Fast Approximation to Large-Kernel Edge-Preserving Filters by Recursive Reconstruction from Image Pyramids

Tianchen Xu[†1,2] Jiale Yang[3] Yiming Qin[3] Bin Sheng[3] Enhua Wu[1,4‡]

[1]Key Lab of System Software (Chinese Academy of Sciences) & State Key Lab of Computer Science, Institute of Software, Chinese Academy of Sciences and University of Chinese Academy of Sciences, China
[2]Advanced Micro Devices, Inc. (AMD), China
[3]Department of Computer Science and Engineering, Shanghai Jiao Tong University, China
[4]Faculty of Science and Technology, University of Macau, China

Figure 1: *Fast real-time edge-preserving filtering in* 2560×1440 *(2K) resolution for (left) depth-of-field rendering effect, (mid) denoising of single sample per pixel (1-spp) ray traced reflection, and (right) depth filtering for normal reconstruction in screen-space particle rendering.*

Abstract

Edge-preserving filters, as known as bilateral filters, are fundamental to graphics rendering techniques, providing greater generality and capability of edge preservation than pure convolution filters. However, sampling with a large kernel per pixel for these filters can be computationally intensive in real-time rendering. Existing acceleration methods for approximating edge-preserving filters still struggle to balance blur controllability, edge clarity, and runtime efficiency. In this paper, we propose a novel scheme for approximating edge-preserving filters with large anisotropic kernels by recursively reconstructing them from multi-image pyramid (MIP) layers that are weightedly filtered in a dual 3×3 *kernel space. Our approach introduces a concise unified processing pipeline independent of kernel size, which includes upsampling and downsampling on MIP layers and enables the integration of custom edge-stopping functions. We also derive the implicit relations of the sampling weights and formulate a weight template model for inference. Furthermore, we convert the pipeline into a lightweight neural network for numerical solutions through data training. Consequently, our image post-processors achieve high-quality and high-performance edge-preserving filters in real-time, using the same control parameters as the original bilateral filters. These filters are applicable for depth-of-fields, global illumination denoising, and screen-space particle rendering. The simplicity of the reconstruction process in our pipeline makes it user-friendly and cost-effective, saving both runtime and implementation costs.*

CCS Concepts

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

1. Introduction

Edge-preserving filters are widely used in graphics rendering for image processing. In real-time rendering, several important image-processing techniques incorporate bilateral filters as a post-processing step. These techniques include depth of fields (DoF), spatial denoising for global illuminations (GI) using spatial-temporal denoising, variable-distance shadow mapping, and screen-space particle-based rendering [vdLGS09]. These applications require the smoothing of specific regions of an image while preserving distinct edges and preventing the blending of pixels

† E-mail: xutc@ios.ac.cn
‡ E-mail: weh@ios.ac.cn (corresponding author)

DOI: 10.2312/pg.20241302

across opposing sides of an edge. For example, when denoising 3D scenes in image space, it is crucial to smooth the pixels on the surfaces of 3D objects while maintaining clarity at the boundaries between different objects or sharp transitions. In image space, this is evident through abrupt gradients in the normal field and depth field. Another example is the DoF effect, where achieving a clear in-focus area while keeping the rest of the image blurred is essential. Additionally, when the foreground is in focus, its pixels should not extend into the background pixels. Conversely, when the background is in focus, the foreground needs to be blurred and superimposed on the background. Therefore, efforts need to be made to separate the foreground and background to process them separately. Edge-preserving filters are specifically designed to address these types of use cases and simplify the workload of processing images with distinct foreground and background regions.

Among edge-preserving filters, bilateral filters represent a fundamental class of algorithms designed for edge-preserving filtering effects. These filters achieve edge detection and avoidance by calculating weighted averages of samples. A typical bilateral filter consists of two filter functions, domain and range functions. The domain function determines the spatial proximity between pixels and controls the extent to which neighboring pixels influence each other during the filtering process. On the other hand, the range function measures the similarity between pixel values and determines the degree of influence based on the intensity or color similarity. This ensures that pixels with similar attributes are given more weight in the filtering process, while pixels with dissimilar attributes have less impact. However, it is difficult to make bilateral filters fast due to the anisotropic range functions. The range functions are data dependent and nonlinear, so it is not strictly separable into vertical and horizontal sampling processing. Although many applications employ separable bilateral filters as an approximation, they can introduce visual artifacts unless other shading techniques are used to hide them. Furthermore, there is still room for speed improvements in the approximation of separable edge-preserving filters for large kernel sizes, because they are still kernel-size dependent. Nevertheless, existing acceleration methods for approximating edge-preserving filters still struggle to balance blur controllability, edge clarity, and runtime efficiency.

This paper presents a novel scheme for approximating edge-preserving filters with large anisotropic kernels. The proposed scheme involves recursive reconstruction of these filters from multi-image pyramid (MIP) layers that are weightedly filtered in a dual 3×3 kernel space. The objective of our research is to approximate the target bilateral filters, assuming that their quality is sufficient for the intended applications. The paper makes the following major technical contributions:

- We propose a dual 3×3-kernel upsampling pattern that enables the integration of custom edge-stopping functions as the range function. To achieve this, we formulate a weight template model, which is an equation with unknown weight values. This model is derived by analyzing the underlying principles of the approximation and studying the relationships among the weight values of the finer and coarser samples.
- Then, we transform our pipeline into a lightweight neural network, leveraging machine learning to solve the unknown weights in the template model.
- Our inference processes can be implemented in a fast upsampling and down sampling pipeline.

2. Related Work

2.1. Convolution filters

Regarding optimization techniques for pure convolution filters, the Kawase filter serves as a classical approximation to Gaussian blur and has gained popularity in gaming for fast HDR bloom effects [Kaw03]. It employs iterative GPU sampling operations within a small domain, processing each pass at the original image resolution. Following this, Bjorge introduced a dual Kawase filter designed for rapid blurry effects on mobile devices [Bjo15], which introduces downsampling and upsampling work-flow with MIP maps for a single image. Then, Xu *et al.* proposed a weight model to approximate some typical convolution filters with MIP layer blending [XRW19]. However, implementing these rapid techniques directly to achieve bilateral filters poses challenges.

2.2. Edge-preserving filters

There are some previous achievements to accelerate real-time edge-aware processing. Chen *et al.* introduced the bilateral grid for edge-aware image processing [CPD07]. Nevertheless, their approach necessitates fine samples within a small spatial kernel. Subsequent studies involved transforming curves from the 2D image manifold into 5D and the real line, employing 1D edge-preserving filters to expedite image processing to accelerate the edge-aware smoothing [GO11,KLB13,CYS15]. Badri *et al.* introduced a first-order approximation method to accelerate edge-aware smoothing [BYA15]. Subsequently, the spatio-temporal edge-preserving filtering method was modified to achieve real-time performance using a permeability filter [SSC*17,MPS19,AS20]. Recently, some works employed a Gaussian adaptive bilateral filter to address inherent issues arising from noise filtering input and its effect on preserving image edges [CTY20,AA21]. Notably, a closely related study introduced Laplacian pyramid filters for edge-aware processing [PHK11]. Nonetheless, their approach may entail substantial computational costs as the kernel size increases.

2.3. Related applications

Depth of fields (DoF) often require large radii, leading to computational loads [Dem04, NKZN08, SSD*09]. To address the increasing computations arising from circle of confusion, heat diffusion equations (HDE) were introduced to simulate DoF by [LDOD06, ZCO10]. Furthermore, Gruen *et al.* enhanced the utilization of GPU parallel computation for the HDE solver [Gru11]. Besides, DoF faces some challenges [Com86, Dem04, BK08]. For example, focused objects might blend onto unfocused background objects [Ham08]. In terms of GI denoising, edge-avoiding filtering methods use À-Trous wavelet transforms to optimize denoisers [Fat09, DSHL10, HDL11]. Recently, Meunier and Harada introduced a weighted À-Trous linear regression instead of the traditional À-Trous wavelet transform to achieve real-time performance

of diffuse GI denoising [MH22]. Besides, screen-space particle-based fluid rendering can also be successfully accelerated through approximated bilateral Gaussian filters [vdLGS09].

