribution, $\gamma(\mathbf{s})$, that best fits the distributions of these directions can be computed as:

$$\mu = \frac{\mathbf{r}}{\|\mathbf{r}\|}, \kappa \approx \frac{3\|\mathbf{r}\| - \|\mathbf{r}\|^3}{1-\|\mathbf{r}\|^2} \tag{11}$$

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.

where $\mathbf{r} = \frac{1}{M}\sum_{i=1}^{M}\mathbf{s}_i$. $\mathbf{r}$ is the unnormalized average direction. The magnitude $\| \mathbf{r} \|$ indicates the angular width of the fitted vMF distribution. The value is between 0 and 1. Imagine that a uniform distribution of directions on the sphere results in $\mathbf{r} = 0$, while an extremely concentrated distribution results in $\mathbf{r} \approx 1$.

Summing vMF Distributions. According to Eq. 11, one vMF can be represented by one unnormalized average direction $\mathbf{r}$. Given one vMF with two parameters (μ, κ), using Eq 11, we can compute the magnitude $\| \mathbf{r} \|$ as

$$\| \mathbf{r} \|^3 - \kappa \| \mathbf{r} \|^2 - 3 \| \mathbf{r} \| + \kappa = 0 \tag{12}$$

This cubic equation has three real roots. But while $\kappa > 1$, only one root is between 0 and 1, which is a valid value for $\| \mathbf{r} \|$. The proof is provided in Appendix A. Using $\| \mathbf{r} \|$, we can compute the scaled unnormalized direction as

$$\mathbf{r} = \| \mathbf{r} \| \mu \tag{13}$$

Eq. 12 and 13 indicate that we can reparameterize one vMF by the unnormalized direction $\mathbf{r}$. This gives us a new parameterization space to manipulate vMFs. Given a set of vMFs with parameters $\{(\mu_j, \kappa_j)\}_M$, we can regard each vMF as an approximations of directions generated by stochastic processes. Therefore, the merging of these vMFs is the merging of all of these directions. We have:

$$\mathbf{r}_m = \frac{1}{M}\sum_{j=1}^{M}\sum_{i=1}^{N_j}\mathbf{s}_{ji} = \frac{1}{M}\sum_{j=1}^{M}\mathbf{r}_j \tag{14}$$

where $\mathbf{r}_m$ is the unnormalized direction for the merged new vMF, and $\mathbf{r}_j$ is the unnormalized direction computed from (μ_j, κ_j). Eq. 14 provides us with a simple way in the the unnormalized direction space to merge vMFs by just linearly summing them.

3.2.2. Summing Glossy BRDFs

Once we have the way to sum vMFs, we use it to sum glossy BRDFs to obtain summed glossy transfer. For one point $\mathbf{q}$ in the region S_{Ω_k}, assuming that the outgoing direction of this sample point is $\mathbf{s}$, and the normal is $\mathbf{n}$, the reflected direction μ of the vMF lobe is computed as, Fig. 3(b):

$$\mu = 2\mathbf{n}\cos(\mathbf{n}, s) - s \tag{15}$$

where μ is the normalized reflected direction at point $\mathbf{q}$ and it is also the lobe direction of reflected vMF distribution of glossy BRDF at point $\mathbf{q}$. Using Eq. 13, we have

$$\mathbf{r} = \| \mathbf{r} \| (2\mathbf{n}\cos(\mathbf{n}, s) - s) \tag{16}$$

So for all surface points in the region S_{Ω_k}, using Eq. 14 and Eq. 16, we can compute the merged vMF of glossy BRDFs on the region as:

$$\begin{aligned}\mathbf{r}_m &= \frac{1}{M}\sum_j \mathbf{r}_j = \frac{1}{M}\sum_{j=1}^{M} \| \mathbf{r}_j \| (2\mathbf{n}_j\cos(\mathbf{n}_j, \mathbf{s}_j) - \mathbf{s}_j) \\ &= \frac{2}{M}\sum_{j=1}^{M} \| \mathbf{r}_j \| \mathbf{n}_j\cos(\mathbf{n}_j, \mathbf{s}_j) - \frac{1}{M}\sum_{j=1}^{M} \| \mathbf{r}_j \| \mathbf{s}_j \end{aligned} \tag{17}$$

