Volume 36 (2017), Number 8 pp. 600–612 COMPUTER GRAPHICS *forum*

Noise Reduction on G-Buffers for Monte Carlo Filtering

Bochang Moon[1,2], Jose A. Iglesias-Guitian[1], Steven McDonagh[1,3], Kenny Mitchell[1,4]

[1]Disney Research, [2]Gwangju Institute of Science and Technology, [3]Imperial College London, [4]Edinburgh Napier University

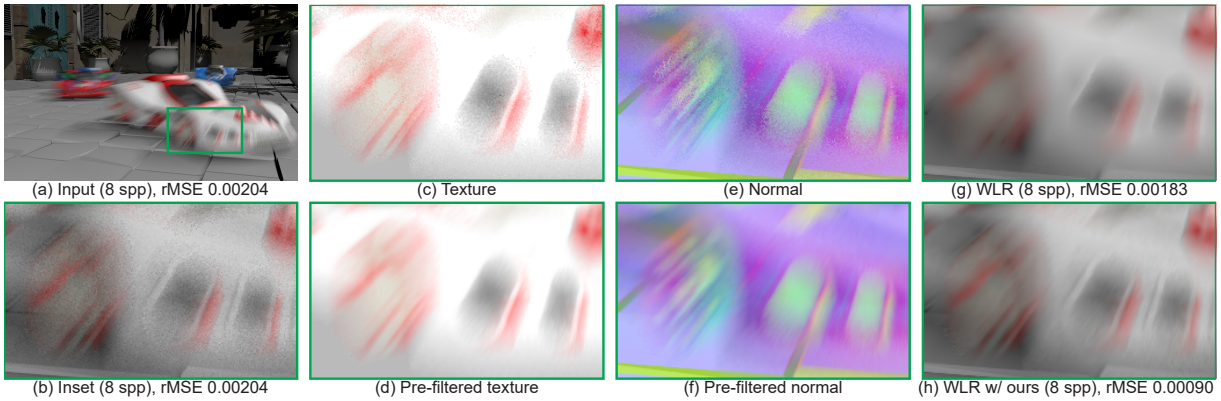

Figure 1: *Our results for the Cars scene where each car has a different motion. The input image (a), and its close-up (b), are generated using 8 samples per pixel (spp) allocated by an adaptive sampler using weighted local regression (WLR) [MCY14]. Recent filtering methods utilize geometric buffers (G-buffers) such as texture (c), normal (e), and depth, which may contain severe noise in regions with strong motion blur effects. As a result, the state-of-the-art method (g) produces over- and under-blurred results in those regions. Our method applies an anisotropic pre-filtering to the noisy feature buffers and generates the pre-filtered G-buffers (d) and (f). The recent filter (h) that utilizes our results instead of the noisy G-buffers, shows a reduced error, i.e., the relative mean squared error (rMSE) [RKZ11], and better preserved edges thanks to our high-quality pre-filtering.*

Abstract

We propose a novel pre-filtering method that reduces the noise introduced by depth-of-field and motion blur effects in geometric buffers (G-buffers) such as texture, normal and depth images. Our pre-filtering uses world positions and their variances to effectively remove high-frequency noise while carefully preserving high-frequency edges in the G-buffers. We design a new anisotropic filter based on a per-pixel covariance matrix of world position samples. A general error estimator, Stein's unbiased risk estimator, is then applied to estimate the optimal trade-off between the bias and variance of pre-filtered results. We have demonstrated that our pre-filtering improves the results of existing filtering methods numerically and visually for challenging scenes where depth-of-field and motion blurring introduce a significant amount of noise in the G-buffers.

Categories and Subject Descriptors (according to ACM CCS): Three-Dimensional Graphics and Realism [I.3.7]: Raytracing—

1. Introduction

Monte Carlo (MC) ray tracing methods, including distributed ray tracing [CPC84] and path tracing [Kaj86], are widely accepted to numerically solve the rendering equation, since they allow the efficient rendering of complex optical phenomena (e.g., depth-of-field or motion blur) by distributing rays according to the underlying analytic function that is being sampled. However, tracing hundreds or thou-

sands of ray samples per pixel is still needed to achieve converged rendering results, leading to large rendering times (e.g., hours) which are often not acceptable for practical purposes. When a relatively small number of samples per pixel (spp) is used, e.g., 8 - 32 spp, the rendered images generally suffer from MC error (i.e., variance), which can be considered one of the main problems of MC ray tracing techniques.

Image filtering methods [McC99] have been widely applied for improving the performance of MC ray tracing, thanks to their main benefits such as inherent simplicity and generality. The high-level behavior consists of taking the rendered image generated with a small number of samples as a noisy input, and producing a filtered image instead of allocating additional samples. Recent filtering methods [ZJL*15] demonstrated that the required number of samples for achieving high-quality rendered images can be drastically reduced by applying sophisticated image filters such as cross-bilateral filter [LWC12], non-local means [RKZ12], and weighted local regression [MCY14].

The main challenge of image filtering for MC ray tracing is in the fundamental difficulty to discern high frequency noise from MC features (e.g., noisy textures). The state-of-the-art filtering methods [LWC12, RMZ13, MCY14, MIGYM15, KBS15] are able to tackle this challenge and produce high-quality filtering results by utilizing additional rendering-specific features. Typical features that these methods employ are geometric features such as normal, texture, and depth, which can be obtained easily during the rendering process. The features boost the filtering quality by identifying the high-frequency edges introduced by discontinuities in G-buffers, but robustly utilizing the features in filtering frameworks can be challenging since these features may themselves contain noise due to distributed effects as shown in Fig. 1.

Previous approaches dealing with noisy features utilize the variances of the geometric features during the filtering process [LWC12] or use an additional pre-filtering process [RMZ13, MCY14, MIGYM15, KBS15], since the noisy features caused by depth-of-field or motion blur often have high variances. The fundamental drawback of these approaches is that the feature variances can be also high where high-frequency edges, which should be preserved properly, exist in focused areas (e.g., noisy textures due to a bump mapping), and thus this often results in under- or over-blurred results.

To alleviate this problem, we propose a novel pre-filtering method for reducing MC noise contained in G-buffers, in order to boost the performance of existing filtering methods for MC ray tracing through a seamless integration with the existing methods. Specifically, we present the following technical contributions:

- We propose a new pre-filtering approach using world positions and their variances to effectively reduce noise in G-buffers.
- Our method employs a per-pixel covariance matrix of world position samples so that detailed features introduced by motion blurring are properly preserved by performing an anisotropic filtering along the major direction of motions per pixel.
- We employ the Stein's unbiased risk estimator to locally estimate the optimal bandwidth for our pre-filtering, which minimizes our pre-filtering error.

We have integrated our approach into state-of-the-art off-line [LWC12, MCY14] and interactive filtering methods [DSHL10, BEM11], and demonstrated that our method improves the filtering quality of previous approaches, by up to a factor of 2×.

2. Related Work

In this section, we mainly review previous filtering methods that employ G-buffer information as filtering features since we aim to improve existing approaches through a new pre-filtering process on these input sources. We refer to a recent survey [ZJL*15] that includes a comprehensive overview of MC noise reduction.

Off-line filtering methods. Utilizing rendering-specific features as edge-stopping functions has a long history. McCool [McC99] presented an anisotropic diffusion process to reduce MC noise and employed geometric features and their variances to avoid excessive over-blurring during the iterative process. Recently, controlling filtering parameters in a data-driven way has been actively studied and demonstrated as an effective route for improving the filtering quality. For example, Sen and Darabi [SD12] estimated optimal parameters for a cross-bilateral filter using mutual information.

More recently, estimating the optimal parameters locally based on error analysis [LWC12, RMZ13, MCY14, MIGYM15] has become a popular approach as it can produce numerically and visually superior results, compared to manually selecting the parameters globally. For example, Li et al. [LWC12] introduced a general error estimator, Stein's unbiased risk estimator [Ste81], to estimate optimal filtering parameters at each pixel. Rousselle et al. [RMZ13] also employed the general estimator for optimizing the cross non-local means filter locally. Moon et al. [MCY14] presented a weighted local regression that performs filtering using features and adjusted parameters for each feature locally so that filtering errors can be minimized. In a similar line, Moon et al. [MIGYM15] proposed an optimization technique for the local regression, which performs reconstructions at only a sparse number of pixels, in order to drastically reduce the filtering overhead. Kalantari et al. [KBS15] demonstrated that optimizing parameters using a machine learning technique can outperform other methods that directly estimate errors

(e.g., mean squared errors) for scenarios where estimating errors is very challenging given a small number of samples (e.g., 8 spp).