2.4. Machine learning-based filters

Some employed convolution layers for approximating edge-preserving filters [XRY*15,WYS*20,LFY*21], while Gharbi *et al.* utilized a convolution neural network to predict coefficients within a locally-affine model in bilateral space [GCB*17]. Shahdoosti and Rahemi utilized edge maps derived from the Canny algorithm to restore edges of noisy images [SR19]. Moreover, Zhu *et al.* proposed a benchmark for edge-preserving image smoothing and built the baselines on deep neural networks [ZLJ*19]. Nonetheless, their methods lose the information of the finer samples and are computationally intensive due to their network structure.

Unlike other approaches, our method is a fast approximation to bilateral filters by employing image pyramids, which smartly applies edge-stopping functions to both downsampling and upsampling processes. Subsequently, the pipeline and sampling pattern are transformed into a pithy network to achieve numerically reduced errors. Additionally, our method is generally designed for various important applications in real-time rendering.

3. Principles and Techniques

A bilateral filter (BLF) combines two filter functions, namely the domain function and the range function, to achieve edge-preserving low-pass filtering [TM98]. The domain function f^{Ω} measures the geometric closeness between the center sample $\boldsymbol{p}$ and its nearby samples $\boldsymbol{q}$. It can be implemented as a convolution function controlled by a uniform or variable blur radius, such as the Gaussian function. On the other hand, the range function f^{R} prevents the nearby samples beyond the edges from affecting the center. Typically, it includes edge-stopping functions that rely on the values or attributes of the samples in the kernel domain. The general form of a BLF, including the joint BLF, can be expressed as follows:

$$I_{\mathrm{BLF}}(\boldsymbol{p}) = \frac{\sum_{\boldsymbol{q}\in\Omega} I(\boldsymbol{q}) f^{\Omega}(\boldsymbol{p},\boldsymbol{q}) f^{\mathrm{R}}(A(\boldsymbol{p}),A(\boldsymbol{q}))}{\sum_{\boldsymbol{q}\in\Omega} f^{\Omega}(\boldsymbol{p},\boldsymbol{q}) f^{\mathrm{R}}(A(\boldsymbol{p}),A(\boldsymbol{q}))} \tag{1}$$

where I and A represent a image sample and its attributes (A can be I for a single-image BLF, while it is a joint BLF when A differs from I), and Ω denotes the kernel domain.

Our fundamental idea is to utilize the filtered samples obtained from a multi-image pyramid (MIP), which are efficiently processed with 2×2-kernel filters, to approximate the desired bilateral filter. As depicted in Figure 2 (a), we initially employed a weighted combination of MIP layers to fit the domain and range functions in the target bilateral filter. However, direct implementation of this approach can lead to banding artifacts due to the fact that a coarse MIP layer stores the average values of a tile, with its center pixel always being the tile center. Therefore, we need to approximate each pixel of the original image as the center pixel from the coarse MIP layers through interpolation. This is accomplished by recursively blending the samples from each adjacent pair of MIP layers and deriving their weights from the target domain and range functions.

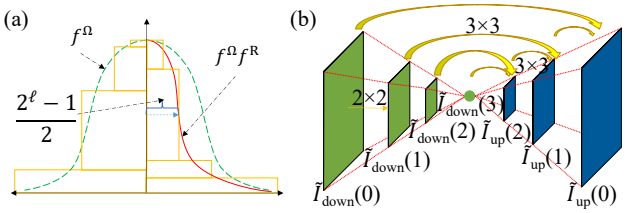


Figure 2: *(a) range function (edge-stopping functions) can change the shape of the original domain (convolution kernel) function anisotropically, and the weights (heights of orange boxes) to fit the target function curve are also adjusted; (b) our recursive reconstruction pipeline.*

For a dynamically updated image in real-time, our working pipeline can be designed as two phases of render passes: downsampling passes and upsampling passes, as illustrated in Figure 2 (b). Xu *et al.* have proven that recursive weighted upsampling can approximate convolution filters using a similar pipeline [XRW19]. For each pixel in an upsampling pass, they blend one sample from the finer level and one bilinearly-interpolated sample from the coarser level. However, this operation is isotropic, and therefore significant improvements are further required to enable edge-stopping functions to approximate bilateral filters, as the edge-stopping functions can modify the weights of MIP samples anisotropically (Figure 2 (a)). Our proposed method can achieve the approximation of edge-preserving filters while still converging within such a fast unified pipeline structure, while the operations inside significantly differ from the similar pipeline. The overall contents of the pipeline are described as follows:

Downsampling process We sample the data and attributes in the finer-level image using 2×2 filters with edge-stopping functions, and then output the coarser samples.

Upsampling process We reconstruct the target bilateral filter by recursively blending the samples from each adjacent pair of MIP layers, starting from the coarsest level to the finest level. The samples are gathered in a dual 3×3 pattern.

3.1. Notations and Formulation

For each adjacent pair of MIP layers, we define the current level ℓ as the finer level and the next level $(\ell+1)$ as the coarser level. A sample from the MIP map is denoted by $\tilde{I}(\boldsymbol{p},\ell)$. We use two MIP maps, $\tilde{I}_{\mathrm{down}}$ generated by downsampling, and $\tilde{I}_{\mathrm{up}}$ generated by upsampling. We first summarize these and other notations used by the later texts in Table 1.

Since our core process is in the upsampling passes, we will present a preview of our sampling pattern and the corresponding formulation in the upsampling process here. We gather 3×3 samples from the finer layer of $\tilde{I}_{\mathrm{down}}$ and 3×3 samples from the coarser layer of the previously upsampled image $\tilde{I}_{\mathrm{up}}$. The sampling positions follow a dual 3×3 kernel pattern, as illustrated in Figure 3. Then, our weight template model can be formulated in Equation (2), as follows:

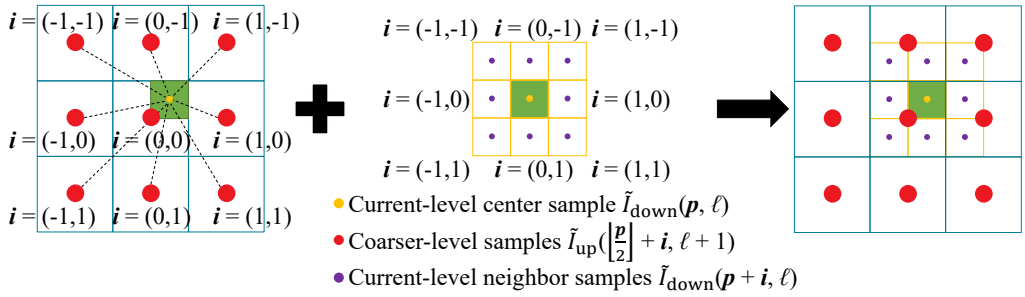


Figure 3: *Upsampling pattern: 9 samples in* 3×3 *kernel at the finer (1 center and 8 supplementary neighbors) and 9 samples in* 3×3 *kernel at the coarser level. Here it shows an example pattern of the center finer sample location at* $\boldsymbol{p}$ *with* $(\boldsymbol{p}_x \mod 2) = 1$ *and* $(\boldsymbol{p}_y \mod 2) = 0$*, and there are the other 3 cases.*

