



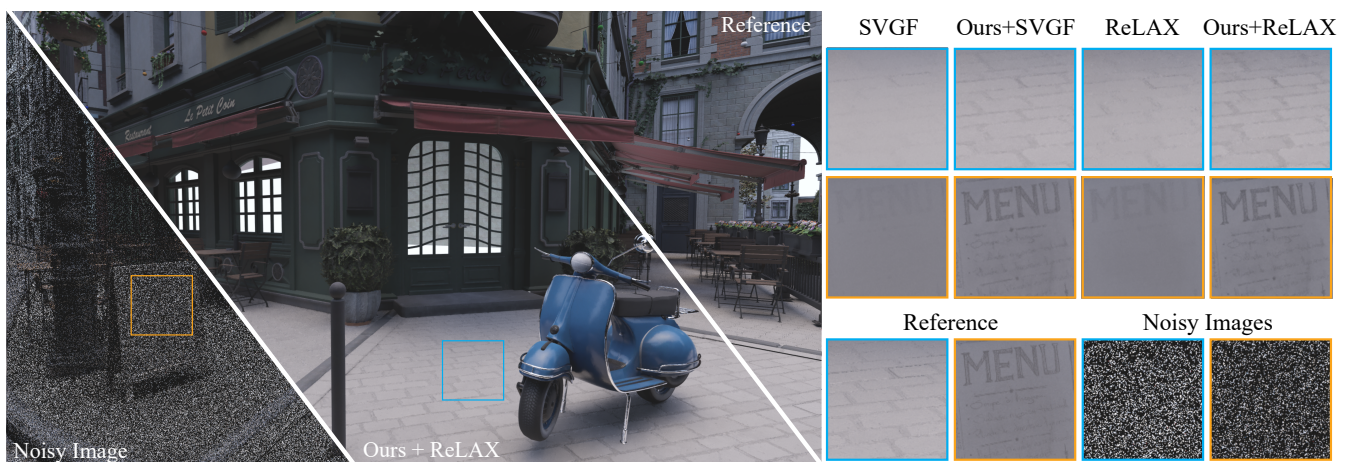


# Real-time Denoising Using BRDF Pre-integration Factorization

Tao Zhuang<sup>1</sup> , Pengfei Shen<sup>1</sup> , Beibei Wang<sup>†2</sup> , and Ligang Liu<sup>1</sup> 

<sup>1</sup>University of Science and Technology of China, China

<sup>2</sup>Nanjing University of Science and Technology, China



**Figure 1:** Our method demodulates the BRDF pre-integration component from filtering and can be easily integrated into existing filtering algorithms. By combining with our method, both SVGF and ReLAX are able to preserve the fine details from BRDF maps much better.

## Abstract

Path tracing has been used for real-time renderings, thanks to the powerful GPU device. Unfortunately, path tracing produces noisy rendered results, thus, filtering or denoising is often applied as a post-process to remove the noise. Previous works produce high-quality denoised results, by accumulating the temporal samples. However, they cannot handle the details from bidirectional reflectance distribution function (BRDF) maps (e.g. roughness map). In this paper, we introduce the BRDF pre-integration factorization for denoising to better preserve the details from BRDF maps. More specifically, we reformulate the rendering equation into two components: the BRDF pre-integration component and the weighted-lighting component. The BRDF pre-integration component is noise-free, since it does not depend on the lighting. Another key observation is that the weighted-lighting component tends to be smooth and low-frequency, which indicates that it is more suitable for denoising than the final rendered image. Hence, the weighted-lighting component is denoised individually. Our BRDF pre-integration demodulation approach is flexible for many real-time filtering methods. We have implemented it in spatio-temporal variance-guided filtering (SVGF), ReLAX and ReBLUR. Compared to the original methods, our method manages to better preserve the details from BRDF maps, while both the memory and time cost are negligible.

## CCS Concepts

- Computing methodologies → Ray tracing;

## 1. Introduction

Path tracing has been widely used in movie production, since it is physically-based and unbiased. Recently, path tracing has also been exploited for real-time applications, thanks to the powerful GPU

<sup>†</sup> Corresponding author. beibei.wang@njust.edu.cn.

device. Unfortunately, path tracing produces noisy rendered results, especially for real-time applications, due to the low ray budget. Monte Carlo denoising or filtering is one way to reduce the noise. Depending on the performance requirements (offline rendering and real-time rendering), the denoising approaches are different. In this paper, we focus on denoising for real-time renderings, which have the following requirements: low ray budgets, high-performance denoising process and temporal stability.

A lot of efforts have been made on real-time Monte Carlo denoising. Schied et al. [SKW\*17] proposed a spatio-temporal variance-guided filtering approach (SVGF) to filter noisy path traced images rendered with 1 sample per pixel (spp). Their filter leverages prior frames' samples to preserve details and decouple sources of noise. To preserve the high-frequency details from textures, they demodulate the diffuse albedo from the pixel color before filtering and multiply the diffuse albedo back after filtering. Recently, NVIDIA released two powerful denoisers: ReLAX and ReBLUR [Zhd], where ReLAX is a variant of SVGF optimized for denoising raytraced specular and diffuse signals and ReBLUR uses recurrent blurring to improve the denoising quality and temporal stability, which is about two times faster than SVGF. In ReBLUR, they guide the filtering process with normal, surface roughness and other factors, to avoid over-blurring when using roughness maps. It works well for surfaces with small roughness, but fails for spatial-varying large roughness. Furthermore, this guiding strategy introduces extra parameters for tweaking.

Another line of methods ([CSS\*17] and [MZV\*20]) leverage neural networks for real-time denoising. Both of them are able to produce high-quality denoised results, however, their performance is much slower than SVGF, which prevents them from being used directly in real-time applications.