Then, we assume that the outgoing directions $\mathbf{s}_j$ do not change too much for this region, so that we are able to use an average outgoing direction $\bar{\mathbf{s}}$ to replace all $\mathbf{s}_j$; and we use the cosine $\cos(\bar{\mathbf{n}}, \bar{\mathbf{s}})$ between the average normal $\bar{\mathbf{n}}$ and the average outgoing direction $\bar{\mathbf{s}}$ to replace $\cos(\mathbf{n}_j, \mathbf{s}_j)$. Eq. 17 can be approximated as follows:

$$\begin{aligned}\mathbf{r}_m &\approx \frac{2\cos(\bar{\mathbf{n}}, \bar{\mathbf{s}})}{M}\sum_{j=1}^{M} \| \mathbf{r}_j \| \mathbf{n}_j - \frac{\bar{\mathbf{s}}}{M}\sum_{j=1}^{M} \| \mathbf{r}_j \| \\ &= 2\cos(\bar{\mathbf{n}}, \bar{\mathbf{s}})A - \bar{\mathbf{s}}B \end{aligned} \tag{18}$$

where

$$A = \frac{1}{M}\sum_{j=1}^{M} \| \mathbf{r}_j \| \mathbf{n}_j, \quad B = \frac{1}{M}\sum_{j=1}^{M} \| \mathbf{r}_j \| \tag{19}$$

The approximation in Eq. 18 has several advantages. First, A and B are two terms only depending on normals and $||\mathbf{r}_j||$ on surfaces, but being independent to the outgoing directions. It decouples the outgoing directions from summing of BRDF lobes. Therefore, it lets us compute A and B for surface regions before we actually trace light transports. Second, A and B are linearly summed from surface points. This makes us be able to use graphic hardware to build MIP-maps on A and B in screen space to fast compute the merged lobe parameter $\| r_m \|$ for different sizes of scene regions. While we get the $\| r_m \|$, we can use Eq. 11 to compute the analytical form of the merged vMF lobe and obtain the average glossy transfer $\bar{\rho}(\bar{\mathbf{q}}, \mathbf{s}', \mathbf{s}_i)$. At this point, we successfully apply linear operators on glossy transfers and integrate the rendering of glossy to glossy reflections in screen space.

4. Screen Space Algorithm and Implementation

In this section we describe the algorithm implementing the derivation in above section to achieve a realtime rendering of glossy to glossy reflections in screen space.

Overview. As many screen-space rendering approaches, our algorithm takes two passes. First, it renders the entire scene to generate G-Buffer that stores geometry and material data on pixels. Second, light transports are traced per pixel in the screen space. While in tracing paths, for each intersection of path, we reconstruct the lobe of glossy transfer using our previous derivations, and integrate it with lighting to compute the glossy to glossy reflections.

4.1. Generating G-buffers

At the first pass, we convert the 3D scene into a 2D representation in screen space. The pseudo code of the pixel shader used in this pass is given in Algorithm 1. First, positions, normals, and materials of scenes are rendered into the geometry buffer (G-buffer) as textures, line 2-5 of Algorithm 1. According to Eq. 18, we need one extra RGBA

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.