Recent methods have demonstrated that high-quality filtering outputs can be produced by fully utilizing geometries through bandwidth optimization, but the performance improvement often decreases when such features contain severe noise introduced by depth-of-field and motion blur effects. Previous methods [LWC12, RMZ13, MCY14, MIGYM15, KBS15] utilized the variances of G-buffers to mitigate the problem. For example, Li et al. [LWC12] normalized the distance of two features by their variances when computing filtering weights of neighboring pixels, as variances of features tend to be high when affected by distributed effects. Rousselle et al. [RMZ13] utilized a non-local mean filter to decrease noise in feature buffers and adjusted filtering parameters using variances of feature buffers. In addition, when low-discrepancy sampling is used, feature variances were estimated using dual buffers where each buffer had half the number of samples to prevent variance overestimation. The local regression approaches [MCY14, MIGYM15] applied a truncated SVD in feature buffers and the truncation level was determined by feature variance. These approaches were able to produce smooth results on problematic regions, e.g., defocused areas, but they can lead to overblurred results on focused areas as such regions can have high variances due to complex shading (e.g., noisy textures). In addition, they did not optimize bandwidths for filtering the features, which can be crucial for achieving high quality results. To reduce noise in G-buffers while preserving high-frequency details, we present a new data-driven pre-filtering process that can be integrated with existing filtering methods to improve their output quality while maintaining a low computational overhead. Specifically, we provide pre-filtered G-buffers and their variances into existing filters so that these methods can produce higher quality results using our technique.

Interactive filtering methods. Interactive filters [DSHL10, BEM11] mainly aim to provide an interactive preview of rendered images. Dammertz et al. [DSHL10] proposed a variant of the A-Trous filter which utilizes geometric features as edge-stopping functions to preserve the edges introduced by geometric discontinuities. Bauszat et al. [BEM11] applied a guided filter that utilizes the correlation between colors and geometric features to preserve geometric edges while removing MC noise. The main assumption underlying existing interactive filters is that the set of rendering features are noise-free and therefore these methods do not possess processes to filter these features in a principled manner. In consequence, these methods might not be able to perform well on scenarios where noise in the G-buffers is present due to defocusing and motion blur effects. The contributions introduced in this work can be applied as part of a pre-filtering technique to complement these methods, enabling them to produce high-quality previews even for scenarios that break the noise-free assumption.

Frequency analysis based reconstruction. Anisotropic filtering based on frequency analysis [DHS*05] has demonstrated high-quality reconstruction results for distributed effects such as depth-of-field [SSD*09], motion blur [ETH*09], as a complementary line of research. Recent methods (e.g., [MVH*14]) even showed that these reconstructions can be performed in real-time. While the image space methods are simple and generalizable, thanks to the intrinsic nature of working in image space, the frequency based reconstruction is able to produce high-quality reconstruction results, even with low sample counts, by utilizing per-sample information such as sampling coordinates on images, lens, and time. However, the latter frequency based approaches often demonstrated that they reduce particular types of noise, introduced by distributed rendering effects. We direct the reader to a recent survey [ZJL*15] for comprehensive analysis.

Our method also deals with noise caused by distributed effects, but our technique mainly aims at reducing noise contained in G-buffers. Specifically, our approach is designed for improving existing image space filters by passing improved G-buffers, instead of directly reducing noise in the output images. One may apply recent anisotropic filters using frequency analysis in noisy G-buffers, but it is non-trivial to efficiently combine these methods with the existing image space filtering. This is mainly because these approaches typically store individual light field samples but recent image space techniques run in a discretized space (i.e., pixel space). Alternatively, our method utilizes per-pixel world positions, and their distribution, to integrate our pre-filtering into existing image-space approaches conveniently.

3. Overview

Our solution for existing filtering methods is designed on top of the observation that state-of-the-art filters [LWC12, RMZ13, MCY14, MIGYM15, KBS15] generally use G-buffers as edge-stopping functions to preserve high-frequency edges in the input image. Unfortunately, the buffers can contain noise due to distributed effects (e.g., motion blur and depth-of-field effects), and can result in noisy outputs as the noise in the G-buffers can be considered as high-frequency edges. To solve this problem, we present a new pre-filtering process for reducing the noise in the G-buffers, as reducing the noise contained in these auxiliary buffers can effectively improve the quality of the final rendered images.

The method proposed in this paper is devised to be easily coupled with existing filtering methods of MC ray tracers (See Fig. 2). Our proposed method only requires a simple pre-processing of these auxiliary buffers in order to boost the

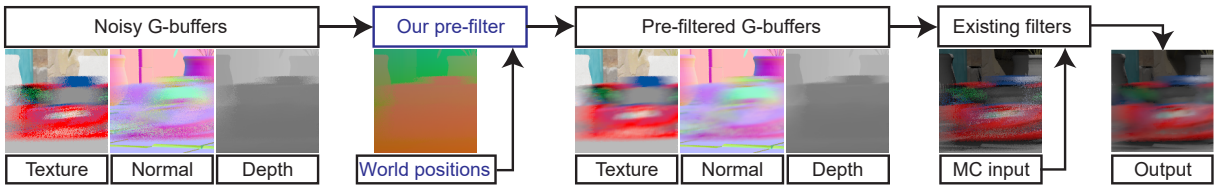

Figure 2: *Our pre-filtering framework. Our method reduces noise contained in G-buffers such as normal, texture, and depth buffers, by applying a new anisotropic filtering based on world positions. Our pre-filtered G-buffers are used as edge-stopping functions of existing filtering methods, which remove MC noise in the input image.*

performance of the existing filters. Once processed, the filtered G-buffers can be used as inputs of existing filters for improving their filtering quality.

4. Pre-filtering of G-buffer Features

In this section, we formalize the problem of reducing MC noise (i.e., variance) in G-buffers (Sec. 4.1). Next, we propose a world positions based pre-filtering (Sec. 4.2), followed by an anisotropic filtering guided by the Mahalanobis distance (Sec. 4.3) and the respective bandwidth optimization (Sec. 4.4).

4.1. Problem definition

Improving the quality of existing filtering methods that use geometric features can be tackled by reducing the noise already present on these features which later feed the filtering algorithms. We formulate the problem as a pre-filtering process as follows:

$$\hat{\mathbf{g}}_c(k) = \frac{1}{W} \sum_{i \in \Omega_c} w_i(k) \tilde{\mathbf{g}}_i(k), \tag{1}$$

where $\hat{\mathbf{g}}_c(k)$ is the filtered feature at center pixel c in the k-th feature buffer, and $w_i(k)$ is a filtering weight allocated to a noisy feature $\tilde{\mathbf{g}}_i(k)$ stored at the i-th neighboring pixel. The neighboring pixels i are selected as the pixels within a regular filtering window Ω_c (e.g., 7×7 window) centered at c and the normalization term W is set as $W = \sum_{i \in \Omega_c} w_i(k)$.

In rendering, the feature images $\tilde{\mathbf{g}}(k)$ are commonly computed by averaging geometries such as normal, texture, and depth samples at each pixel. Unfortunately these G-buffers, which serve as edge-stopping functions of existing filtering methods, can contain a significant amount of noise due to a variety of distributed effects and it can lead to the sub-optimal results of previous filtering methods. To tackle this problem, we apply a pre-filtering (Eq. 1) to pre-filter the input features $\tilde{\mathbf{g}}(k)$. This allows the estimation of the unknown features $\mathbf{g}(k)$ that can only be computed exactly with an infinite number of samples. Technically, our pre-filtering tries to minimize a filtering error, $|\hat{\mathbf{g}}_c(k) - \mathbf{g}_c(k)|^2$ by locally controlling the filtering weight, $w_i(k)$.