Table 1: *Notations.*

Name	Domain	Description
$\boldsymbol{p}$	$\mathbb{Z}^2_{[0,(W,H))}$	pixel (x,y) of image in size (W,H)
$\boldsymbol{m}$	$\mathbb{Z}^2_{[0,(W,H))}$	major pixel for 2×2 downsampling
$\boldsymbol{i}, \boldsymbol{j}$	$\{-1,0,1\}^2$	pixel position offsets
ℓ	$\mathbb{N}$	MIP level
$\tilde{I}_{\text{down}}(\boldsymbol{p}, \ell)$	$\mathbb{R}^3$	sample value of pixel $\boldsymbol{p}$ at level ℓ from the downsampled MIP
$\tilde{I}_{\text{up}}(\boldsymbol{p}, \ell)$	$\mathbb{R}^3$	sample value of pixel $\boldsymbol{p}$ at level ℓ from the upsampled MIP
$\tilde{A}(\boldsymbol{p}, \ell)$	$\mathbb{R}^n$	attribute value of pixel $\boldsymbol{p}$ at level ℓ
f^{R}	$\mathbb{R}$	range function
$f^{\text{R}}_{\text{f}}(\boldsymbol{i})$	$\mathbb{R}$	range function for a finer sample, $f^{\text{R}}\left(\tilde{A}(\boldsymbol{p}, \ell), \tilde{A}(\boldsymbol{p} + \boldsymbol{i}, \ell)\right)$, for short
$f^{\text{R}}_{\text{c}}(\boldsymbol{i})$	$\mathbb{R}$	range function for a coarser sample, $f^{\text{R}}\left(\tilde{A}(\boldsymbol{p}, \ell), \tilde{A}(\lfloor \frac{\boldsymbol{p}}{2} \rfloor + \boldsymbol{i}, \ell + 1)\right)$, for short
$w_{\text{f}}(\boldsymbol{p}, \boldsymbol{i}, \ell)$	$[0,1]$	weight function for a finer sample
$w_{\text{c}}(\boldsymbol{p}, \boldsymbol{i}, \ell)$	$[0,1]$	weight function for a coarser sample
w^{-1}	$[0,1]$	normalization factor
$w^{\text{R}}_{\text{f}}(\boldsymbol{p}, \boldsymbol{i}, \ell)$	$[0,1]$	finer-sample weight-contribution term influenced by the range functions of coarse samples
$w_{\text{b}}(\boldsymbol{p}, \boldsymbol{j}, \ell)$	$[0,1]$	basic weight for a coarser sample
$w_{\text{d}}(\boldsymbol{p}, \boldsymbol{j})$	$[0,1]$	interpolation weight for a coarser sample to correct the symmetry, evaluated according to its distance to center $\boldsymbol{p}$
$w^{\Omega}_{\text{f}}(\boldsymbol{i}, \ell)$	$[0,1]$	weight distribution for a finer sample influenced by its domain function; unknown, solved by training
$w^{\Omega}_{\text{c}}(\boldsymbol{i}, \ell + 1)$	$[0,1]$	weight distribution for a coarser smp. influenced by its domain function; unknown, solved by training
γ	$[0,1]$	exponential weight for w^{R}_{f} and w_{c} interpolation, solved by tuning

$$\begin{aligned}
\tilde{I}_{\text{up}}(\boldsymbol{p}, \ell) &= w^{-1} \sum_{\boldsymbol{i}} \tilde{I}_{\text{down}}(\boldsymbol{p} + \boldsymbol{i}, \ell) w_{\text{f}}(\boldsymbol{p}, \boldsymbol{i}, \ell) \\
&\quad + w^{-1} \sum_{\boldsymbol{i}} \tilde{I}_{\text{up}}\left(\lfloor \frac{\boldsymbol{p}}{2} \rfloor + \boldsymbol{i}, \ell + 1\right) w_{\text{c}}(\boldsymbol{p}, \boldsymbol{i}, \ell), \text{ where} \qquad (2)
\end{aligned}$$

$$\begin{cases}
w^{-1} = \dfrac{1}{\sum_{\boldsymbol{i}} w_{\text{f}}(\boldsymbol{p}, \boldsymbol{i}, \ell) + \sum_{\boldsymbol{i}} w_{\text{c}}(\boldsymbol{p}, \boldsymbol{i}, \ell)} & (2a) \\
w_{\text{f}}(\boldsymbol{p}, \boldsymbol{i}, \ell) = w^{\Omega}_{\text{f}}(\boldsymbol{i}, \ell) f^{\text{R}}_{\text{f}}(\boldsymbol{i}) + w^{\text{R}}_{\text{f}}(\boldsymbol{p}, \boldsymbol{i}, \ell) & (2b) \\
w_{\text{c}}(\boldsymbol{p}, \boldsymbol{i}, \ell) = w_{\text{b}}(\boldsymbol{p}, \boldsymbol{i}, \ell) f^{\text{R}\gamma}_{\text{c}}(\boldsymbol{i}) & (2c) \\
w^{\text{R}}_{\text{f}}(\boldsymbol{p}, \boldsymbol{i}, \ell) = \left(\sum_{\boldsymbol{j}} w_{\text{b}}(\boldsymbol{p}, \boldsymbol{j}, \ell) \left(1 - f^{\text{R}\gamma}_{\text{c}}(\boldsymbol{j})\right)\right) \dfrac{f^{\text{R}}_{\text{f}}(\boldsymbol{i})}{\sum_{\boldsymbol{j}} f^{\text{R}}_{\text{f}}(\boldsymbol{j})} & (2d) \\
w_{\text{b}}(\boldsymbol{p}, \boldsymbol{i}, \ell) = w_{\text{d}}(\boldsymbol{p}, \boldsymbol{i}) f^{\text{R}}_{\text{c}}(\boldsymbol{i}) w^{\Omega}_{\text{c}}(\boldsymbol{i}, \ell + 1) & (2e) \\
w_{\text{d}}(\boldsymbol{p}, \boldsymbol{i}) = \prod_{a \in \{x,y\}} \dfrac{17 - 3|4\boldsymbol{i}_a - 2(\boldsymbol{p}_a \mod 2) + 1|}{24} & (2f) \\
1 = \sum_{\boldsymbol{i}} w^{\Omega}_{\text{f}}(\boldsymbol{i}, \ell) + \sum_{\boldsymbol{i}} w_{\text{d}}(\boldsymbol{p}, \boldsymbol{i}) w^{\Omega}_{\text{c}}(\boldsymbol{i}, \ell + 1) & (2g)
\end{cases}$$

In the equation above, the generation of MIP map $\tilde{I}_{\text{down}}$ will be introduced in section 3.2, and the details of our weight template model will be derived in section 3.3.