In this paper, we present a BRDF pre-integration demodulation filtering approach, which can preserve the fine details from spatial-varying BRDF maps, like roughness maps, with negligible cost. Specifically, we reformulate the rendering equation into two components: the *BRDF pre-integration* component and the weighted-lighting component. The BRDF pre-integration component is precomputed. The weighted-lighting component is evaluated with Monte Carlo path tracing and is later denoised individually. Then the BRDF pre-integration is multiplied with the denoised weighted-lighting component to obtain the final result. Our method is based on two key insights: the BRDF pre-integration component is noise-free and the weighted-lighting component is low-frequency. The noise-free BRDF pre-integration is capable of preserving the BRDF map details and the low frequency of the weighted-lighting components enables a better filtering. We implement our method in SVGF, ReLAX and ReBLUR, respectively. In all these filtering algorithms, our method is able to preserve the details of BRDF maps much better, while both the memory and time cost is negligible. As far as we know, our method is the first to use BRDF pre-integration factorization for denoising. Our contributions include:

- reformulating the rendering equation into a noise-free BRDF pre-integration component and a low-frequency weighted-lighting component, and
- introducing a BRDF pre-integration demodulation filtering ap-

proach, which enables preserve the material details for existing filtering algorithms.

In the next section, we review some of the previous works on Monte Carlo denoising and BRDF pre-integration. Then, we recap the theoretical basis of our method in Section 3 and present our method in Section 4. We present our results and analyze performances in Section 5, and then conclude in Section 6.

## 2. Previous Work

In this section, we first briefly go over the offline Monte Carlo denoising methods, and then review the real-time denoising methods.

**Offline Denoising.** Image space-based approaches have achieved impressive results at a reduced sampling rate [SZR\*15]. They treated denoising as a regression problem, and used different regression models for filtering: zero-order linear regression model ([SD12], [RMZ13], [MJL\*13] and [ZRJ\*15]), first-order or higher-order models ([MCY14], [BRM\*16] and [MMMG14]). The zero-order models have less flexibility, due to the limitations of their explicit filters.

Recently, deep learning-based methods ([KBS15], [BVM\*17], [VRM\*18], [YWY\*19], [WW19], [LWWH20], and [XZW19]) have been successfully exploited for Monte Carlo denoising. They are able to denoise images rendered with pretty high sampling rate. Later, sample-based approaches ([GLA\*19] and [LWY\*21]) further improve the denoising quality for renderings with low sampling rate, at the cost of both time and memory, which was later improved by Munkberg et al. [MH20] via a layering embedding approach.

All these methods are not applicable for real-time renderings, due to either the high sampling rate requirement or the expensive denoising cost.

**Real-time Denoising.** Different from the offline denoising, the real-time denoising has strict requirements: low sample rate of the input noisy image (1 spp), critical denoising performance and temporal stability.

Bilateral filtering [TM98] was commonly used for real-time rendering, which includes guided image filtering [BEM11] and wavelet method [DSHL10]. All of these methods reuse the samples from the spatial domain. Since the input noisy image only has 1 spp, reusing samples from temporal domain allows to produce cleaner images. Temporal anti-aliasing [Kar14] is widely used in video games. Schied et al. [SKW\*17] proposed spatio-temporal variance guided filtering (SVGF), which greatly improves the temporal stability. SVGF is able to preserve the details from diffuse map, but fails for the other maps, e.g. roughness map. Later, SVGF was improved by Schied et al. [SPD18] and Zeng et al. [ZLY\*21] respectively, by utilizing temporal gradient to reduce lag and ghosting, or correctly handling motion vectors of shadows, glossy reflections and occlusions. Mara et al. [MMBJ17] introduced a factored approximation of a material-based Monte Carlo integrator and use it for filtering. They separate the Fresnel term from the rendering equation, which is also not helpful for the BRDF maps. Recently, NVIDIA proposed two denoisers: ReLAX and ReBLUR [Zhd],

where ReLAX is a variant of SVGF optimized for denoising ray-traced specular and diffuse signals and ReBLUR uses recurrent blurring to improve the denoising quality and the temporal stability, which is about two times faster than SVGF. In ReBLUR, they guide the filtering with normal, surface roughness, and other factors, to avoid over-blurring. ReBLUR is suitable for surfaces with small roughness, but fails for spatial-varying large roughness. Furthermore, this guiding strategy yields extra parameters for tweaking. In this paper, we factorize the BRDF pre-integration from filtering to better preserve the details from BRDF maps, which is simple and does not require any tweaking.

Machine learning has also been used for real-time denoising. Chaitanya et al. [CSS\*17] proposed a recurrent neural network (RNN) model to denoise under-sampled video renderings at interactive frame rate, which is also not suitable for real-time applications. Recently, Meng et al. [MZV\*20] proposed a more efficient network, using neural bilateral grid, which is faster than Chaitanya et al. [CSS\*17], but is still heavy (the simplest version is about two times slower than SVGF). Our method could be integrated in these methods, and we leave it for the further work.

**BRDF Pre-integration.** The radiance of a shading point is computed as the integral of the multiplication of the lighting and the BRDF over the incoming direction. Both lighting and BRDF are functions of incoming direction, and they are not separable in theory. Karis [KAR13] separates the lighting and the BRDF for environment lighting as an approximation, and precomputes both the BRDF and the lighting integration, which avoids the expensive sampling at run-time, but leads to obvious difference with the rendering equation. Wang et al. [WDH20] extended the BRDF pre-integration to real-time glints rendering. Stachowiak [STA15] introduced the ratio estimator to the rendering equation, by moving the BRDF and the cosine term outside the integral and precomputing them, resulting in less variance. Compared to their work, we use a noise-free BRDF term rather than a simulated value, which yields better filtering results. Heitz et al. [HHM18] presented a shadow denoising approach with ratio estimator to reduce variance. However, the BRDF pre-integration in their method requires an analytical solution for the unshadowed direct illumination, which is feasible for their application, but impossible for ours.