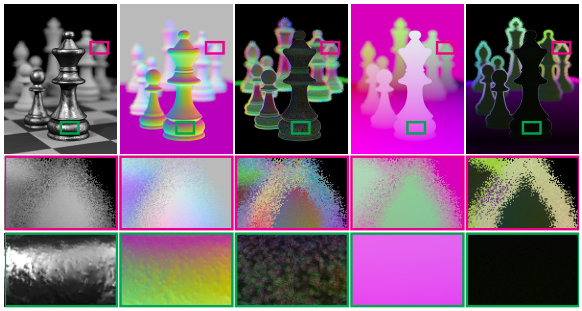

Figure 3: *Variances of normal and world positions for the Chess scene. Given the input image (a), we visualize the normals (b) and their variances (c) that may be utilized for estimating errors in the normal buffer. Unfortunately, the variances in the focused area (bottom row) can be high due to the bump mapping, but these details need to be preserved since the details are not actual errors but a shading detail. As a result, estimating the amount of noise in G-buffers based on the variances can introduce over-blurring for the detailed edges. We mitigate this problem, by utilizing world positions (d) and their variances (e), resulting in high variances only on the defocused area (middle row).*

4.2. Pre-filtering using world positions

Our key idea for performing pre-filtering is to utilize world positions, i.e., the intersection points between scenes and primary rays. Let us define the j-th intersection point between scenes and the j-th ray at a pixel i as $\mathbf{s}'_{i,j}$, and our world position sample $\mathbf{s}_{i,j}$ is computed using the intersection point:

$$\begin{pmatrix} \mathbf{s}_{i,j} \\ 1 \end{pmatrix} = M^{-1} \begin{pmatrix} \mathbf{s}'_{i,j} \\ 1 \end{pmatrix}, \tag{2}$$

where M is the 4×4 transformation matrix, which can include per-sample motion such as rotation and translation.

When a sample does not include a motion, the matrix becomes the identity matrix. In this case, our world position sample $\mathbf{s}_{i,j}$ is equivalent to the intersection point $\mathbf{s}'_{i,j}$.

The main reason for using the modified intersection point $\mathbf{s}_{i,j}$ is that it allows the computation of a per-pixel covariance matrix that estimates a per-pixel motion introduced by motion blurring. For example, suppose a rotating sphere without translations. When we compute the intersection points between the sphere and the rays within a pixel, the points would be very similar without regards to the rotation speed. On the other hand, our modified world position samples can have a different distribution with regard to a motion. Our modification can provide a high-quality filtering result for motion blurring effects, since it can provide a crucial hint on the anisotropic weighting that considers the distribution caused by a local motion. The details of how we utilize the distribution are in the subsequent section (Sec. 4.3).

After computing each world position sample, we simply average the samples within a pixel in order to perform our pixel based pre-filtering, i.e., $\tilde{\mathbf{p}}_i = 1/n_i \sum_{j=1}^{n_i} \mathbf{s}_{i,j}$, where n_i is the number of world position samples at pixel i. Given the world position $\tilde{\mathbf{p}}_i$, we define the filtering weight $w_i(k)$ at a neighboring pixel i for the k-th feature as the following:

$$w_i(k) \equiv w(\tilde{\mathbf{p}}_i, \tilde{\mathbf{p}}_c) = exp\left(-\frac{d(\tilde{\mathbf{p}}_i, \tilde{\mathbf{p}}_c)}{2h^2}\right), \qquad (3)$$

where $d(\tilde{\mathbf{p}}_i, \tilde{\mathbf{p}}_c)$ is a distance function that computes a similarity between two world positions stored in pixel i and center c. Note that the filtering weight function $w_i(k)$ is independent from the type of feature, k, instead of adjusting the weight for each feature type. We determine how to compute the distance function $d(\tilde{\mathbf{p}}_i, \tilde{\mathbf{p}}_c)$ and the filtering bandwidth term h in the subsequent sections.

The key intuition behind the world positions based weighting is that we can robustly identify the problematic regions, e.g., noisy geometries in defocused areas, by estimating the variances of the worlds positions. For example, the variances are typically low in the focused areas since the detailed edges, e.g., noisy normals, are introduced by a shading process, as shown in Fig. 3. In addition, there can be high correlation between the world positions and other feature buffers (e.g., texture, normal, and depth), as the buffers are typically sampled at intersection points (i.e., world positions) between rays and scenes. Our approach utilizes this correlation through our world position based weighting, and filters the feature buffers with equivalent amounts of smoothing, while identifying the problematic areas.

As an alternative, one may estimate noise in G-buffers by utilizing the variances of the buffers, and then apply a pre-filtering on each buffer based on the estimated variances since the variances tend to be high for problematic regions such as defocused areas. These approaches, however, can fail to preserve very detailed edges in focused areas, as those areas typically have high variances. For example, normals in focused areas (Fig. 3) have detailed edges and corresponding high variances due to the bump mapping, but these edges should be preserved as they are not MC noise, despite exhibiting high variances. As a result, it can be fundamentally difficult to estimate the optimal pre-filtering weight, based on the feature variances.

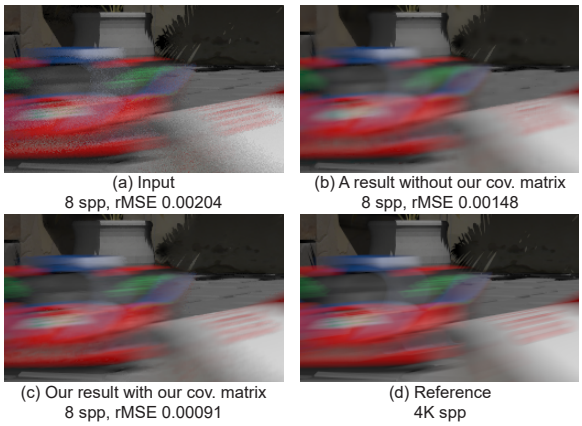

Figure 4: *Results with and without our per-pixel covariance matrix. Our method performs an anisotropic pre-filtering by utilizing the matrix, which is computed by world position samples per pixel. Our result (c) shows numerically and visually better results over the alternative (b) that does not utilize the covariance matrix. To compute these results, we apply the pre-filtering approaches to the state-of-the art filtering method, WLR.*

4.3. Mahalanobis distance on world positions

To compute the weighting function (Eq. 3), we need to define a distance metric that measures a similarity between two world positions. The world positions can have irregular density, and thus its ranges can vary locally, e.g., 1 to 1000. It requires a normalization process on the world positions so that pre-filtering can be performed properly. In addition, the distance metric needs to consider the dominant direction of motions at each pixel, in order to perform an anisotropic pre-filtering along the direction. To this end, we employ the Mahalanobis distance with a per-pixel covariance matrix S_c of world position samples:

$$\begin{aligned} d(\tilde{\mathbf{p}}_i, \tilde{\mathbf{p}}_c) &= \Delta(\tilde{\mathbf{p}}_i, \tilde{\mathbf{p}}_c)^T S_c^{-1} \Delta(\tilde{\mathbf{p}}_i, \tilde{\mathbf{p}}_c) + \Delta(\tilde{\mathbf{p}}_i, \tilde{\mathbf{p}}_c)^T S_i^{-1} \Delta(\tilde{\mathbf{p}}_i, \tilde{\mathbf{p}}_c) \\ &= \Delta(\tilde{\mathbf{p}}_i, \tilde{\mathbf{p}}_c)^T (S_c^{-1} + S_i^{-1}) \Delta(\tilde{\mathbf{p}}_i, \tilde{\mathbf{p}}_c), \end{aligned} \qquad (4)$$

where $\Delta(\tilde{\mathbf{p}}_i, \tilde{\mathbf{p}}_c) \equiv \tilde{\mathbf{p}}_i - \tilde{\mathbf{p}}_c$. We compute the 3×3 covariance matrices S_c and S_i at each pixel c and i using the world position samples $\mathbf{s}_{c,j}$ and $\mathbf{s}_{i,j}$ (Eq. 2). We include how to compute the covariance matrix in Appendix B.

In Fig. 4, we compare our approach with an alternative that does not utilize the per-pixel covariance matrix. For the al-

ternative method, we set the covariance matrix as the identity matrix. As shown in this figure, our result preserves the high-frequency edges better and effectively removes the noise introduced by the motion blur compared to the tested alternative, since the covariance matrix allows our method to perform an anisotropic pre-filtering along the major direction of motions per pixel.