Algorithm 1 Generating G-Buffer

```
1:  procedure PIXELSHADER
2:      N ← Normal
3:      Pos ← mul(objPos, World)
4:      D ← DiffuseColor
5:      S ← SpecularColor
6:      κ ← Kappa
7:      //convert the κ to || r ||
8:      || r ||= Convert(κ)
9:      //RGB channels store A =|| r || n
10:     R.rgb ←|| r || *N
11:     //A channel stores B =|| r ||
12:     R.a ←|| r ||
13: end procedure
```

texture in G-buffer, where RGB channels store $\| \mathbf{r}_j \| n_j$ and the A channel stores $\| \mathbf{r}_j \|$, line 8-9. To convert the κ to $\| \mathbf{r}_j \|$, we solve the cubic equation, Eq. 12, in a preprocess, and build a table for range $[5, 4000]$. For glossy BRDFs with larger κ, we treat them as specular BRDFs, since the distributions of such vMF lobes are very concentrate. At runtime, we use the table to obtain the value of $\| \mathbf{r}_j \|$ for every pixel, line 10.

After the first pass, we build MIP-maps for the texture stored in G-buffer so that it allows us to efficiently compute the merged glossy BRDF at different regions in the scene. Specifically, for a certain region size, the merged glossy BRDF of this region is just computed by fetching a certain level in the MIP-maps.

4.2. Tracing Glossy to Glossy Reflections

The pseudo code of the pixel shader used in the second pass to trace glossy to glossy reflections is shown in Algorithm 2. For each surface point of pixel, we use importance sampling, Eq. 4, to generate a number of paths. For each path, we use the geometry stored in the G-buffer to calculate the intersection region by ray marching as that in SSDO [RGS09]. With ray marching, we iteratively move path forward along the ray direction until we get within a threshold distance of the surface, line 3 of Algorithm 2. If there does not exist any intersections, we directly compute environmental lighting from the direction using BRDF parameters at this surface point (μ_p, κ_p), line 4-6. If there exists intersection along this path, we first use the cone size and the marched distance to compute a region size. Then, we project this region back to camera to compute the MIP-map level in screen space, line 8. Using the MIP-map level, we fetch filtered values A and B from R_{buf} at the screen position of ray intersection to compute the unnormalized direction $\mathbf{r}$ using Eq. 18, line 9-11. As long as we get $\mathbf{r}$, we then use Eq. 11 to recover the vMF parameters (μ, κ) to reconstruct the vMF lobe, line 12-14. If the lighting is from local lights, the outgoing radiance is directly computed by combing directional contributions from

Algorithm 2 Tracing glossy to glossy reflections

```
1:  procedure PIXELSHADER
2:      for all paths do
3:          hitPoint ←RAYMARCH(p, path)
4:          if nohit then
5:              Color ← Color+ENVMAP(μ_p, κ_p)
6:              continue
7:          end if
8:          mLevel ←LEVEL(RayLen, EyeDis, Weight)
9:          n̄ ← N_Buf(hitPointTexCrd, mLevel)
10:         r̄n ← R_Buf(hitPointTexCrd, mLevel)
11:         r ← 2 * r̄n.rgb * max(0, dot(n̄, p)) − p * r̄n.a
12:         || r ||← length(r)
13:         μ ← r/ || r ||
14:         κ ← (3* || r || − || r ||^3)/(1− || r ||^2)
15:         //Compute local lighting
16:         Color ← Color+BRDF(μ, κ, LPos)
17:         //Compute environmental lighting
18:         Color ← Color+ENVMAP(μ, κ)
19:     end for
20: end procedure
```

local lights and vMFs, line 15. If the light sources are local lights, we directly use filtered BRDF to compute local shading, line 16. If the lighting is from environment map, we use prefiltered environment map [KVHS00] to compute the environment lighting. Here, we ignore the visibility of the second bounce to simplify the computation, line 18.

5. Results

We implement our algorithm in Microsoft DirectX 11 on a PC with Intel i7 3770 CPU and NVIDIA GeForce GTX 980 graphics card. Since we compute everything from scratch in each frame, the user can dynamically manipulate any element in the scene, including lighting, viewpoint, material, and geometry.