Inspired by these works, we also factorize a BRDF pre-integration term by re-arranging the rendering equation, and then we use this term for denoising. Compared to previous works, our method is the first time, to our best knowledge, to use BRDF pre-integration factorization in denoising for modern real-time rendering. The novelty has two folds. First, we use pre-integrated BRDF factorizations to help with demodulation prior to denoising in low sample rate real-time path tracers. Second, the derivation of BRDF pre-integration is different from the previous ones.

### 3. Background and Motivation

#### 3.1. The Rendering Equation

The outgoing radiance  $L(\omega_o)$  of a shading point seen from a pixel is computed with the Rendering Equation [Kaj86]:

$$L(\omega_o) = \int_{\Omega} f_r(\omega_i, \omega_o) L(\omega_i) \cos \theta_i d\omega_i, \quad (1)$$

where  $\omega_o$  and  $\omega_i$  represent the outgoing direction and the incoming direction, respectively, and  $f_r(\omega_i, \omega_o)$  represents the BRDF at the shading point.

Monte Carlo based approaches (e.g. path tracing) are used to solve Equation 1. In real-time rendering, low sampling rate (e.g. 1spp) is used, yielding noisy results.

Depending on the types of materials,  $f_r(\omega_i, \omega_o)$  is usually categorized into diffuse and specular in real-time rendering. The diffuse BRDF is independent of the angular domain, hence we have:

$$L(\omega_o) = f_r^{\text{diffuse}} \int_{\Omega} L(\omega_i) \cos \theta_i d\omega_i. \quad (2)$$

This formula enables a noise-free BRDF value  $f_r^{\text{diffuse}}$  and makes the separation of the material and lighting possible for denoising.

#### 3.2. Monte Carlo Denoising

Monte Carlo denoising aims at finding a reasonable filter  $\Phi$  and corresponding parameters  $\theta$ , given the input data  $x$  from a rendering pipeline, to output a noise-free image  $\hat{c}$ :

$$\hat{c} = \Phi(x; \theta), \quad (3)$$

where  $x$  consists of the pixels' radiance  $c$  and the optional G-buffers.

Instead of denoising the pixel's radiance directly, the albedo modulation is usually performed before filtering, by removing the effects of diffuse map (using Equation 2), and then including them back after filtering. The high-frequency details could be preserved with this albedo demodulation.

#### 3.3. Motivation

However, in practice, physically-based material models (e.g. the Cook-Torrance model [CT82]) are usually preferred, with the following formula:

$$f_r(\omega_i, \omega_o) = \frac{F(\omega_i, \omega_h) D(\omega_h) G(\omega_i, \omega_o)}{4|\omega_h \cdot \omega_i| |\omega_h \cdot \omega_o|}, \quad (4)$$

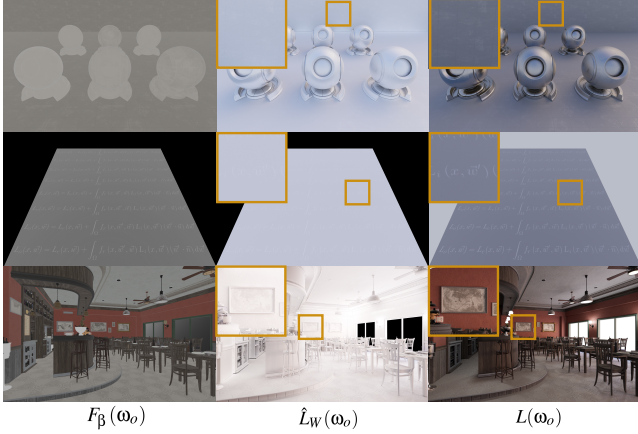
where  $\omega_h$  is the half vector between  $\omega_i$  and  $\omega_o$ ,  $F(\omega_i, \omega_h)$  is the Fresnel term,  $D(\omega_h)$  is the normal distribution function (NDF) and  $G(\omega_i, \omega_o)$  is the shadowing-masking function. The NDF has an importance parameter – roughness, which mainly affects the material appearance.

With the Cook-Torrance model as BRDF, the same solution as the diffuse albedo demodulation does not hold, since the effect of the roughness on the final radiance is not linear. It is not obvious to derive a formula with the BRDF outside the integral, equivalent to Equation 2.

Instead we propose a BRDF pre-integration factorization approach which separates the BRDF into an analytical component and a numerical component. The analytical component is noise-free and is able to preserve high-frequency details in BRDF maps.

## 4. Our Method

In this section, we first reformulate the rendering equation, by introducing a BRDF pre-integration term (Sec. 4.1), and then propose to demodulate the BRDF pre-integration term for filtering (Sec. 4.2). Next, we implement our method in three existing filtering approaches (SVGF, ReLAX and ReBLUR) to improve their denoising quality (Sec. 4.3).



**Figure 2:** Visualization of  $F_{\beta}$ ,  $\hat{L}_W$  and  $L$ . As shown in the insets, the  $\hat{L}_W$  has less details than  $L$ .