Defining an appropriate distance between two measurements plays a key role in many computer vision and machine learning algorithms, and extensive studies are available [YJ06]. The most common yet simple metric is the Euclidean distance and this corresponds to setting the covariance matrix $(S_c^{-1} + S_i^{-1})$ to the identity. The main advantage of using the Mahalanobis distance, compared to the simple Euclidean metric, is that we can naturally account for correlation or covariance between measurements [XNZ08]. In our case, motion blur can introduce a shared distribution of features that requires a reconstruction along the major motion direction. Our approach controls the shape of our filter based on the per-pixel distribution of world positions, and estimates its optimal size to minimize our filtering error (in the next section).

4.4. Optimization of bandwidth

Given the Mahalanobis distance metric (Eq. 4), we estimate an optimal value for the bandwidth term h such that the error $|\hat{\mathbf{g}}_c(k) - \mathbf{g}_c(k)|^2$ is minimized per each center pixel c. This optimization can be fundamentally challenging as discussed, and thus we instead try to minimize an additional error term $||\hat{\mathbf{p}}_c - \mathbf{p}_c||^2$ which is related to the following equation:

$$\hat{\mathbf{p}}_c = \frac{1}{W} \sum_{i \in \Omega_c} w(\tilde{\mathbf{p}}_i, \tilde{\mathbf{p}}_c) \tilde{\mathbf{p}}_i. \quad (5)$$

Intuitively speaking, we estimate an optimal weight $w(\tilde{\mathbf{p}}_i, \tilde{\mathbf{p}}_c)$ at each pixel so that our filtered world position $\hat{\mathbf{p}}_c$ is closely matched with the unknown world position $\mathbf{p}_c$ that would be computed with an infinite number of samples. Then, we pre-filter all other available geometries using the weighting function, defined by the bandwidth found by simultaneously optimizing world position information.

To estimate the optimal bandwidth, we employ Stein's unbiased risk estimator [Ste81, BL07] that estimates the expected mean squared error term $E||\hat{\mathbf{p}}_c - \mathbf{p}_c||^2$. To apply the general estimator, we assume a statistical model, $\tilde{\mathbf{p}}_c = \mathbf{p}_c + \varepsilon_c I$, where the noise term ε_c follows the normal distribution, $\varepsilon_c \sim N\left(0, \sigma^2(\tilde{\mathbf{p}}_c)/n_c\right)$ and I is the 3×3 identity matrix. The $\sigma^2(\tilde{\mathbf{p}}_c)$ is the sample variance of the world position and n_c is the sample count at the pixel c, respectively. Given the normality assumption, we can compute the unbiased estimation for our pre-filtering error as the following:

$$SURE = \frac{1}{D}||\hat{\mathbf{p}}_c - \tilde{\mathbf{p}}_c||^2 - \frac{\sigma^2(\tilde{\mathbf{p}}_c)}{n_c} + \frac{2\sigma^2(\tilde{\mathbf{p}}_c)}{n_c D} div(\hat{\mathbf{p}}_c), \quad (6)$$

where D is the dimension of the input vector $\tilde{\mathbf{p}}_c$, i.e., $D = 3$, and $div(\hat{\mathbf{p}}_c)$ is the divergence term of the filtered value. We include the computation of the divergence term in our appendix A.

Given a user-defined set of bandwidths, we run our pre-filtering (Eq. 5) with bandwidth values selected from this set and estimate the corresponding error (Eq. 6). Then, we select the bandwidth which has a minimal estimated error as the estimated optimal bandwidth $\hat{h}_{opt}$. Note that the estimation (Eq. 6) can provide negative values, since SURE is an unbiased estimator of the mean squared error and the estimator has its own variance, as discussed in a recent paper [LWC12]. We select a bandwidth with a minimal error as optimal, even if the estimated error is negative.

One may employ this general estimator to apply a pre-filtering for each G-buffer, instead of our world-position based pre-filtering. In Fig. 5, we compare this alternative with our method. The alternative approach directly applies a pre-filtering on the normal buffer and the optimal bandwidths are estimated by the error estimator that we employ for our method. As shown in Fig. 5, this approach fails to preserve the detailed edges generated by a bump mapping, since those areas have high variances in the normal buffer. This indicates that directly feeding the variances into the SURE based bandwidth selection can lead to sub-optimal bandwidths in focused areas, e.g., (c) in Fig. 5, as the SURE (Eq. 6) typically predicts lower errors from large bandwidths than those predicted from smaller bandwidths, due to the high variances. Our method, however, mitigates this problem by utilizing world positions and their variances.

5. Implementation Details

We have implemented our pre-filtering using CUDA, and applied it to reduce noise in the G-buffers before conducting filtering methods for Monte Carlo ray tracing. We have used a 7×7 filtering window Ω_c and estimated an optimal bandwidth h_{opt} using a user-defined set $(0.0, 0.4, 0.8, 1.2, 1.6, 2.0)$. In practice, the smallest value 0.0 can be considered as an exception in order to make our pre-filtering consistent. In this case, our pre-filtering produces input values as outputs without performing any smoothing and our pre-filtering error is equal to the input variance, i.e., $\sigma^2(\tilde{\mathbf{p}}_c)/n_c$. We have found that the estimated optimal bandwidths using Stein's unbiased risk estimator can be very noisy, as reported in the previous paper [LWC12]. As suggested in the previous work, we use an additional smoothing that filters the estimated noisy bandwidths. Specifically, we

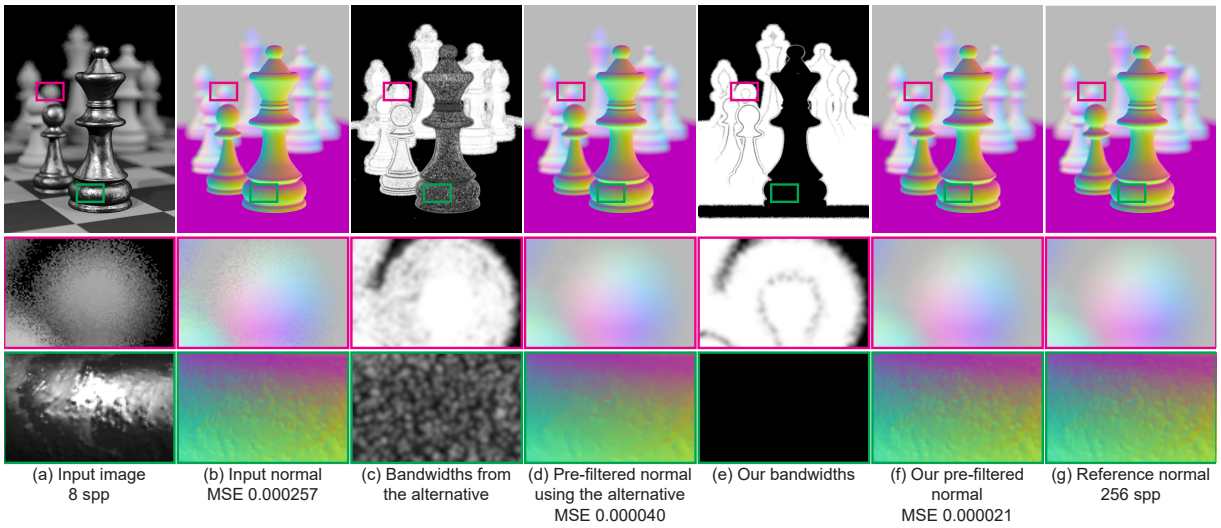

Figure 5: *Visualizations for estimated bandwidths. MC ray tracing produces the input image (a) and normal buffer (b), and the normal buffer contains noise due to the depth-of-field effect. As an alternative of our pre-filtering, we test a pre-filtering based on the variances of the normals. The alternative shows a smooth result on the defocused area (second row), but removes high-frequency details in the focused region (bottom row) due to the large bandwidths caused by the high variances of the normals. For our visualization purpose, the bandwidth values are mapped into a normalized range, 0 (smallest) to 1 (largest one). Our method using world positions, however, preserves high frequency details thanks to the small bandwidths, while removing noise in the defocused areas. The mean squared error (MSE) is computed using the reference normal buffer computed with 256 spp.*

applied a simple Gaussian filtering with a bandwidth 1.5 so that estimated optimal bandwidths can be smoothed. Analogously, we have also applied the same Gaussian filtering to the per-pixel covariance matrix S_c since the elements of the matrix can be noisy as well especially when a small number of samples (e.g., 4) is used.