3.2. Downsampling Process

During the downsampling process, source samples and attributes are filtered using 2×2 kernels to prepare the samples for the upsampling processes. In each pass, we also incorporate the range function into the 2×2 kernel to preserve the edges, instead of simply using an averaging operation. Since a 2×2 kernel does not have a center pixel, we introduce the concept of a major pixel $\boldsymbol{m}$ to serve a similar purpose. The major pixel is the one closest to the average of the 2×2 pixels in the finer image. For a generalized formulation to determine it, we can substitute the average value into the center-pixel variable in the range function, and then evaluate the range-function value for each 2×2 pixel. The major pixel is identified as the one with the highest range-function value. Since the range function is only used for comparison, it can be unnormalized for implementation optimization. Thus, we can generalize the downsampling operation into the following mathematical form:

$$\tilde{I}_{\text{down}}(\boldsymbol{p},\ell+1)=\frac{\sum_{\boldsymbol{i}\in\{0,1\}^2}\tilde{I}_{\text{down}}(\boldsymbol{q},\ell)f^{\text{R}}\left(\tilde{A}(\boldsymbol{m},\ell),\tilde{A}(\boldsymbol{q},\ell)\right)}{\sum_{\boldsymbol{i}\in\{0,1\}^2}f^{\text{R}}\left(\tilde{A}(bmm,\ell),\tilde{A}(\boldsymbol{q},\ell)\right)} \quad (3)$$

$$\text{where}\begin{cases}\boldsymbol{q}=2\boldsymbol{p}+\boldsymbol{i}\text{ for short}\\ \arg\max_{\boldsymbol{m}} f^{\text{R}}\left(\sum_{\boldsymbol{i}\in\{0,1\}^2}\frac{\tilde{A}(\boldsymbol{q},\ell)}{4},\tilde{A}(\boldsymbol{m},\ell)\right)\end{cases}$$

For attributes such as the normal vector, depth, roughness, and circle-of-confusion (CoC) values in G-buffers, we generally use the same formula as Equation (3) to substitute I. However, for denoising purposes, we aim to further enhance the impact of the edge-stopping functions (refer to Figure 4). As a result, we directly take the attributes of the major pixel $\boldsymbol{m}$ as the coarser output, as they have the maximum range-function value. This approach follows the concept known as nearest neighbor upsampling [JB11].

(a) PSNR SSIM: (b) 28.32 0.96672(c) 41.72 0.99514(d) 43.69 0.99637

Figure 4: *Different downsampling strategies for denoising: (a) ground-truth reference, (b) no range function – shaded or bright speck artifacts at the corner and near the creases of the checkerboard (marked by red boxes), (c) weighted average of attributes with the range function – weak bright specks and slightly blurry boundaries of the checkerboard tiles (marked by red boxes), and (d) attribute enhancement with the major pixel – quality closer to the ground truth (higher PSNR and SSIM) than (c).*

3.3. Upsampling Process

The upsampling process is the primary stage where the target-filtered result is reproduced by utilizing the downsampled outcomes. It entails conducting upsampling passes to hierarchically reconstruct the target from low resolution to high resolution. The fundamental components of this process are our dual 3×3 sampling pattern and sampling weight template. Ultimately, we convert the template model into a neural network to train the values for the numerical solution of the unknown weights.

3.3.1. Weight Template Model

We first focus on the weights of the coarser samples, which are highly correlated with the range function (edge-stopping functions) and involve complex processing.

Based on the sampling pattern depicted in Figure 3, the 3×3 kernel for the coarser samples is asymmetric with respect to the output pixel position (which coincides with the center finer sample). To ensure symmetric contributions to the output pixel position, we introduce weight factor w_{d} for interpolation. Similar to bilinear interpolation, the function w_{d} for each coarser sample can be decomposed into the multiplication of two per-dimension factors,

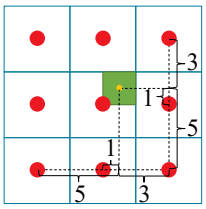


Figure 5: *For each dimension of x and y, the distances* $d_a(\boldsymbol{p},\boldsymbol{i})=|4\boldsymbol{i}_a-2(\boldsymbol{p}_a \mod 2)+1|, a\in\{x,y\}$ *from the coarser samples to the output pixel position can be* 1, 3, *and* 5 *in integer units.*

$w_{\text{d}}(\boldsymbol{p},\boldsymbol{i})=\alpha_{\text{d}}(d_x)\cdot\alpha_{\text{d}}(d_y)$. We model α_{d} using a decreasing linear function with respect to the horizontal or vertical distance to the finer center, $d_a(\boldsymbol{p},\boldsymbol{i})=|4\boldsymbol{i}a-2(\boldsymbol{p}_a \mod 2)+1|$, where $a\in x,y$, for each dimension of x and y. The distance calculation is illustrated in Figure 5. According to the domain of $\boldsymbol{p}$ and $\boldsymbol{i}$, the range of d_a is $\{1,3,5\}$, while the corresponding range of the signed distance is either $\{-1,3,-5\}$ or $\{1,-3,5\}$. Therefore, the model of the per-dimension factor $\alpha_{\text{d}}(d_a)$ for wd satisfies the interpolation constraint that the signed distance of the finer center, which is zero, should be expressible by linear interpolation, as demonstrated in the following equations:

$$\begin{cases}\alpha_{\text{d}}(1)-3\alpha_{\text{d}}(3)+5\alpha_{\text{d}}(5)=0\\ \alpha_{\text{d}}(1)+\alpha_{\text{d}}(3)+\alpha_{\text{d}}(5)=1\\ \alpha_{\text{d}}(d_a)=kd_a+b\end{cases} \quad (4)$$

$$\text{Solution: }\alpha_{\text{d}}(d_a)=-\frac{1}{8}d_a+\frac{17}{24}=\frac{17-3d_a}{24}$$

We then obtain the value of w_{d} as shown in Equation (2f).

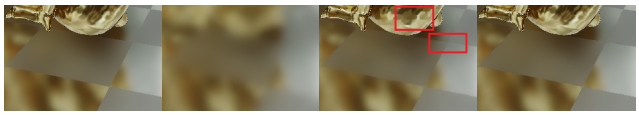
Subsequently, we reduce the contribution of each coarser sample that is separated from the center finer sample by an edge by masking its range function (edge-stopping function). Then, we can calculate the basic unnormalized weight w_{b} of each coarser sample by multiplying all weight factors for each coarser sample, including w_{d}, f_{c}^{R}, and w_{c}^{Ω}, as shown in Equation (2e), where the unknown w_{c}^{Ω} needs to be determined through data training:

$$w_{\text{b}}(\boldsymbol{p},\boldsymbol{i},\ell)=w_{\text{d}}(\boldsymbol{p},\boldsymbol{i})f_{\text{c}}^{\text{R}}(\boldsymbol{i})w_{\text{c}}^{\Omega}(\boldsymbol{i},\ell+1) \quad (5)$$

Here, we introduce the weights of the finer samples. The weight function of each finer sample consists of two terms: one influenced by the domain function and the other influenced by the range function of the coarser samples, as shown in Equation (2b):

$$w_{\text{f}}(\boldsymbol{p},\boldsymbol{i},\ell)=w_{\text{f}}^{\Omega}(\boldsymbol{i},\ell)f_{\text{f}}^{\text{R}}(\boldsymbol{i})+w_{\text{f}}^{\text{R}}(\boldsymbol{p},\boldsymbol{i},\ell) \quad (6)$$

For each finer sample, except for the impact of its range function f_{f}^{R}, the weight factor w_{f}^{Ω} in the term related to the domain function needs to be determined through data training. When the contribution of the coarser sample is weakened by its range function, we introduce the contribution of the finer samples to compensate for its reduced contribution. This can effectively weaken some dithered artifacts, as shown in Figure 6. We first average the finer samples masked by their own range functions f_{f}^{R}, normalize the result, and obtain the compensatory contribution $\frac{\sum_{\boldsymbol{i}}\tilde{I}_{\text{down}}(\boldsymbol{p}+\boldsymbol{i},\ell)f_{\text{f}}^{\text{R}}(\boldsymbol{i})}{\sum_{\boldsymbol{j}}f_{\text{f}}^{\text{R}}(\boldsymbol{j})}$.