### 4.1. BRDF Pre-Integration

We reformulate Equation 1 as:

$$L(\omega_o) = \int_{\Omega} f_r(\omega_i, \omega_o) L(\omega_i) \cos \theta_i d\omega_i, \quad (5)$$

$$= F_{\beta}(\omega_o) \int_{\Omega} \frac{f_r(\omega_i, \omega_o) L(\omega_i) \cos \theta_i d\omega_i}{F_{\beta}(\omega_o)}, \quad (6)$$

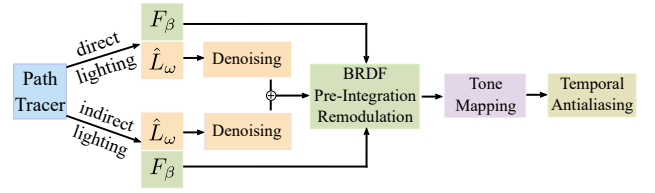
where

$$F_{\beta}(\omega_o) = \int_{\Omega} f_r(\omega_i, \omega_o) \cos \theta_i d\omega_i. \quad (7)$$

Inspired by Karis [KAR13],  $F_{\beta}(\omega_o)$  can be precomputed as a function of  $\cos \theta_o$  and the surface roughness  $\alpha$ , when the NDF is isotropic and the Fresnel term is approximated with the Schlick approximation, resulting in a 2D table. In our implementation, we tabulate  $F_{\beta}(\omega_o)$  with a resolution of  $256 \times 256$ , where each pixel is estimated by sampling  $\omega_i$  with 1024 samples. We will show how to use Equation 6 for filtering in the next section.

### 4.2. BRDF Demodulation for Filtering

If the radiance  $L(\omega_o)$  in Equation 1 is filtered directly, the fine details are blurred. Thus, previous works, such as [BVM\*17] and [SKW\*17], demodulate the diffuse albedo before filtering and then remodulate it with the filtered results, or remodulate the non-visibility term ([HHM18]) with the filtered shadows. Instead we propose to demodulate the BRDF pre-integration term  $F_{\beta}(\omega_o)$ .



**Figure 3:** Overview of the integration of our method into SVGF. Starting from the buffers  $\hat{L}_W$  and  $F_{\beta}$  (direct lighting and indirect lighting separately) produced by the path tracer, we denoise the  $\hat{L}_W$  buffers first and then multiply the denoised  $\hat{L}_W$  with  $F_{\beta}$  to get the denoised results. In the end, we perform tonemapping and temporal antialiasing (TAA) in the post-process.

For clarity reasons, we rewrite Equation 6 into two components: a BRDF pre-integration component  $F_{\beta}(\omega_o)$  and a weighted-lighting component  $L_W(\omega_o)$ .

$$L(\omega_o) = F_{\beta}(\omega_o) L_W(\omega_o), \quad (8)$$

where

$$L_W(\omega_o) = \int_{\Omega} \frac{f_r(\omega_i, \omega_o) \cos \theta_i L(\omega_i)}{F_{\beta}(\omega_o)} d\omega_i. \quad (9)$$

$L_W(\omega_o)$  can be rewritten as

$$L_W(\omega_o) = \int_{\Omega} W(\omega_o, \omega_i) L(\omega_i) d\omega_i, \quad (10)$$

where  $W(\omega_o, \omega_i) = f_r(\omega_i, \omega_o) \cos \theta_i / F_{\beta}(\omega_o)$ .  $L_W(\omega_o)$  is the integral of the incident lighting weighted by the BRDF value, called *weighted-lighting* term.  $W(\omega_o, \omega_i)$  is the normalized BRDF and here serves as a normalized filtering kernel for the lighting.

Now, the radiance of a pixel consists of two terms:  $F_{\beta}(\omega_o)$  and  $L_W(\omega_o)$ .  $F_{\beta}(\omega_o)$  is precomputed with a 2D table and is noise-free.  $L_W(\omega_o)$  is evaluated with the Monte Carlo path tracing at run-time and estimated with  $\hat{L}_W(\omega_o)$ :

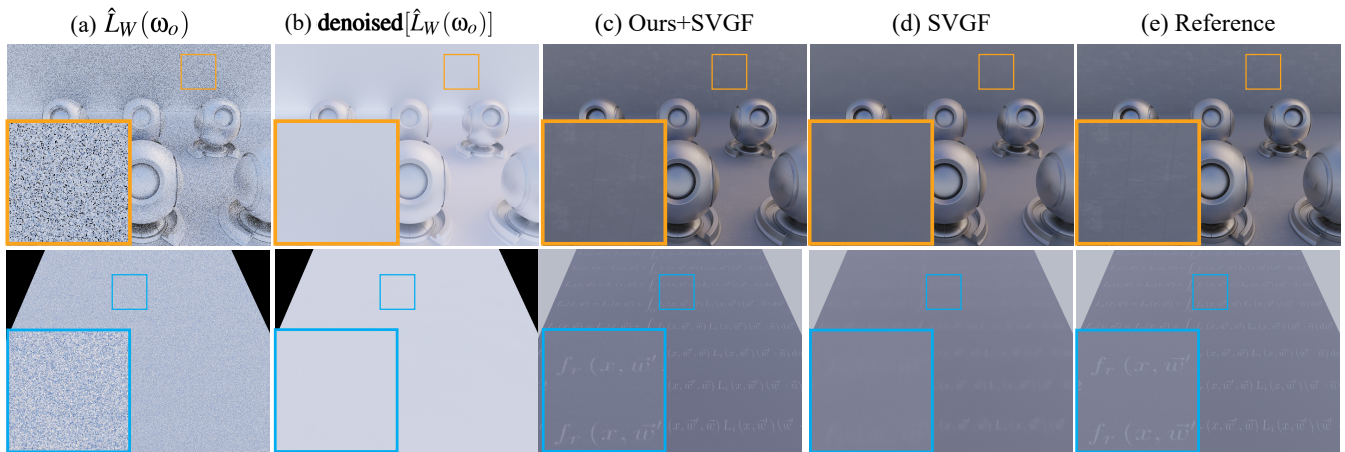
$$\hat{L}_W(\omega_o) \approx \sum_k \frac{W(\omega_o, \omega_i^{(k)}) L(\omega_i^{(k)})}{p(\omega_i^{(k)})}, \quad (11)$$

where  $k$  is the samples per pixel and  $p(\omega_i)$  is the probability density function (PDF) for importance sampling. Both  $\hat{L}_W(\omega_o)$  and  $F_{\beta}(\omega_o)$  are the outputs of renderings.