Variances of pre-filtered features. We provide smoothed geometric features to existing filtering methods instead of noisy features, and thus we need to compute the variances of the pre-filtered features since the existing filtering methods can utilize the variances in addition to the features. For example, Li et al. [LWC12] employed the feature variances to compute the filtering weight between two pixels in their cross-bilateral filter and Moon et al. [MCY14] applied a truncated singular value decomposition using the variances. For a seamless integration of our method into the existing method, we compute the filtered variance which can be normally smaller than the input variance. We employ the variance estimation technique using dual buffers, recently proposed in [RMZ13]. Specifically, we split world position samples into two buffers, and apply our pre-filtering to noisy G-buffers using each world position buffer, respectively. The output variances are simply estimated as squared differences of each pre-filtered G-buffer. In addition, we apply a Gaussian filter with a small bandwidth (e.g., a pixel width) to each world position buffer before performing our pre-filtering.

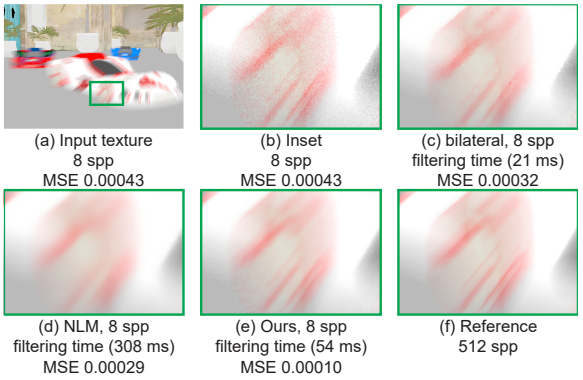

Figure 6: *Pre-filtering results for the noisy texture buffer, i.e., (a) and (b). The bilateral filter (c) leaves some high-frequency noise on the motion blurred area and NLM (d) produces over-blurred results. Our method (e) generates better results than the previous approaches, since our technique performs an anisotropic filtering along the motion direction guided by distributions of world positions.*

We calculate the variances when computing the pre-filtered features, and pass those maps into existing filtering methods. The main practical benefit of our pre-filtering is that we do not need to modify the existing methods thanks to our seamless integration.

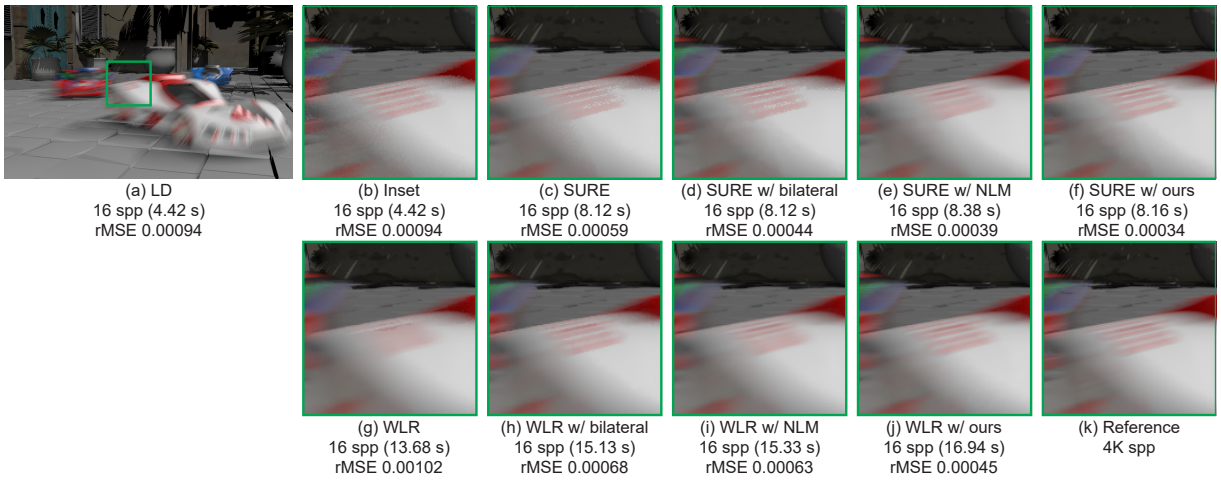

Figure 7: *Results for the Cars scene (each car has a different motion). SURE and WLR produces under- and over-blurred result, respectively. The previous methods exploiting our pre-filtered G-buffers show improved results.*

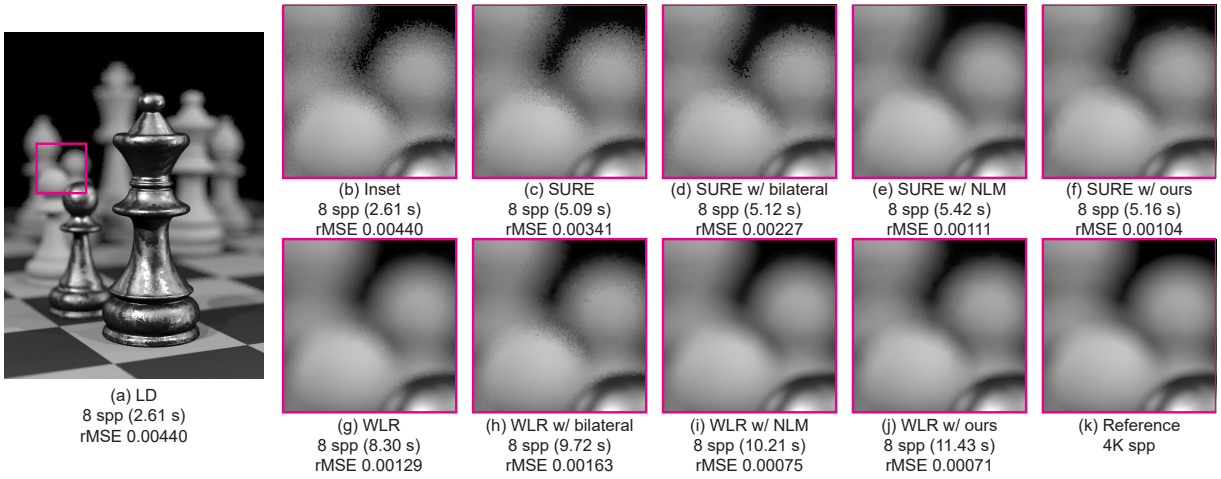

Figure 8: *Results for the Chess scene. SURE (c) and WLR (g) produce noisy results on the defocused area due to noisy features such as texture, normal, and depth. Our pre-filtering on the noisy features improves the filtering quality of the previous methods in terms of numerical accuracy and visual quality.*

6. Results and Discussions

We have demonstrated our pre-filtering with recent off-line filtering methods such as Stein's unbiased risk estimator (SURE) [LWC12] and weighted local regression based adaptive rendering (WLR) [MCY14]. We have utilized the GPU implementation provided by the authors for WLR and converted their CPU implementation of SURE into a CUDA implementation. For the selected off-line filtering methods, we have tested different pre-filtering methods such as a bilateral filter, non-local means (NLM) [RMZ13], and our method.