Example scenes. We test our algorithm on three example scenes, the Dragon, the Warrior and the Transformer, Fig. 1. Most results are lit under environmental lighting, but in Fig. 6, we also show an example of using a local point to illuminate the Dragon scene. More results can be found in the supplementary video, including animations of these scenes, and user's interactive browsing in these scenes. The video is captured on the fly. In the Dragon scene, a dragon is set on a platform with four curved slopes. The total number of vertices is 42.2 thousand. The glossiness coefficients, κ, of the dragon, the platform and the plane are 64, 192 and 256, respectively. In the Warrior scene, two animated warriors are fighting with a dragon. This scene has 24.5 thousand vertices. The κ, are 192, 64 and 384 for the dragon, the warriors, and the ground, respectively. We tessellate the ground into bumpy surfaces using tessellation shader to demonstrate the capability of our method handling reflections bounced on

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.

complex geometry surfaces. The Transformer scene has one transformer and one car, which is the most complex scene in this paper with 152.6 thousand vertices. The κ of the transformer, the car and the ground are 64, 128 and 384. The ground is also tessellated to generate more geometry details.

Render Quality. Since our method makes several approximations to enable the realtime rendering, as well as makes improvements to add glossy to glossy reflections, we compare our method with other two approaches, shown in Fig. 4. Two zoomed results for all approaches are highlighted for comparisons.

We first compare results of our method against a standard screen-space path tracer, where 256 paths per pixel are traced in the screen space using importance sampling. The traced results are used as reference images. Compared with these reference images, it can be seen that our method mainly captures the position and shape of highlights produced by glossy to glossy reflections, for examples, the highlights from dragon to the slope in the Dragon scene, and the interreflections between the car and the ground in the Transformer scene. However, due to the small number of sample rays, our method also produces some errors at capturing reflections from areas of complex geometry. The obvious example is the highlights produced by dragon head in the Dragon scene. Since the depth and geometry vary much at the head region, our method does not fully produce all glossy to glossy reflections. The second method we compared with is the SSDO [RGS09], where we extend the restriction on diffuse materials of SSDO to handle a glossy receiver. Comparing our results with those generated by SSDO, it is obvious that our method better preserves many glossy to glossy reflections that can not be captured by SSDO. For almost every zoomed region, there are highlights missed by SSDO, for examples, the dragon body region in the Dragon scene, the neck of dragon in the Warrior scene, and the wrest of the Prime model in the Transformer scene. These indirect highlights are mainly generated by light-glossy-glossy-eye paths. The contributions through glossy-glossy paths are highly directional, therefore can not be well approximated by a color buffer used in SSDO. Results verify our motivation that we need to store glossy transfers instead of colors to capture glossy to glossy reflections.

In Fig. 5, we show three results using different number of samples to trace secondary light transports. As can be seen, in the Transformer scene, 8 secondary samples are sufficient to produce good results. 4 samples are also able to produce plausible results. Therefore, we use 4 and 8 samples to generate all results.

Performance. In Table 1, we summarize the performance of applying our algorithm and SSDO on these 3 example scenes with different resolutions. Since our method works completely in image space, we can render scenes with different complexity of geometry at similar frame rates. One important factor impacting on the performance is the number of paths sampled for the secondary light transports. Once the sample number doubles, the FPS approximately drops half. Another important factor is the number of pixels. From 1280×720 to 1920×1080, the pixel number increases 2.25 times, and the performance drops approximately 2 times. Since we use tessellation in some scenes, we also give out the FPS while using tessellation shaders, shown in brackets in Table 1. Comparing the performance of our method and the SSDO, our method only has a negligible overhead, $3\% \sim 6\%$ loss at frame rate, but has a significant improvement at capturing the glossy bounces between objects.