(a) PSNR SSIM: (b) 21.60 0.79829(c) 30.86 0.95641(d) 42.05 0.99441

Figure 6: *Different upsampling strategies: (a) ground-truth reference, (b) no range function – too blurry, (c) no compensation – slightly blurry on the boundaries of checkerboard tiles and the dragon blurrier than expected (marked by red boxes), and (d) our full upsampling model – quality closer to the ground truth (higher PSNR and SSIM) than (c).*

The weights $w_{\mathrm{f}}^{\mathrm{R}}$ of these finer samples are influenced by the range function of the coarser samples. Hence, we adopt a linear interpolation to distribute the weight value w_{b} to $w_{\mathrm{f}}^{\mathrm{R}}$ and w_{c}, as shown in Equation (2c and 2d).

$$\begin{cases} w_{\mathrm{c}}(\boldsymbol{p},\boldsymbol{i},\ell) & = w_{\mathrm{b}}(\boldsymbol{p},\boldsymbol{i},\ell) f_{\mathrm{c}}^{\mathrm{R}\gamma}(\boldsymbol{i}) \\ w_{\mathrm{f}}^{\mathrm{R}}(\boldsymbol{p},\boldsymbol{i},\ell) & = \left(\sum_{\boldsymbol{j}} w_{\mathrm{b}}(\boldsymbol{p},\boldsymbol{j},\ell) \left(1 - f_{\mathrm{c}}^{\mathrm{R}\gamma}(\boldsymbol{j})\right) \right) \dfrac{f_{\mathrm{f}}^{\mathrm{R}}(\boldsymbol{i})}{\sum_{\boldsymbol{j}} f_{\mathrm{f}}^{\mathrm{R}}(\boldsymbol{j})} \end{cases} \tag{7}$$

If the range function value is large, the contribution will shift towards the coarser samples. Conversely, the contribution will shift towards the finer samples. However, the impact of the range function on $w_{\mathrm{f}}^{\mathrm{R}}$ varies depending on the specific bilateral model. Therefore, we use multiple powers of the range function as linear interpolation parameters $f_{\mathrm{c}}^{\mathrm{R}\gamma}$ with assigned exponential weights γ that need to be determined through parameter tuning.

Additionally, all weights related to the domain function in Equation (2) obey a constraint that their sum should be 1, as shown in Equation (2g). This constraint is evident because if we remove all range functions by constantly setting each value of the range function to 1, the model becomes a convolution-approximation model [XRW19].

3.3.2. Training with Machine Learning

In each upsampling pass, there are a total of 19 unknown weights: 9 for the coarser weights, 9 for the finer weights, and one exponential weight γ. Equation (2g) allows us to eliminate the unknown weight for the center finer sample. As a result, we need to solve for 18 unknown weights per upsampling pass through data training.

For the downsampling passes, we simply precompute the multi-image pyramid (MIP), so our focus is primarily on the upsampling passes. We employ stochastic gradient estimation followed by gradient descent to solve for the unknown weights [DHB24]. To prevent divergence, we reject the iteration's result if the error is even larger than that of the previous iteration.

When it comes to model training, collecting a large amount of data from real-world applications that produce results with joint bilateral filters as ground-truth data can be challenging. To simplify the training process, we leverage 4096 images (resolution $\geq 256 \times 256$) randomly selected from the ILSVRC dataset [RDS*15], and obtain the learned functions and weights that are used during inference. This is possible because the unknown weights are solely related to the domain function mathematically. However, the range functions can interfere with the training accuracy.

To generate ground truth data, we strictly filter single color-images using edge-stopping functions generated with Gaussian functions of color difference, each with different deviations for added variety. We then employ pixel loss ($\mathcal{L}_1$ loss) and perceptual loss [JAFF16] to penalize our results. Here, perceptual loss is incorporated to improve the structural similarity to the ground truth. Our ablation analysis indicates its effectiveness in reducing the visual difference in blurriness intensities and distribution, as exemplified in Figure 7. Then, the optimization of functions and weights is performed using the Adam optimizer.

(a) PSNR SSIM: (b) 44.17 0.99269 (c) 47.08 0.99782

Figure 7: *Learned results (Huangshan pine leaves) using different loss functions (blur radius fixed to unified value 24 for clarity): (a) ground-truth reference, (b) $\mathcal{L}_1$ loss only – observable differences from the ground truth in blurriness intensities and distribution (marked by red boxes, please zoom in), and (c) $\mathcal{L}_1$ + perceptual loss – visual quality improved with higher PSNR and SSIM than (b).*

During the inference stage, the reconstruction processes utilize the learned weights to obtain the desired targets. Once the weights are learned, we organize them into a look-up table through layer collapse and expansion. This allows us to virtually traverse the forward pipeline for inference using Equation (2) on compute shaders.

4. Use-Case Implementations

Using our method, we implement three important use cases in fully real-time rendering applications.

4.1. Depth of Fields (DoF)

To implement the post-process DoF effect, we use adaptive bilateral depth filtering [WYS*12] as a basis. This approach provides a perfect background blur and effectively prevents the foreground in focus from blending into the background. However, it does not handle the case where the foreground is blurred over a sharp background in focus. Therefore, we have made some simple modifications to improve their model and make it more like a bilateral filter (although not strictly a bilateral filter) as our target. The target filter functions are defined as follows:

$$\begin{cases} f^{\Omega}(\boldsymbol{p},\boldsymbol{q}) & = \exp\left(-\dfrac{18\|\boldsymbol{p}-\boldsymbol{q}\|^2}{A^2(\boldsymbol{p})} - \dfrac{\|\boldsymbol{p}-\boldsymbol{q}\|^2}{2\sigma^2} \right) \\ f^{\mathrm{R}}(A(\boldsymbol{p}),A(\boldsymbol{q})) & = \exp\left(-\dfrac{18\|\boldsymbol{p}-\boldsymbol{q}\|^2}{\min^2\{A(\boldsymbol{p}),A(\boldsymbol{q})\}} \right) \end{cases} \tag{8}$$

where attribute A represents the circle-of-confusion (CoC) that can be derived from the depth buffer, and σ is a adjustment parameter of the domain function to control the blur radius.

Subsequently, we follow Equation (3) to downsample both color and CoC values to generate MIPs. In each upsampling pass, we evaluate the range function $f_c^R(\boldsymbol{i})$ for each coarser sample by substituting $\|\boldsymbol{p}-\boldsymbol{q}\|$ with $\frac{\sqrt{2^{\ell+1}-1}}{2}$.

4.2. Spatial Denoising for Global Illuminations

Spatial and temporal denoising is one of the most valuable use cases in real-time rendering, especially with the increasing popularity of ray tracing. In addition to temporal supersampling, the spatial filtering component also has a significant impact on the overall denoising quality. Our GI denoiser algorithm is based on spatiotemporal variance-guided filtering [SKW*17], with the edge-stopping functions referring to FidelityFX [AMD20], using a Gaussian-based bilateral filter:

$$\begin{cases} f^{\Omega}(\boldsymbol{p},\boldsymbol{q}) & = \exp\left(-\dfrac{9\|\boldsymbol{p}-\boldsymbol{q}\|^2}{2(\sigma_\alpha\sqrt{\alpha_{\boldsymbol{p}}}+1)^2}\right) \\ f^{R}(A(\boldsymbol{p}),A(\boldsymbol{q})) & = \left(\hat{\boldsymbol{N}}_{\boldsymbol{p}}\cdot\hat{\boldsymbol{N}}_{\boldsymbol{q}}\right)^{\sigma_{\boldsymbol{N}}}\cdot\exp\left(-\sigma_z z_{\boldsymbol{p}}|z_{\boldsymbol{p}}-z_{\boldsymbol{q}}|\right) \\ & \quad\cdot\text{smoothstep}\left(0,0.5,|\sqrt{\alpha_{\boldsymbol{q}}}-\sqrt{\alpha_{\boldsymbol{p}}}|\right) \end{cases} \tag{9}$$

where attributes $A=(\hat{\boldsymbol{N}},z,\sqrt{\alpha})$ denote the normal vector, depth, and roughness values accordingly, and $\sigma_{\boldsymbol{N}}$, σ_z, and σ_α are their adjustment parameters correspondingly. Function $\text{smoothstep}(e_0,e_1,x)$ refers to the smooth Hermite interpolation for short in graphics terminology.