In addition, we have also verified our method with recent interactive filtering approaches such as A-Trous [DSHL10] and guided filters [BEM11], which utilize normal, texture, and depth as edge-stopping functions. Furthermore, we have also tested the low discrepancy (LD) sampling that uses a uniform sampling and pixel reconstruction filter (e.g., box filter), without applying a post filtering process. We have only used direct illumination, in order to clearly show filtering results affected by tested pre-filtering approaches. For our tests, we have used a Windows machine with an Intel Xeon CPU E5 3.00 GHz CPU and NVidia GTX 1080 graphics hardware. As a numerical measure, we use the relative

Figure 9: *Results for the San Miguel scene where the depth-of-field effect is simulated. In the defocused area, the previous off-line methods (SURE and WLR) leave some noise due to noisy G-buffers, but the existing methods combined with our pre-filtering produce numerically and visually better results.*

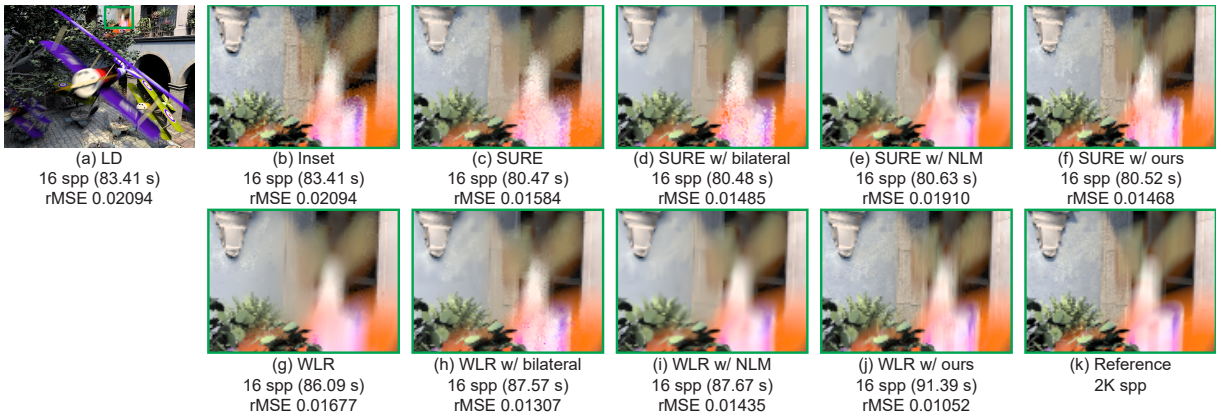

Figure 10: *Results for the Planes scene. The zoom-in region shows a challenging scenario where motion exists behind the complex geometries. Our method improves the results of previous methods (SURE and WLR) by pre-filtering the motion blurred area while preserving the geometries in the focused region.*

mean squared error (rMSE) [RKZ11] that was commonly adopted in the state-of-the-art filtering approaches.

Benchmarks. We test our pre-filtering for the rendering scenes, 1) Cars (1000×600), 2) Chess (750×1000), 3) San Miguel (1024×1024), 4) Planes (800×600), and 5) Furry Bunny (1024×1024), where the numbers in the parentheses are tested image resolutions. For the Cars scene (Fig. 1, 7), we simulate different motions for each toy car, which introduces different amounts of noise in feature buffers. This scenario introduces a challenge for our method as our pre-filtering kernel shape should be an anisotropic one along each motion direction so that high-frequency edges in G-buffers are properly preserved. In the Chess scene (Fig. 8), a strong depth-of-field effect is simulated where the features become noisy on the defocused regions. The chess object in the focused region has noisy normals due to bumped mapping, and thus a pre-filtering needs to distinguish the noisy features from noise in the defocused areas. Given the San Miguel scene (Fig. 9, 13), we test a depth-of-field effect on complex geometries, which introduces severe noise in the feature buffer. For the Planes scene (Fig. 10), we add toy planes that have different motions into the San Miguel scene. It provides a challenging scenario for our method since some motion blurred areas are mixed with static but complex geometries, i.e., tree leaves. Given the Furry Bunny

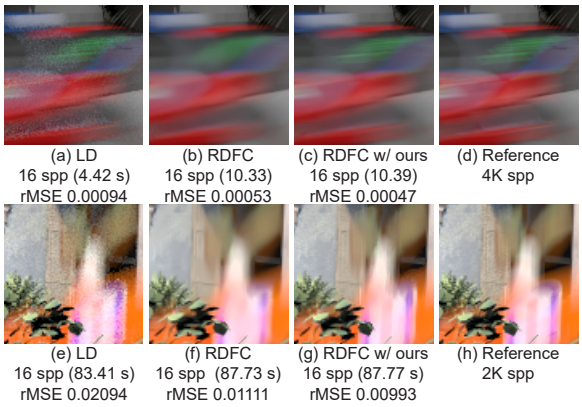

Figure 11: *Results for the Cars and Planes scene. A recent filtering method, RDFC, produces improved results when our pre-filtering is combined.*

scene (Fig. 14), we also test a very challenging scenario where each model with fur has different motion. This scene provides a failure case of our pre-filtering, since this test makes our per-pixel statistics very noisy especially when we use small sample counts.

Comparisons with other pre-filtering. We compare our pre-filtering with other methods such as a simple bilateral and an optimized non-local means (NLM) filter [RMZ13]. For testing the previous methods, we apply these filters to each feature buffer separately as performed in the previous work [RMZ13]. We select the parameters of the bilateral filter so that the numerical errors of the filter can be minimized. Note that this parameter selection cannot be performed in practice, as it is not possible to know the ground truth image. For the NLM filter, we split feature samples into two buffers (i.e., dual buffers), and then estimate variances of features using the squared differences between the two buffers, as described in [RMZ13]. In addition, we utilize the implementation provided by the authors. As shown in Fig. 6, the simple bilateral leaves some noise in the motion blurred area. One may increase the (range) parameter to further reduce the noise, but it can lead to over-blurring in other areas (e.g., focused area). The NLM filter produces smooth results, but tends to fail to capture fine details caused by motion blur. In comparison, our method preserves the details well as it performs a more robust filtering based on world positions and their distributions. Also, our filtering time (e.g., 54 ms) is much lower than that of the relatively expensive patch-based method (e.g., 308 ms).

Pre-filtering for off-line filtering methods. Given the Cars scene (Fig. 7), the method based on the local regression (i.e., WLR) shows slightly lower quality than SURE in terms of the numerical accuracy, since the correlation between features and colors in the scene is not high. For other scenes (Fig. 8, 9 and 10), WLR shows better results than SURE as the local regression utilizes the existing correlation using the linear models. These previous techniques generate improved results compared to LD, which does not utilize any post filtering approaches. However these previous methods show under- or over-blurred results in the motion blurred areas (in Fig. 7 and 10) and defocused regions (in Fig. 8 and 9). This is mainly because the G-buffers that these methods employ are noisy.

Tested pre-filtering approaches (bilateral, NLM, and our method) alleviate this problem by passing filtered G-buffers and their variances into the off-line techniques. While the filtering results combined with a simple bilateral filter still show noisy results, NLM and our method enable the off-line filters to generate smooth results on the defocused or motion blurred areas. Specifically, our pre-filtering and NLM show similar results for the depth-of-field scenarios (Fig. 8 and 9), but our method outperforms NLM given the motion blur benchmarks (Fig. 7 and 10) as our pre-filtering guided by world positions (and their covariances) leads to a sheared reconstruction along object motions. In addition, NLM does not improve the result of SURE, since NLM removes high-frequency information in the focused area for the Planes scene (e.g., tree leaves in Fig. 10). Our pre-filtering reduces these over-blurring artifacts by utilizing the variances of world positions, instead of the variances of each feature.

We also tested our pre-filtering with the robust denoising using feature and color (RDFC) [RMZ13], which has a sophisticated pre-filter using NLM. Our approach is designed only for removing noise in G-buffers, and thus our pre-filtering does not reduce noise in other features (e.g., visibility) of the RDFC. Specifically, our filter takes noisy G-buffers as inputs and passes filtered buffers (and variances) to RDFC, which performs its own pre-filtering (i.e., NLM) and main filtering. As shown in Fig. 11, our technique improves the results of RDFC both numerically and visually.

Numerical Convergence. We have evaluated rMSE numbers for our tested scenes in Table 1 (also shown in Fig. 12), when different numbers of samples are used. The existing off-line methods aim to reduce MC noise in rendered images generated by not only small numbers of samples (e.g., 8 to 16) but also relatively large numbers of samples (e.g., 32 to 64), and thus a pre-filtering should improve the filtering quality of existing approaches for a different number of samples. As shown in the table and figure, our pre-filtering consistently reduces the numerical errors for WLR and SURE. It indicates that our method is able to improve the previous methods when a different amount of noise exists in G-buffers thanks to our high-quality pre-filtering. In addition, we improve the filtering quality of the state-of-the-art methods, by up to a factor of 2×. For example, the rMSE of SURE with our method (0.00104 with 8 spp) is lower than the error of SURE without our pre-filtering (0.00140 with 16 spp) for the Chess scene. Also, the error of WLR with our approach

(0.00026 with 32 spp) is lower than that of WLR without our pre-filter (0.00037 with 64 spp) for the Cars scene.