Resolution	Scene	Samples	Time costs		
			SSDO	Ours	%.
			FPS	FPS	
1280×720	Dragon	N=4	132.3	124.4	5.9
		N=8	69.7	65.1	6.6
	Warrior	N=4	126.8	120.3(76.5)	5.1
		N=8	66.2	64.1(47.3)	3.2
	Transformer	N=4	123.9	115.9(72.4)	6.4
		N=8	65.0	61.7(45.2)	5.1
1920×1080	Dragon	N=4	65.0	61.3	5.7
		N=8	36.5	34.5	5.4
	Warrior	N=4	60.6	58.1	4.2
		N=8	34.5	32.8	4.7
	Transformer	N=4	54.2	52.4	3.4
		N=8	30.0	28.5	3.7

Table 1: *Statistics of example scenes. From left to right: resolution, scene, the number of sample rays per pixel and the time costs of SSDO and our method. The FPS is averaged for a period of time. Note that some of our example scenes are rendered with tessellation shaders. Their FPSs are shown in brackets. As can be seen, our method is only slightly slower than SSDO, about* 3% *to* 6% *FPS drop.*

6. Discussion and Limitation

Our method employ screen-space rendering framework, therefore inherits its advantages and disadvantages. The basic assumption of a screen space approach is that the local geometry is sufficiently represented by 3D spatial proximity of the nearby pixels. However, this assumption may fail in some cases. For those invisible objects or objects out of the screen, our method can not capture the glossy reflections from them.

Several approximations are made in our derivation of fitting multiple vMFs. To better understand the benefits and loss of these approximations, we show some comparisons in Fig. 6. The test scene is the Dragon scene, but lit by a local light. Fig. 6(a) is a reference image generated by screen-space path tracing. In Fig. 6(b), we use the iterative EM algorithm [HSRG07] to optimize the merging of multiple vMFs. Fig. 6(c) shows our result with 8 samples. In Fig. 6(d), we do not use MIP-mapped vMFs but directly use the vMF at hit point of secondary path to render the image. From this comparison, it can be seen that the vMF reconstructed from

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.

(a) Reference (b) Our method (c) SSDO

Figure 4: *Results comparison.*

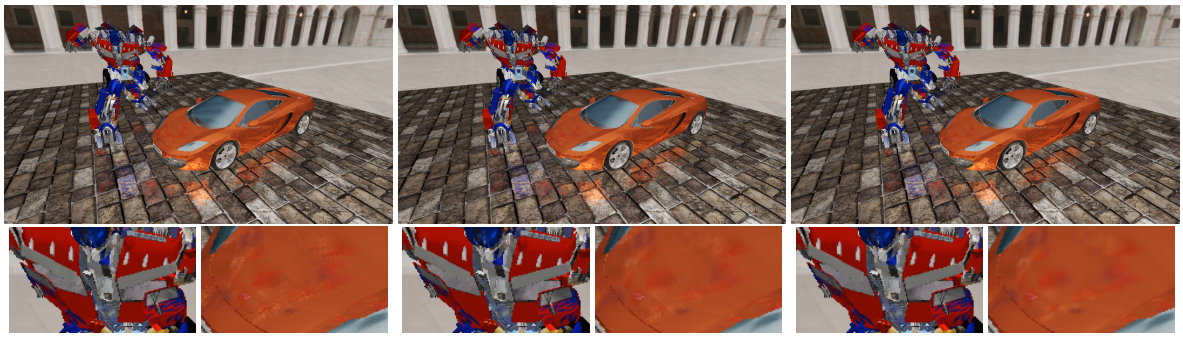
our MIP-maps well captures the mainly indirect glossy to glossy reflections, e.g. on the dragon head. Comparing with the result without using MIP-maps, our approximation on multiple vMFs effectively improve the lighting effects between glossy surfaces. Compared with the fitted vMF computed from the EM algorithm, our approximation loses some accuracy. There are two main sources to produce the quality loss. One is the approximation to decouple the outgoing direction from the bounced lobes of vMFs. The other is that due to the limitation of MIP-map, we only fit multiple vMFs into one vMF, but in [HSRG07], maximum 6 lobes are used to cluster and fit multiple vMFs, which generates a better approximation to the reference. Thus, our approximation may not be able to capture very complex reflections that requires more than one vMF to fit the glossy transfers. We regard it as an important future direction for our work.