During the downsampling passes, we apply Equation (3) to filter the color and use the major-pixel attribute scheme to enhance the downsampled attributes. In each upsampling pass, we evaluate the range function $f_c^R(\boldsymbol{i})$ of each coarser sample by substituting α, $\hat{\boldsymbol{N}}$, and z with the corresponding attributes in the coarser sample. Furthermore, for diffuse denoising, we set the roughness $\sqrt{\alpha}$ to be 1 and remove the roughness comparison factor from Equation (9).

4.3. Screen-Space Particle Rendering

Bilateral filters can also be used to filter the depth field for normal field reconstruction in screen-space particle rendering. This technique is widely used in recent real-time game engines for particle-based liquid rendering, as it offers efficiency compared to other strict fluid rendering methods. By blurring the depth values of spheres using a bilateral filter, the normal vectors can be recovered from the depth field, resulting in a droplet-condensed liquid effect [vdLGS09]. Although this use case may appear simple, we have incorporated it to showcase the versatility of our method in addressing various real-time rendering problems across different domains. The target bilateral filter is modeled as follows:

$$\begin{cases} f^{\Omega}(\boldsymbol{p},\boldsymbol{q}) & = \exp\left(-\dfrac{\|\boldsymbol{p}-\boldsymbol{q}\|^2}{2\sigma^2}\right) \\ f^{R}(A(\boldsymbol{p}),A(\boldsymbol{q})) & = \exp\left(-\sigma_z z_{\boldsymbol{p}}|z_{\boldsymbol{p}}-z_{\boldsymbol{q}}|\right) \end{cases} \tag{10}$$

where attribute $A=z$ represents the depth value, and σ is an adjustment parameter of the domain function that controls the blur radius. In this case, we use the same edge-stopping function as the depth factor in Equation (9).

5. Results

In our experiments, we conduct several groups of comparative test cases to verify the quality and performance advantages of our method. All 2560 × 1440-resolution (2K) test cases run on a system equipped with an AMD Ryzen™ 7 5700X CPU with 8 cores running at 3.4GHz and a Radeon™ RX 6800 GPU (core clock 1815MHz, memory clock 2000MHZ, and memory size 16GB). We also tested 1080p results on a laptop equipped with an AMD Ryzen™ 9 5900HX CPU and an NVIDIA RTX 3080 GPU.

5.1. Quality Evaluations

For quality verification, we measure the peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) of our generated images to the ground truth images by brute-forcing bilateral filtering. In our experiments, the average PSNR and SSIM are 41.17 and 0.99254, respectively. Additionally, the PSNR and SSIM of each example are attached in the corresponding figures. Then, we discuss the qualitative results of each use case accordingly.

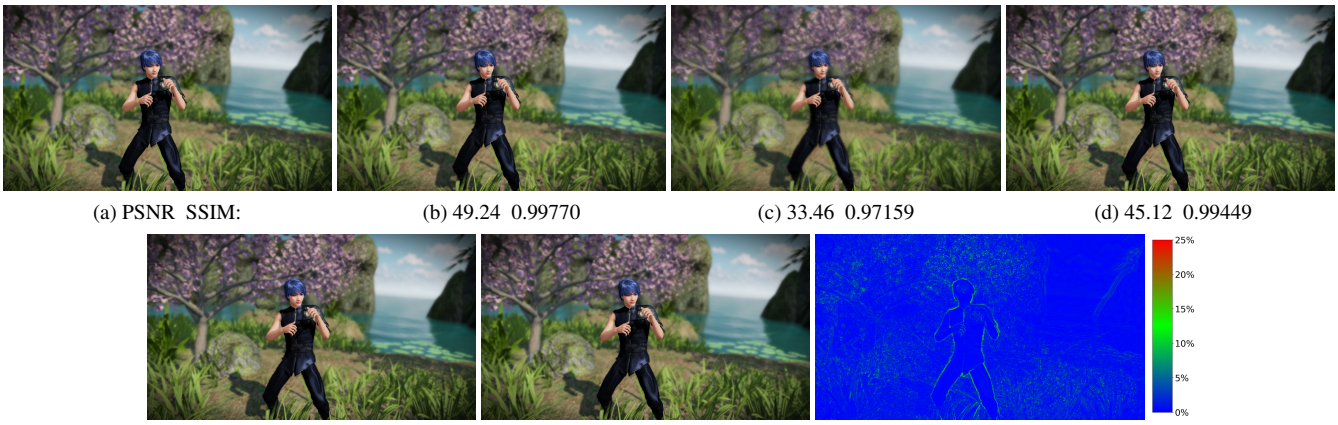
Depth of Fields Our DoF test program showcases the real-time application of our DoF method with character animation in a complex 3D scenario. Figure 8 illustrates a set of representative screenshots captured from the results of the DoF test. The images, in sequential order, include the ground-truth reference (a), the approximation using a separable-pass bilateral filter (b), the nonuniform convolution approximation with bilateral-filtered MIP maps (c) [XRW19], the simulation using the heat diffusion solver (d) [Gru11], the approximation using our weight template model without neural network training (e), and our full solution with neural network training (f).

It is important to note that bilateral filters are not strictly separable, and the separable-pass bilateral filter (b) only deviates from the ground truth on high-frequency edges.

Method (c) solely employs our downsampling process while retaining the upsampling process of the nonuniform convolution approximation. It successfully avoids blurry colors bleeding into sharp pixels but fails to prevent pixel colors in focus from bleeding into the blurry background pixels, thus not fully satisfying the edge-preserving requirement. In contrast, our method preserves edge clarity and prevents color bleeding in both directions.

Method (e) uses the following settings for the unknown weights: $w_f^{\Omega}(\boldsymbol{0},\ell)=\frac{4^{\ell}\partial f^{\Omega}}{\partial\ell}$ [XRW19], $w_f^{\Omega}(\boldsymbol{i},\ell)=0$ for $\boldsymbol{i}\neq\boldsymbol{0}$, $w_c^{\Omega}(\boldsymbol{i},\ell+1)=1-w_f^{\Omega}(\boldsymbol{0},\ell)$ for $\forall\boldsymbol{i}$, and $\gamma=1$, serving as an analytic solution. Although it produces an acceptable visual result, noticeable differences from the ground truth are observed. Quantitatively, our full solution (f) with neural network training significantly improves the quality and reduces the error compared to the ground truth, as indicated by PSNR and SSIM.