After rendering,  $\hat{L}_W(\omega_o)$  is denoised individually and then multiplied with  $F_{\beta}(\omega_o)$  to get the final pixel color:

$$L(\omega_o) \approx F_{\beta}(\omega_o) \text{denoised}[\hat{L}_W(\omega_o)]. \quad (12)$$

**Insights.** In our approach, the key point is the demodulation and remodulation of the BRDF pre-integration term  $F_{\beta}(\omega_o)$ . There are two insights behind this. First,  $F_{\beta}(\omega_o)$  is able to represent the BRDF on the surface, which might include spatial-varying BRDF maps, e.g. the roughness map. Thanks to the precomputation of  $F_{\beta}$ , our method gets a noise-free estimation of the BRDF integration term, to better preserve the details in BRDF maps. Second,  $\hat{L}_W$  tends to be smoother than  $L$ , without the BRDF pre-integration component. Actually, it is more reasonable to filter the



**Figure 4:** With the noisy  $\hat{L}_W$  as the input (a), we denoise it first (b) and then remodulate with BRDF pre-integration to get the final result (c). Compared to SVGF (d), our method preserves the details much better.

low-frequency term  $\hat{L}_W$  than  $L$ . In Figure 2, we show  $\hat{L}_W$  and  $L$  computed with high sampling rate, and it's obvious that  $\hat{L}_W$  has less details than  $L$ .

Existing approaches ([SKW\*17]) demodulate the diffuse albedo before filtering and remodulate it back after filtering to preserve the details from the diffuse map. This idea only works for diffuse BRDF, and it's not applicable for microfacet model with other BRDF maps (e.g., the roughness map), while our method is suitable for any SVBRDF maps, including roughness maps, normal maps and diffuse maps, which are commonly used in real-time rendering.

### 4.3. Integrations to Existing Filtering Methods

Our BRDF pre-integration demodulation approach is flexible and can be universally integrated for many real-time filtering methods. We implement our method for three existing filtering methods: SVGF, ReLAX and ReBLUR. All these methods have achieved higher quality, thanks to our method.

**BRDF Demodulation for SVGF.** We follow the main implementation of SVGF, except for the following changes (see Figure 3):

- During rendering, the renderer produces the  $\hat{L}_W$  and  $F_\beta$  as the outputs for each frame.
- During reconstruction, the  $\hat{L}_W$  buffers from the current frame and the prior frames are filtered, resulting in the denoised  $\hat{L}_W$ . Then, the denoised  $\hat{L}_W$  remodulate with the  $F_\beta$  to get the pixel color.
- If the shading point has a diffuse BRDF, we perform the same operation as SVGF.

**BRDF Demodulation for ReLAX and ReBLUR.** Similar to SVGF, we also integrate our method into ReLAX and ReBLUR. We treat the original methods as a black box, and only change the inputs and the outputs of the filtering from the  $L$  buffer to the  $\hat{L}_W$  buffer and change the demodulation from the diffuse albedo to  $F_\beta$ .