Currently, we only consider two bounces of light transports, the light-glossy-glossy-eye paths. Our method can be directly extended to more bounces by simply tracing extra bounces after two bounces. However, with increase of the depth of paths, the scalability of our method is poor. It will be an interesting and important topic to enable realtime long path tracing.

7. Conclusions

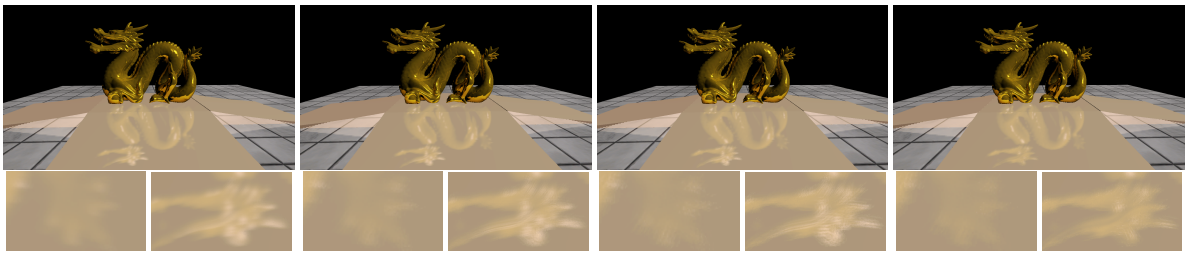
We present a new realtime rendering technique to render screen-space glossy to glossy reflections. Our basic idea is to compute and store glossy transfers in screen space instead of only the color buffer used in previous screen-space methods. To achieve it, we first derive a new parameterization of vMF distribution to linearly sum multiple vMFs, which is hardware friendly and computed in realtime. Using this

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.



(a) N=4 (b) N=8 (c) N=16

Figure 5: *Results generated by different number of secondary sample paths.*



(a) Reference (b) EM algorithm (c) MIP-mapping (d) Without MIP-mapping

Figure 6: *Results using different optimization to merge multiple vMFs.*

new parameterization, we then present a new linear filtering technique to decouple reflected directions from the filtering of reflected glossy BRDFs so that we are able to build MIP-maps on vMFs of glossy BRDFs in the screen space and create filtered radiance response functions at runtime. We demonstrate our method in a realtime application for rendering scenes with glossy objects. Compared with SSDO, our approach only requires one extra texture and has a negligible overhead, 3% $\sim$ 6% loss at framerate in our examples, but enables glossy to glossy reflections. We believe that the consideration of multiple bounces of light transport in screen-space rendering framework is an interesting avenue of further research bringing together physically plausible complex lighting effects of dynamic scenes and real-time frame rates.

Acknowledgements

This work was partially supported by NSFC (No. 61472350), the 863 program of China (No. 2012AA011902), the Specialized Research Fund for the Doctoral Program of Higher Education of China (No. 20110101130011) and the Fundamental Research Funds for the Central Universities (2015XZZX005-05).

Appendix A: The roots of the cubic equation

In the Appendix, we prove that the cubic equation, Eq. 12, has one and only one real root between 0 and 1.

Note that $\kappa > 1$. The discriminant of this cubic equation is:

$$\Delta = 4(27 + 9\kappa^2 + \kappa^4) > 0 \tag{20}$$

Thus, the equation has three distinct real roots.