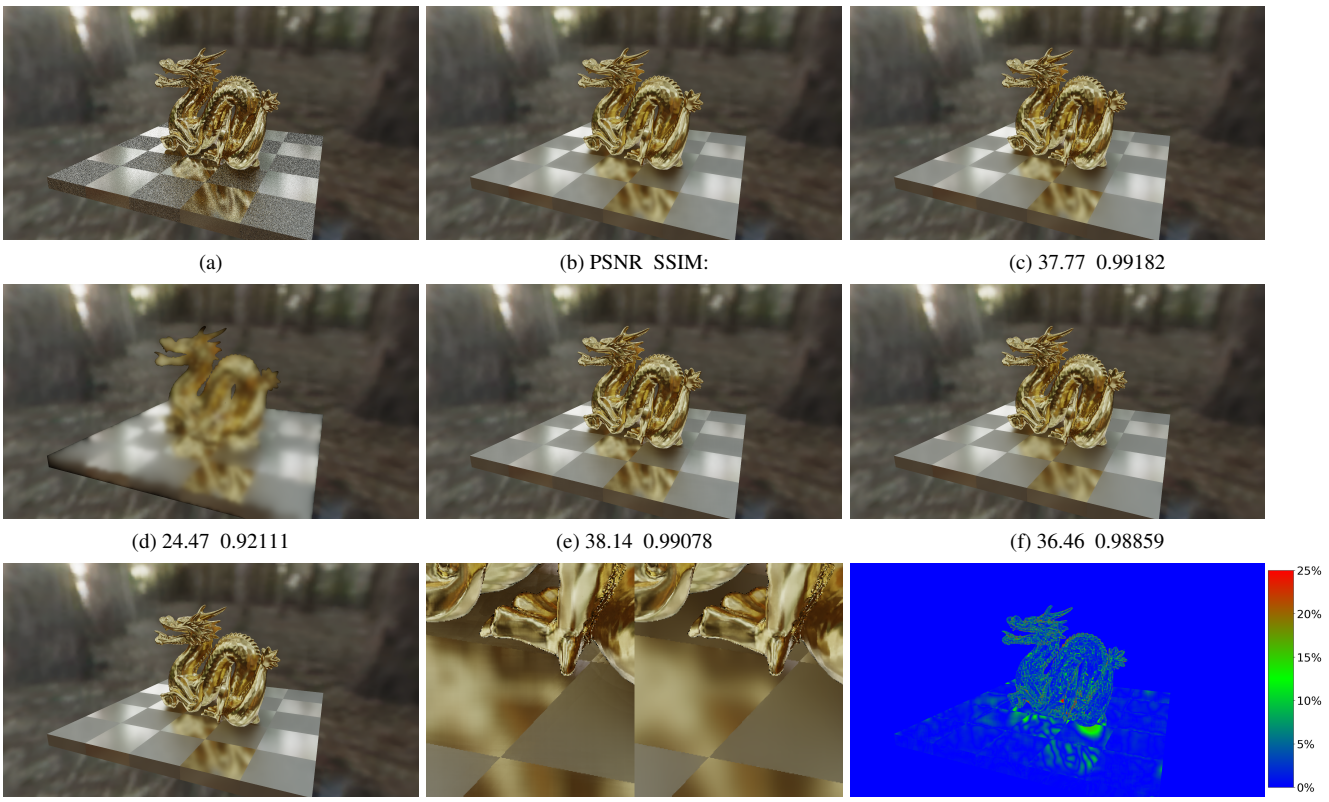
Our test program offers a comparative mode, which is capable of rendering the results of all methods to be compared for the same frame. We have also written a script to automatically capture 10,000 (10K) random frame screenshots and calculate the PSNR and SSIM metrics. The average scores for our method are PSNR: 42.87 and SSIM: 0.99059, while the lowest scores are PSNR: 39.39 and SSIM: 0.98126. Furthermore, we have recorded a video demo



(a) PSNR SSIM: (b) 49.24 0.99770 (c) 33.46 0.97159 (d) 45.12 0.99449

(e) 39.74 0.98273 (f) 42.48 0.99005 (g)

Figure 8: *Real-time DoF-effect quality: (a) ground-truth reference, (b) approximation by the separable-pass bilateral filter, (c) nonuniform convolution approximation with bilateral-filtered MIP maps, (d) simulation by heat-diffusion equation solver, (e) approximation by our weight template model without neural network training, (f) our full solution with neural network training, and (g) the difference heap map from the ground truth to our result (please zoom in the 2K images).*



(a) (b) PSNR SSIM: (c) 37.77 0.99182

(d) 24.47 0.92111 (e) 38.14 0.99078 (f) 36.46 0.98859

(g) 39.68 0.99394 (h) (i)

Figure 9: *Real-time reflection denoising quality: (a) input noised texture by 1-spp ray tracing, (b) ground-truth reference, (c) approximation by separable-pass bilateral filter, (d) nonuniform convolution approximation with bilateral-filtered MIP maps, (e) À-Trous-wavelet denoising, (f) approximation by our weight template model without neural network training, (g) our full solution with neural network training (h) zoomed-in details of À-Trous-wavelet denoising (left) and our method (right), and (i) the difference heap map from the ground truth to our result (please zoom in the 2K images).*

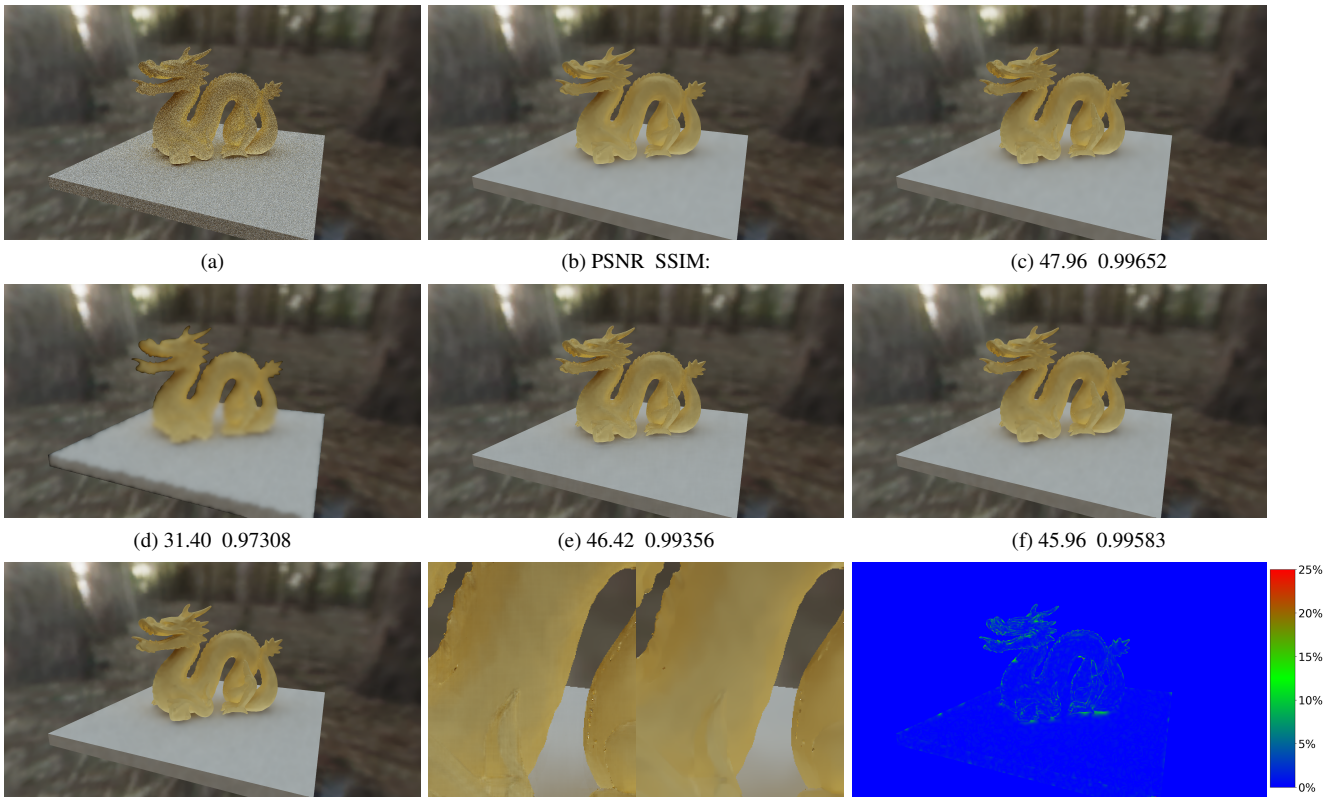


Figure 10: *Real-time diffuse denoising: (a) input noised texture by 1-spp ray tracing, (b) ground-truth reference, (c) approximation by separable-pass bilateral filter, (d) nonuniform convolution approximation with bilateral-filtered MIP maps, (e) À-Trous-wavelet denoising, (f) approximation by our weight template model without neural network training, (g) our full solution with neural network training (h) zoomed-in details of À-Trous-wavelet denoising (left) and our method (right), and (i) the difference heap map from the ground truth to our result (please zoom in the 2K images).*

of our test program. This demo demonstrates that our method functions effectively in real-world applications without any noticeable temporal artifacts.