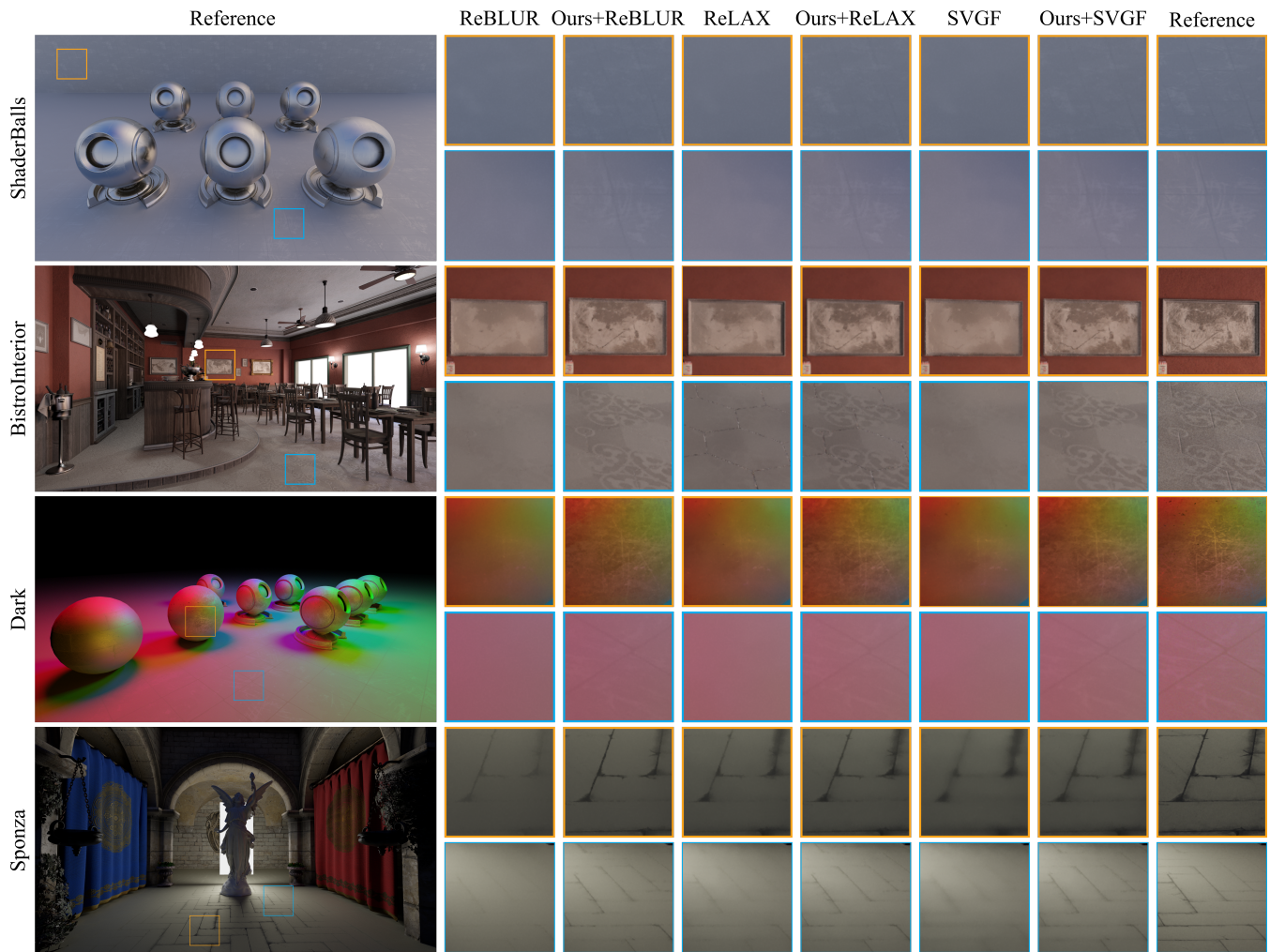
## 5. Results and Discussion

We implement our method in three existing filtering approaches: SVGF, ReLAX and ReBLUR. For ReLAX and ReBLUR, we use the sample code of NVIDIA Ray-tracing Denoiser (NRD) and modify the original methods with our method. For SVGF, we implemented the original version of SVGF [SKW\*17] in the sample renderer of NRD, for better comparison. All timings in this section are measured on a RTX 3070 GPU. For all the comparisons, the resolution of the images is set as  $1920 \times 1080$ . We use the results rendered with path tracing (4096 spp) as the reference. The difference with the reference is measured with RMSE (Root Mean Square Error) and SSIM (Structural SIMilarity). Regarding the microfacet models, we use GGX as the normal distribution function and Schlick approximation for the Fresnel term.

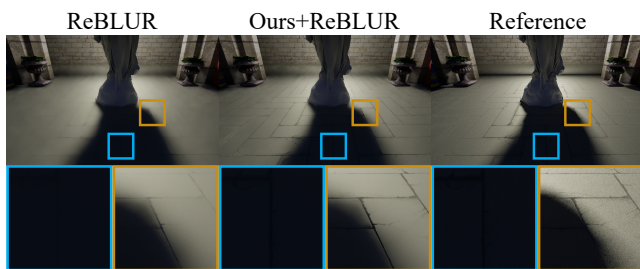
**Quality Validation.** In Figures 1 and 5, we apply our method to three existing methods, i.e., SVGF, ReLAX and ReBLUR, and compare them with the original methods on five scenes. In Figure 4, we show the intermediate buffers ( $\hat{L}_W$  and  $\text{denoised}[\hat{L}_W(\omega_o)]$ ) in our results for the ShaderBalls and Word scenes. All scenes have roughness maps. Our method is able to preserve fine details from BRDF maps and improves the filtering quality significantly, while the details are missing in the original methods. The errors for all these methods with the references are reported in Table 1. That combining our method with the original methods improves the filtering quality, while the error is subtle to the original methods.

In Figure 6, we compare our result (ours + ReBLUR) with ReBLUR on shadow areas. By comparison, our method produces similar results as the original methods.

**Performance Measurement.** The run-time costs of both our method and the original methods are reported in Table 1. Our run-time cost is almost identical to the original methods. In the pre-computation step, our method has a small overhead (1 seconds and 128KB), due to the precomputation of  $F_\beta$ , but  $F_\beta$  can be reused for any isotropic microfacet model. Therefore, with only a small pre-



**Figure 5:** Comparison among SVGF (ours), SVGF, ReLAX (ours), ReLAX and ReBLUR (ours) and ReBLUR.



**Figure 6:** Comparison between our method and the original methods on the scenario with shadows and shadow edges.

computation time and storage cost, our method improves filtering quality significantly.

**Temporal Stability.** In Figure 7, we compare the temporal stability of our method and the original methods (SVGF and ReBLUR)

on the BistroInterior scene [Lum17], using static light, camera and scene settings. The temporal error is measured by the average luminance of the difference between consecutive frames. By comparison results our method has the same temporal stability as the original methods. All these methods have large temporal errors at the beginning due to the missing of the history information and then they become stable.

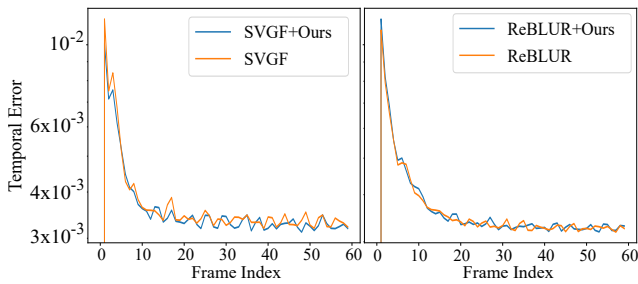
**Limitation and Discussion.** Our method only handles isotropic material for now, but it can be easily extended for anisotropic materials, by including one more dimension for BRDF pre-integration term. Our method treats the original methods as a black box and only changes their inputs and outputs. We believe that a deeper coupling with the original methods will further improve the filtering quality, e.g., using  $F_{\beta}$  to guide the filtering radius.