To find these roots, we first take the derivative of this cubic equation as:

$$3 \parallel \mathbf{r} \parallel^2 -2\kappa \parallel \mathbf{r} \parallel -3 = 0 \tag{21}$$

where it has two real roots:

$$\parallel \mathbf{r} \parallel = \frac{\kappa \pm \sqrt{\kappa^2 + 9}}{3} \tag{22}$$

It can be seen that $\parallel \mathbf{r}_1 \parallel = \frac{\kappa - \sqrt{\kappa^2+9}}{3} < 0$, and $\parallel \mathbf{r}_2 \parallel = \frac{\kappa + \sqrt{\kappa^2+9}}{3} > 0$. Thus, this cubic function decreases monotonically between $\parallel \mathbf{r}_1 \parallel$ and $\parallel \mathbf{r}_2 \parallel$.

When $\parallel \mathbf{r} \parallel = 0$, we have:

$$\parallel \mathbf{r} \parallel^3 -\kappa \parallel \mathbf{r} \parallel^2 -3 \parallel \mathbf{r} \parallel +\kappa = \kappa > 0 \tag{23}$$

When $\parallel \mathbf{r} \parallel = 1$, we have:

$$\parallel \mathbf{r} \parallel^3 -\kappa \parallel \mathbf{r} \parallel^2 -3 \parallel \mathbf{r} \parallel +\kappa = 1 - 3\kappa < 0 \tag{24}$$

Therefore, there is one and only one real root between 0 and 1 for this cubic equation.

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.

References

[BAEDR08] BEN-ARTZI A., EGAN K., DURAND F., RAMAMOORTHI R.: A precomputed polynomial representation for interactive brdf editing with global illumination. *ACM Trans. Graph. 27*, 2 (May 2008), 13:1–13:13. 3

[BAOR06] BEN-ARTZI A., OVERBECK R., RAMAMOORTHI R.: Real-time brdf editing in complex lighting. *ACM Trans. Graph. 25*, 3 (July 2006), 945–954. 3

[BDGS05] BANERJEE A., DHILLON I. S., GHOSH J., SRA S.: Clustering on the unit hypersphere using von mises-fisher distributions. *J. Mach. Learn. Res. 6* (Dec. 2005), 1345–1382. 4

[BSD08] BAVOIL L., SAINZ M., DIMITROV R.: Image-space horizon-based ambient occlusion. In *ACM SIGGRAPH 2008 Talks* (New York, NY, USA, 2008), SIGGRAPH '08, ACM, pp. 22:1–22:1. 1, 3

[CK07] COLBERT M., KŘIVÁNEK J.: Gpu-based importance sampling. In *GPU Gems 3*. Addison-Wesley Professional, 2007. 4

[CNS*11] CRASSIN C., NEYRET F., SAINZ M., GREEN S., EISEMANN E.: Interactive indirect illumination using voxel cone tracing: A preview. In *Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2011), I3D '11, ACM, pp. 207–207. 4

[CPC84] COOK R. L., PORTER T., CARPENTER L.: Distributed ray tracing. *SIGGRAPH Comput. Graph. 18*, 3 (Jan. 1984), 137–145. 2

[HF14] HERMANNS L., FRANKE T. A.: Screen space cone tracing for glossy reflections. In *ACM SIGGRAPH 2014 Posters* (New York, NY, USA, 2014), SIGGRAPH '14, ACM, pp. 102:1–102:1. 1, 3

[HSRG07] HAN C., SUN B., RAMAMOORTHI R., GRINSPUN E.: Frequency domain normal map filtering. *ACM Trans. Graph. 26*, 3 (July 2007). 2, 3, 4, 7, 8

[JM12] JAKOB W., MARSCHNER S.: Manifold exploration: A markov chain monte carlo technique for rendering scenes with difficult specular transport. *ACM Trans. Graph. 31*, 4 (July 2012), 58:1–58:13. 2

[JSG09] JIMENEZ J., SUNDSTEDT V., GUTIERREZ D.: Screen-space perceptual rendering of human skin. *ACM Trans. Appl. Percept. 6*, 4 (Oct. 2009), 23:1–23:15. 3