Pre-filtering for interactive filters. Given the San Miguel scene (Fig. 13), we have tested our pre-filtering for interactive filters, A-Trous and guided filters. The filters tend to produce very noisy results on the defocused area as these filters do not have a pre-filtering process to address noise in G-buffers. Our method, however, significantly boosts the filters by passing pre-filtered G-buffers into the inputs of the filters, and the methods with our pre-filtering show smoothed results on the region. In addition, the numerical errors are reduced from 0.02289 and 0.02313 to 0.01200 and 0.01675 for the A-trous and guided filters, respectively.

Computational overhead. The computation times of our pre-filtering are 53 ms, 66 ms, 96 ms, and 47 ms for the Cars, Chess, San Miguel, and Planes scenes, respectively. The filtering times of the off-line methods are 0.93 s and 4.58 s (Cars), 1.14 s and 3.67 s (Chess), 1.49 s and 7.78 s (San Miguel), and 0.74 s and 5.03 s (Planes scenes) for SURE and WLR, respectively. As a result, the percentages of our computational overheads over the existing off-line filters are 6.1 % and 1.3 % for SURE and WLR on average. For the interactive filters, the percentages are increased to 33 % and 21 % for the A-Trous and guided filter as these filters are much faster than the off-line filters. Our method introduces a relatively non-negligible overhead for the interactive filters, but the results of the interactive filters are significantly improved since these filters without our pre-filtering cannot address noise in G-buffers.

We have observed that the filtering times for WLR increase when our pre-filtering is applied, since the computational time of WLR is dependent from the variance of G-buffers. Especially, WLR estimates the dimensionality of local features using the variance, and the computational overhead increases as the dimension increases. As our method lowers the variances, the dimension tends to be increased. For example, the filtering time for the Cars scene is increased from 4.58 s (without our method) to 7.84 s (with ours). For other filters such as SURE and interactive filters, the computational times are the same since the number of computations for these methods are independent of the variance of G-buffers. Overall, our computational overhead can be considered acceptably small if we consider the quality improvement of the previous methods.

Limitations and future work. Our method relies on the per-pixel covariance matrix to guide an anisotropic kernel shape with the Mahalanobis distance for motion blurring effects, but the shape can be sub-optimal when a non-linear motion is simulated since the shape is an ellipsoid, rather than an arbitrary shape. It would be interesting to approximate a non-linear motion as multiple linear motions and then apply a separate pre-filtering for each motion.

spp	8	16	32	64
Method	Cars			
SURE	0.00143	0.00059	0.00031	0.00017
SURE with ours	0.00073	0.00034	0.00019	0.00011
WLR	0.00184	0.00102	0.00061	0.00037
WLR with ours	0.00090	0.00045	0.00026	0.00017
Method	Chess			
SURE	0.00341	0.00140	0.00055	0.00022
SURE with ours	0.00104	0.00055	0.00024	0.00012
WLR	0.00129	0.00054	0.00028	0.00016
WLR with ours	0.00071	0.00033	0.00017	0.00010
Method	San Miguel			
SURE	0.05118	0.03608	0.01251	0.00608
SURE with ours	0.03088	0.02051	0.00833	0.00433
WLR	0.02097	0.01201	0.00703	0.00428
WLR with ours	0.01207	0.00743	0.00473	0.00305
Method	Planes			
SURE	0.02676	0.01584	0.00869	0.00487
SURE with ours	0.02646	0.01468	0.00772	0.00429
WLR	0.02994	0.01677	0.00934	0.00551
WLR with ours	0.01745	0.01052	0.00636	0.00394

Table 1: *rMSE comparisons for the off-line filters, SURE and WLR, given a different number of samples per pixel.*

Another limitation of our approach is that per-pixel statistics such as per-pixel covariance or variances of world positions can be noisy when a small number of samples is used. Fig. 14 illustrates this scenario where two furry objects have different motions. Our pre-filtering improves the result of the WLR even for this case, thanks to the improved feature buffers passed by our method. However, the previous method still generates over-blurred results, since our pre-filtering does not provide high-quality pre-filtered buffers due to noisy statistics caused by complex geometries and a low sample count. As discussed in the implementation section (Sec. 5), we apply a simple filtering to the covariance matrix and estimate optimal bandwidths to alleviate this problem. However, it would be ideal to design a robust estimation process for these per-pixel statistics without applying the additional smoothing. We leave these potential research direction to our future work.

7. Conclusions

In this paper, we have presented a pre-filtering technique that reduces noise while preserving detailed edges in G-buffers, which mainly aims to boost existing filtering techniques. As a key idea, we propose a world position (and its variance) based pre-filtering process which robustly identifies problematic regions caused by distributed effects such as depth-of-field and motion blur. Our pre-filtering improves the filtering quality of state-of-the-art filtering methods by generating filtered G-buffers in terms of numerical accuracy and

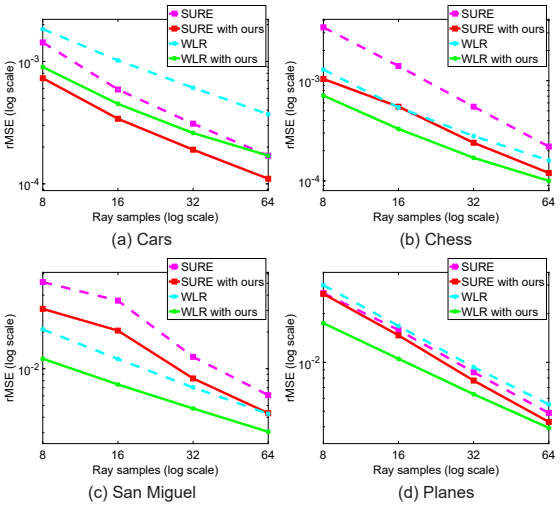

Figure 12: *rMSE convergence graphs. These graphs are generated using the rMSE numbers in Table 1.*

visual quality, and this improvement is achieved by seamless integration.

Acknowledgements

We appreciate the insightful feedback of the anonymous reviewers. The Cars, Chess, and San Miguel scenes are courtesy of Maggie Kosek, Wojciech Jarosz, and Guillermo M. Leal Llaguno. Also, the Plane and Bunny models are courtesy of Joanna Jamrozy and the Stanford Computer Graphics Laboratory. This work was funded by Innovate UK project #101858, and supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2017R1C1B2003893).

Appendix A: Computing the divergence.

We compute the divergence term $div(\hat{\mathbf{p}}_c) \equiv \sum_{k=1}^{3} \frac{\partial \hat{\mathbf{p}}_{c,k}}{\partial \tilde{\mathbf{p}}_{c,k}}$ in Eq. 6, where $\hat{\mathbf{p}}_{c,k}$ and $\tilde{\mathbf{p}}_{c,k}$ denote the k-th elements in the vectors $\hat{\mathbf{p}}_c$ and $\tilde{\mathbf{p}}_c$ respectively. Given our pre-filtering equation (Eq. 5), the partial derivative term can be computed by the quotient rule:

$$\begin{aligned}\frac{\partial \hat{\mathbf{p}}_{c,k}}{\partial \tilde{\mathbf{p}}_{c,k}} &= \frac{1}{W^2}\left\{\frac{\partial}{\partial \tilde{\mathbf{p}}_{c,k}}\left(\sum_{i\in\Omega_c} w_i \tilde{\mathbf{p}}_{i,k}\right) W - \frac{\partial W}{\partial \tilde{\mathbf{p}}_{c,k}}\left(\sum_{i\in\Omega_c} \tilde{\mathbf{p}}_{i,k}\right)\right\} \\ &= \frac{1}{W}\left(\sum_{i\neq c} \frac{\partial w_i}{\partial \tilde{\mathbf{p}}_{c,k}} \tilde{\mathbf{p}}_{i,k} + 1 - \hat{\mathbf{p}}_{c,k} \sum_{i\in\Omega_c} \frac{\partial w_i}{\partial \tilde{\mathbf{p}}_{c,k}}\right), \end{aligned} \tag{7}$$

where $w_i \equiv w(\tilde{\mathbf{p}}_i, \tilde{\mathbf{p}}_c)$. In the equation above, the derivative $\frac{\partial w_i}{\partial \tilde{\mathbf{p}}_{c,k}}$ of the weight function (Eq. 3) is computed as the following:

$$\frac{\partial w_i}{\partial \tilde{\mathbf{p}}_{c,k}} = \frac{1}{h^2}\left(S_{c,k}^{-1} + S_{i,k}^{-1}\right)\left(\tilde{\mathbf{p}}_i - \tilde{\mathbf{p}}_c\right) w_i, \tag{8}$$

where $S_{c,k}^{-1}$ and $S_{i,k}^{-1}$ are the k-th row vectors in the inverse covariance matrices S_c^{-1} and S_i^{-1} respectively. We plug this derivative into the equation (Eq. 7), and achieve the final equation by applying some elementary manipulations:

$$\begin{aligned}\frac{\partial \hat{\mathbf{p}}_{c,k}}{\partial \tilde{\mathbf{p}}_{c,k}} = \frac{1}{W} + \frac{1}{Wh^2}\Bigg\{ &\sum_{i\neq c}\left(S_{c,k}^{-1} + S_{i,k}^{-1}\right)\left(\tilde{\mathbf{p}}_i - \tilde{\mathbf{p}}_c\right) w_i \tilde{\mathbf{p}}_{i,k} \\ &- \hat{\mathbf{p}}_{c,k} \sum_{i\in\Omega_c}\left(S_{c,k}^{-1} + S_{i,k}^{-1}\right)\left(\tilde{\mathbf{p}}_i - \tilde{\mathbf{p}}_c\right) w_i \Bigg\}. \end{aligned} \tag{9}$$