## 6. Conclusion

In this paper, we have proposed a BRDF pre-integration factorization denoising approach, via reformulating the rendering equation

**Table 1:** RMSE and SSIM of our methods (combined with SVGF, ReLAX and ReBLUR) and the original methods for our test scenes. The highest quality is shown in bold. For RMSE, a smaller value means higher quality; for SSIM, a larger value means higher quality.)

	BistroExterior			BistroInterior			Dark			ShaderBalls			Sponza		
	RMSE	SSIM	T(ms)	RMSE	SSIM	T(ms)	RMSE	SSIM	T(ms)	RMSE	SSIM	T(ms)	RMSE	SSIM	T(ms)
SVGF(ours)	0.0311	0.9795	17.31	0.0357	0.9719	13.00	<b>0.0184</b>	<b>0.9932</b>	14.29	0.0166	0.9689	8.80	0.0217	0.9756	31.61
SVGF	0.0314	0.9791	17.15	0.0367	0.9710	12.87	0.0185	0.9931	14.20	0.0166	0.9687	8.75	0.0217	0.9754	31.51
ReBLUR(ours)	0.0337	0.9760	15.56	<b>0.0355</b>	0.9722	11.59	0.0249	0.9896	13.26	0.0208	0.9577	7.30	0.0260	0.9653	15.99
ReBLUR	0.0353	0.9738	15.34	0.0366	0.9711	11.29	0.0253	0.9896	13.03	0.0215	0.9574	7.16	0.0265	0.9637	15.70
ReLAX(ours)	<b>0.0297</b>	<b>0.9813</b>	15.88	0.0397	<b>0.9724</b>	11.55	0.0184	0.9932	13.48	<b>0.0156</b>	<b>0.9726</b>	7.37	<b>0.0205</b>	<b>0.9780</b>	30.18
ReLAX	0.0297	0.9813	15.71	0.0400	0.9719	11.48	0.0186	0.9930	13.31	0.0167	0.9683	7.32	0.0206	0.9779	29.96

**Figure 7:** Temporal stability of various algorithms (SVGF (ours), SVGF, ReBLUR (ours) and ReBLUR), with temporal error, which is computed as the average luminance of the per-pixel differences, for a fixed view and lighting configuration. By comparison, combination of our method with these methods keeps the temporal stability.

into two components: the BRDF pre-integration component and the weighted-lighting component. This new formula allows leaving the low-frequency part for filtering, and avoids the noise-free high-frequency part from over-blurring. It is easily integrated into any existing filtering approaches (e.g. SVGF, ReLAX and ReBLUR), and improves the filtering quality significantly, with negligible runtime cost.

We have implemented our approach on top of three non-learning filtering methods, and we believe that the learning-based denoising methods can also benefit from our method.

**Acknowledgments.** We thank the reviewers for their valuable comments. This work has been partially supported by the National Natural Science Foundation of China under grant No. 62025207, 62172220, and 61802187, the Fundamental Research Funds for the Central Universities No. 30920021133, China Postdoctoral Science Foundation under grant No.2020M671500, and Jiangsu Postdoctoral Science Foundation under grant 2020Z165.

## References

- [BEM11] BAUSZAT P., EISEMANN M., MAGNOR M.: Guided image filtering for interactive high-quality global illumination. In *Computer Graphics Forum* (2011), vol. 30, Wiley Online Library, pp. 1361–1368. 2
- [BRM\*16] BITTERLI B., ROUSSELLE F., MOON B., A.IGLESIAS-GUITIÁN J., ADLER D., MITCHELL K., JAROSZ W., NOVÁK J.: Non-

linearly weighted first-order regression for denoising Monte Carlo renderings. *Computer Graphics Forum* 35, 4 (2016), 107–117. 2

- [BVM\*17] BAKO S., VOGELS T., MCWILLIAMS B., MEYER M., NOVÁK J., HARVILL A., SEN P., DEROSE T., ROUSSELLE F.: Kernel-predicting convolutional networks for denoising Monte Carlo renderings. *ACM Transactions on Graphics (TOG) (Proceedings of SIGGRAPH 2017)* 36, 4 (July 2017). 2, 4
- [CSS\*17] CHAITANYA C. R., S.KAPLAYAN A., SCHIED C., SALVI M., LEFOHN A., NOWROUZSAHRA D., AILA T.: Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder. *ACM Trans. Graph.* 36, 4 (July 2017), 98:1–98:12. 2, 3
- [CT82] COOK R. L., TORRANCE K. E.: A Reflectance Model for Computer Graphics. *ACM Transactions on Graphics (TOG)* 1, 1 (1982), 7–24. 3
- [DSHL10] DAMMERTZ H., SEWITZ D., HANIKA J., LENSCH H. P.: Edge-avoiding a-trous wavelet transform for fast global illumination filtering. In *Proceedings of the Conference on High Performance Graphics* (2010), Citeseer, pp. 67–75. 2
- [GLA\*19] GHARBI M., LI T.-M., AITTALA M., LEHTINEN J., DURAND F.: Sample-based Monte Carlo denoising using a kernel-splatting network. *ACM Trans. Graph.* 38, 4 (2019), 125:1–125:12. 2
- [HHM18] HEITZ E., HILL S., MCGUIRE M.: Combining analytic direct illumination and stochastic shadows. In *Proceedings of I3D* (2018), I3D '18. 3, 4
- [Kaj86] KAJIYA J. T.: The rendering equation. *SIGGRAPH* 20, 4 (Aug. 1986), 143–150. 3
- [KAR13] KARIS B.: Real shading in unreal engine 4. *SIGGRAPH Courses: Physically Based Shading in Theory and Practice*. 3, 4
- [Kar14] KARIS B.: *High Quality Temporal Supersampling*. Tech. rep., 2014. URL: <http://advances.realtimerendering.com/s2014/index.html>. 2
- [KBS15] KALANTARI N. K., BAKO S., SEN P.: A machine learning approach for filtering Monte Carlo noise. *ACM Transactions on Graphics (TOG) (Proceedings of SIGGRAPH 2015)* 34, 4 (2015). 2
- [Lum17] LUMBERYARD A.: Amazon lumberyard bistro, open research content archive (orca), July 2017. <http://developer.nvidia.com/orca/amazon-lumberyard-bistro>. URL: <http://developer.nvidia.com/orca/amazon-lumberyard-bistro>. 6
- [LWWH20] LIN W., WANG B., WANG L., HOLZSCHUCH N.: A detail preserving neural network model for monte carlo denoising. *Computational Visual Media Journal* 6 (2020), 157–168. 2
- [LWY\*21] LIN W., WANG B., YANG J., WANG L., YAN L.: Path-based Monte Carlo Denoising Using a Three-Scale Neural Network. *Computer Graphics Forum* (2021). 2
- [MCY14] MOON B., CARR N., YOON S.-E.: Adaptive rendering based on weighted local regression. *ACM Trans. Graph* 33, 5 (2014), 170:1–170:14. 2