[Kaj86] KAJIYA J. T.: The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1986), SIGGRAPH '86, ACM, pp. 143–150. 3

[KVHS00] KAUTZ J., VÁZQUEZ P.-P., HEIDRICH W., SEIDEL H.-P.: Unified approach to prefiltered environment maps. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000* (London, UK, UK, 2000), Springer-Verlag, pp. 185–196. 6

[LW93] LAFORTUNE E. P., WILLEMS Y. D.: Bi-directional path tracing. In *CompuGraphics '93* (1993), pp. 145–153. 2

[LWDB10] LAURIJSSEN J., WANG R., DUTRÉ P., BROWN B. J.: Fast estimation and rendering of indirect highlights. In *Proceedings of the 21st Eurographics Conference on Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2010), EGSR'10, Eurographics Association, pp. 1305–1313. 2, 3, 4

[LWLD11] LAURIJSSEN J., WANG R., LAGAE A., DUTRÉ P.: Pre-computed gathering of multi-bounce glossy reflections. *Computer Graphics Forum 30*, 8 (2011), 2270–2278. 2, 3

[Mit07] MITTRING M.: Finding next gen: Cryengine 2. In *ACM SIGGRAPH 2007 Courses* (New York, NY, USA, 2007), SIGGRAPH '07, ACM, pp. 97–121. 1, 3

[MKB*05] MERTENS T., KAUTZ J., BEKAERT P., VAN REETH F., SEIDEL H.-P.: Efficient rendering of local subsurface scattering. *Computer Graphics Forum 24*, 1 (2005), 41–49. 1, 3

[ML09] MCGUIRE M., LUEBKE D.: Hardware-accelerated global illumination by image space photon mapping. In *Proceedings of the Conference on High Performance Graphics 2009* (New York, NY, USA, 2009), HPG '09, ACM, pp. 77–89. 3

[NKLN10] NGUYEN C. H., KYUNG M.-H., LEE J.-H., NAM S.-W.: A pca decomposition for real-time brdf editing and relighting with global illumination. *Computer Graphics Forum 29*, 9 (2010), 1469–1478. 3

[RGS09] RITSCHEL T., GROSCH T., SEIDEL H.-P.: Approximating dynamic global illumination in image space. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2009), I3D '09, ACM, pp. 75–82. 3, 6, 7

[RS09] ROBISON A., SHIRLEY P.: Image space gathering. In *Proceedings of the Conference on High Performance Graphics 2009* (New York, NY, USA, 2009), HPG '09, ACM, pp. 91–98. 1, 3

[SA07] SHANMUGAM P., ARIKAN O.: Hardware accelerated ambient occlusion techniques on gpus. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2007), I3D '07, ACM, pp. 73–80. 1, 3

[SZC*07] SUN X., ZHOU K., CHEN Y., LIN S., SHI J., GUO B.: Interactive relighting with dynamic brdfs. *ACM Trans. Graph. 26*, 3 (July 2007). 3

[Vea98] VEACH E.: *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford, CA, USA, 1998. AAI9837162. 2

[VG97] VEACH E., GUIBAS L. J.: Metropolis light transport. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1997), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., pp. 65–76. 2

[WWZ*09] WANG R., WANG R., ZHOU K., PAN M., BAO H.: An efficient gpu-based approach for interactive global illumination. *ACM Trans. Graph. 28*, 3 (July 2009), 91:1–91:8. 3

[XCM*14] XU K., CAO Y.-P., MA L.-Q., DONG Z., WANG R., HU S.-M.: A practical algorithm for rendering interreflections with all-frequency brdfs. *ACM Trans. Graph. 33*, 1 (Feb. 2014), 10:1–10:16. 2, 3

[YWY08] YU X., WANG R., YU J.: Interactive glossy reflections using gpu-based ray tracing with adaptive lod. *Comput. Graph. Forum 27*, 7 (2008), 1987–1996. 2

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.