Appendix B: Calculating the per-pixel covariance.

We calculate the 3×3 covariance matrix S_c (in Eq. 4) at each pixel c based on the world position samples $\mathbf{s}_{c,j}$ (Eq. 2). Let us define $S_c^{r,l}$ as the element at r-th row and l-th column in the matrix, and this value can be computed:

$$S_c^{r,l} = \frac{\sum_{j=1}^{n_c}\left(s_{c,j}^r - \bar{s}_{c,j}^r\right)\left(s_{c,j}^l - \bar{s}_{c,j}^l\right)}{n_c}, \tag{10}$$

where the $s_{c,j}^r$ and $s_{c,j}^l$ are the r-th and l-th element in the vector $\mathbf{s}_{c,j}$, respectively. Also, the $\bar{s}_{c,j}^r$ and $\bar{s}_{c,j}^l$ are the sample means of $s_{c,j}^r$ and $s_{c,j}^l$.

References

[BEM11] BAUSZAT P., EISEMANN M., MAGNOR M.: Guided image filtering for interactive high-quality global illumination. *Computer Graphics Forum 30*, 4 (2011), 1361–1368. 601, 602, 607

[BL07] BLU T., LUISIER F.: The SURE-LET approach to image denoising. *IEEE Transactions on Image Processing 16*, 11 (2007), 2778–2786. 605

[CPC84] COOK R. L., PORTER T., CARPENTER L.: Distributed ray tracing. *SIGGRAPH Comput. Graph. 18*, 3 (1984), 137–145. 600

[DHS*05] DURAND F., HOLZSCHUCH N., SOLER C., CHAN E., SILLION F. X.: A frequency analysis of light transport. *ACM Trans. Graph. 24*, 3 (2005), 1115–1126. 602

[DSHL10] DAMMERTZ H., SEWTZ D., HANIKA J., LENSCH H. P. A.: Edge-avoiding A-Trous wavelet transform for fast global illumination filtering. In *High Performance Graphics* (2010), pp. 67–75. 601, 602, 607

[ETH*09] EGAN K., TSENG Y.-T., HOLZSCHUCH N., DURAND F., RAMAMOORTHI R.: Frequency analysis and sheared reconstruction for rendering motion blur. *ACM Trans. Graph. 28*, 3 (2009), 93:1–93:13. 602

[Kaj86] KAJIYA J. T.: The rendering equation. In *ACM SIGGRAPH '86* (1986), pp. 143–150. 600

[KBS15] KALANTARI N. K., BAKO S., SEN P.: A machine learning approach for filtering Monte Carlo noise. *ACM Trans. Graph. 34*, 4 (2015), 122:1–122:12. 601, 602

Figure 13: *Results with the interactive filters. A-Trous (c) and guided filters (e) produce fairly noise results on the defocused area since these filters do not have a functionality to deal with noisy G-buffers, and these results are improved when our pre-filtered G-buffers are used instead of the noisy G-buffers.*

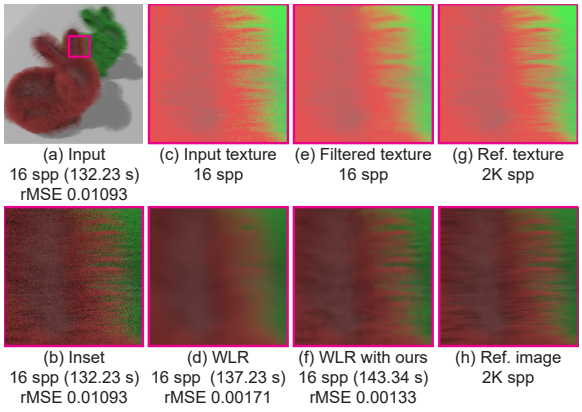

Figure 14: *Failure case given the furry Bunny models. Two complex models have different motions which make our per-pixel statistics very noisy. Given this challenging scenario, our method fails to produce high-quality pre-filtering results, e.g., over-blurred textures (e) compared to the reference textures (g). Our technique still improves the filtering quality of an existing method, WLR, but the final filtered image (f) loses the very detailed edges, compared to the reference image (h).*

[LWC12] LI T.-M., WU Y.-T., CHUANG Y.-Y.: SURE-based optimization for adaptive sampling and reconstruction. *ACM Trans. Graph. 31*, 6 (2012), 194:1–194:9. 601, 602, 605, 606, 607

[McC99] MCCOOL M. D.: Anisotropic diffusion for Monte Carlo noise reduction. *ACM Trans. Graph. 18*, 2 (1999), 171–194. 601

[MCY14] MOON B., CARR N., YOON S.-E.: Adaptive rendering based on weighted local regression. *ACM Trans. Graph. 33*, 5 (2014), 170. 600, 601, 602, 606, 607

[MIGYM15] MOON B., IGLESIAS-GUITIAN J. A., YOON S.-E., MITCHELL K.: Adaptive rendering with linear predictions. *ACM Trans. Graph. 34*, 4 (2015), 121:1–121:11. 601, 602

[MVH*14] MUNKBERG J., VAIDYANATHAN K., HASSELGREN J., CLARBERG P., AKENINE-MÖLLER T.: Layered reconstruction for defocus and motion blur. *Comput. Graph. Forum 33*, 4 (2014), 81–92. 602

[RKZ11] ROUSSELLE F., KNAUS C., ZWICKER M.: Adaptive sampling and reconstruction using greedy error minimization. *ACM Trans. Graph. 30*, 6 (2011), 159:1–159:12. 600, 608

[RKZ12] ROUSSELLE F., KNAUS C., ZWICKER M.: Adaptive rendering with non-local means filtering. *ACM Trans. Graph. 31*, 6 (2012), 195:1–195:11. 601

[RMZ13] ROUSSELLE F., MANZI M., ZWICKER M.: Robust denoising using feature and color information. *Computer Graphics Forum 32*, 7 (2013), 121–130. 601, 602, 606, 607, 609

[SD12] SEN P., DARABI S.: On filtering the noise from the random parameters in Monte Carlo rendering. *ACM Trans. Graph. 31*, 3 (2012), 18:1–18:15. 601

[SSD*09] SOLER C., SUBR K., DURAND F., HOLZSCHUCH N., SILLION F.: Fourier depth of field. *ACM Trans. Graph. 28*, 2 (2009), 18:1–18:12. 602

[Ste81] STEIN C. M.: Estimation of the mean of a multivariate normal distribution. *The annals of Statistics* (1981), 1135–1151. 601, 605

[XNZ08] XIANG S., NIE F., ZHANG C.: Learning a mahalanobis distance metric for data clustering and classification. *Pattern Recognition 41*, 12 (2008), 3600–3612. 605

[YJ06] YANG L., JIN R.: Distance metric learning: A comprehensive survey. *Michigan State Universiy 2* (2006), 78. 605

[ZJL*15] ZWICKER M., JAROSZ W., LEHTINEN J., MOON B., RAMAMOORTHI R., ROUSSELLE F., SEN P., SOLER C., YOON S.-E.: Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. *Computer Graphics Forum 34*, 2 (2015), 667–681. 601, 602