- [MH20] MUNKBERG J., HASSELGREN J.: Neural denoising with layer embeddings. *Computer Graphics Forum* 39, 4 (2020), 1–12. 2
- [MJL\*13] MOON B., JUN J. Y., LEE J., KIM K., HACHISUKA T., YOON S.-E.: Robust image denoising using a virtual flash image for Monte Carlo ray tracing. *Computer Graphics Forum* 32, 1 (2013), 139–151. 2
- [MMBJ17] MARA M., MCGUIRE M., BITTERLI B., JAROSZ W.: An efficient denoising algorithm for global illumination. *High Performance Graphics* 10 (2017), 3105762–3105774. 2
- [MMMG14] MOON B., MCDONAGH S., MITCHELL K., GROSS M.: Adaptive polynomial rendering. *ACM Trans. Graph* (2014), 10. 2
- [MZV\*20] MENG X., ZHENG Q., VARSHNEY A., SINGH G., ZWICKER M.: Real-time Monte Carlo Denoising with the Neural Bilateral Grid. In *Eurographics Symposium on Rendering - DL-only Track* (2020), Dachsbacher C., Pharr M., (Eds.). 2, 3
- [RMZ13] ROUSSELLE F., MANZI M., ZWICKER M.: Robust denoising using feature and color information. *Computer Graphics Forum* 32, 7 (2013), 121–130. 2
- [SD12] SEN P., DARABI S.: On filtering the noise from the random parameters in Monte Carlo rendering. *ACM Transactions on Graphics* 31, 3 (2012), 15. 2
- [SKW\*17] SCHIED C., KAPLANYAN A., WYMAN C., PATNEY A., CHAITANYA C. R. A., BURGESS J., LIU S., DACHSBACHER C., LEFOHN A., SALVI M.: Spatiotemporal variance-guided filtering: Real-time reconstruction for path-traced global illumination. In *Proceedings of High Performance Graphics* (2017), HPG '17, Association for Computing Machinery. 2, 4, 5
- [SPD18] SCHIED C., PETERS C., DACHSBACHER C.: Gradient estimation for real-time adaptive temporal filtering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1, 2 (2018), 1–16. 2
- [STA15] STACHOWIAK T.: Stochastic screen-space reflections. In *ACM SIGGRAPH Courses 2015: Advances in Real-Time Rendering in Games* (2015). 3
- [SZR\*15] SEN P., ZWICKER M., ROUSSELLE F., YOON S.-E., KALANTARI N.: Denoising your Monte Carlo renders: Recent advances in image-space adaptive sampling and reconstruction. *ACM SIGGRAPH 2015 Courses* (2015). 2
- [TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. In *ICCV* (1998), IEEE, pp. 839–846. 2
- [VRM\*18] VOGELS T., ROUSSELLE F., MCWILLIAMS B., RÖTHLIN G., HARVILL A., ADLER D., MEYER M., NOVÁK J.: Denoising with kernel prediction and asymmetric loss functions. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2018)* 37, 4 (2018), 124:1–124:15. 2
- [WDH20] WANG B., DENG H., HOLZSCHUCH N.: Real-time glints rendering with pre-filtered discrete stochastic microfacets. *Computer Graphics Forum* 39, 6 (2020), 144–154. 3
- [WW19] WONG K.-M., WONG T.-T.: Deep residual learning for denoising monte carlo renderings. *Computational Visual Media* 5 (09 2019). doi:10.1007/s41095-019-0142-3. 2
- [XZW19] XU B., ZHANG J., WANG R.: Adversarial monte carlo denoising with conditioned auxiliary feature modulation. *ACM Transactions on Graphics* 38, 6 (2019), 1–12. 2
- [YWY\*19] YANG X., WANG D., YIN B., WEI X., HU W., ZHAO L., ZHANG Q., FU H.: Demc: A deep dual-encoder network for denoising monte carlo rendering. *Journal of Computer Science and Technology* 34, 5 (2019), 1123–1135. 2
- [Zhd] ZHDAN D.: Fast denoising with self stabilizing recurrent blurs (presented by nvidia). GDC 2020. 2
- [ZLY\*21] ZENG Z., LIU S., YANG J., WANG L., YAN L.-Q.: *Temporally Reliable Motion Vectors for Real-time Ray Tracing*. Tech. Rep. 2, 2021. 2
- [ZRJ\*15] ZIMMER H., ROUSSELLE F., JAKOB W., WANG O., ADLER D., JAROSZ W., SORKINE-HORNUNG O., SORKINE-HORNUNG A.: Path-space motion estimation and decomposition for robust animation filtering. *Computer Graphics Forum* 34, 4 (2015), 131–142. 2