Spatial Denoising for GI Figure 9 presents the representative screenshots from the results of the reflection-denoising test. The images, shown in sequential order, include the input image with noise obtained through ray tracing with a single sample per pixel (1-spp) (a), the ground-truth reference generated by brute-forcing bilateral filtering (b), the nonuniform convolution approximation with bilateral-filtered MIP maps (c) [XRW19], the À-Trous wavelet denoiser (d) [DSHL10, HDL11], the approximation using our weight template model without neural network training (e), and our full solution with neural network training (f).

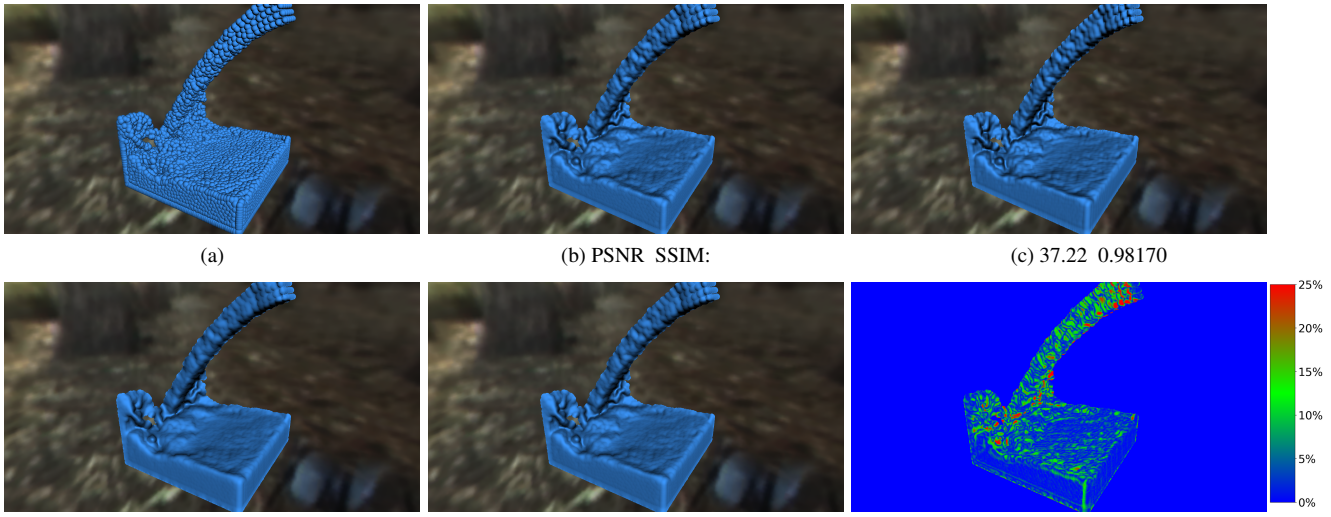
The figure illustrates that method (c), which solely relies on bilateral-filtered downsampling, has limitations in preserving edge clarity. This is because it cannot effectively reduce the contributions of coarse samples beyond the edges. In contrast, our method allows for the integration of a custom edge-stopping function in the upsampling process, resulting in improved edge preservation.

The À-trous wavelet method achieves high PSNR and SSIM values. Although it is generally close to the ground truth according to the metrics, it can introduce visual artifacts between some suspended tiles, as shown in Figure 9 (h). To address this, additional techniques such as the edge-tracing technique [MH22] for anti-aliasing may be required.

For an analytic solution based on our method, we employ the same weight settings in the DoF test, referred to as method (f). The PSNR and SSIM indicate that the error is larger compared to other methods, although there are no visible artifacts. However, after applying the trained weights, the quality is significantly improved.

Similar to the DoF test, we have also scripted a benchmark for 10K random frames. The average scores for our method are PSNR: 39.72 and SSIM: 0.99412, while the lowest scores are PSNR: 37.68 and SSIM: 0.98936. Besides, Figure 1 showcases the fully rendered result with temporal denoising. The supplementary video also demonstrates the dynamic results, confirming the stability of our method when combined with a general temporal denoiser.

Additionally, Figure 10 presents the results of diffuse denoising for 1-spp ray tracing. In method (f) for diffuse GI, we set $\gamma = 0$, and other weights are same to reflection denosing and DoF. It high-



(a) (b) PSNR SSIM: (c) 37.22 0.98170

(d) 33.96 0.97977 (e) 37.56 0.99069 (f)

Figure 11: *Normal reconstruction in real-time screen-space particle rendering: (a) normal reconstruction from the depth without filtering, (b) ground-truth reference, (c) approximation by separable-pass bilateral filter, (d) approximation by our weight template model without neural network training, (e) our full solution with neural network training, and (f) and the difference heap map from the ground truth to our result (please zoom in the 2K images).*

lights the effectiveness of our method for general diffuse global illumination (GI), with even better performance for denoising low-frequency GI, albeit requiring larger kernels. The metric analysis aligns with the discussion above regarding reflection GI. In the 10K-frame benchmark for diffuse denoising, the average scores for our method are PSNR: 47.79 and SSIM: 0.99687, while the lowest scores are PSNR: 45.85 and SSIM: 0.99223. As diffuse denoising operates on a low-frequency radiance field, the scores are generally higher than those for other use cases. Therefore, we halve the weight when calculating the total average scores for all tested applications.

Screen-Space Particle Rendering Figure 11 shows the representative screenshots that visualize the simply lit results of the screen-space particle rendering (SSPR) [vdLGS09]. We used an integer radius (pixel count in one direction excluding the center) of 24 (49×49 kernel) for the Gaussian domain function, resulting in a standard deviation of $\sigma \approx 16.67$ in Equation (10) based on the 3σ model. In game engines, this filter is typically approximated using a separable scheme. The PSNR and SSIM metrics indicate that our trained result improves upon the manually set solution and achieves a higher quality than the separable implementation. The final shaded result is displayed in Figure 1 and the attached video demo. In the 10K-frame benchmark, the average scores for our method are PSNR: 37.62 and SSIM: 0.99074, while the lowest scores are PSNR: 35.84 and SSIM: 0.98423.

Summary The heat map visualizations in Figure 8–11 indicate that the errors in our method are primarily concentrated on the edges when compared to the ground truth. Although the values on the edges differ, our method provides edge clarity that fulfills the intended application purpose.

5.2. Performance Evaluations

For performance verification, we measure the average time cost of the filtering process for each animated benchmark program (at least playing 1 minute) corresponding to the test scenes of Figures 8–11, using GPU time-stamp queries. The performance comparisons are presented in Table 2 to illustrate the results. We find that achieving the ground truth through strict bilateral filtering is hardly feasible for real-time rendering demands. In test cases with a resolution of 2560×1440 (2K), our method performs, on average, 10 times faster than the approximation achieved by separable-pass bilateral filters commonly used in practical applications.

Table 2: *Average time-cost statistics (in ms) of the each tested application using different methods in* 2560×1440 *(aka 2K) resolution (only filtering processes are measured). Other (method): HDE for DoF, and À-Trous-wavelet for denoising.*

Test case	Ground truth	Separable	Ours	Other
DoF	731.817	3.831	0.362	1.687
Denoiser	47.558	3.360	0.373	0.703
SSPR	34.543	3.186	0.286	NA

In terms of DoF performance, we compare our approach with the heat-diffusion equation (HDE) solver used in certain gaming applications. At a 2K resolution, our method shows an average speed improvement of 4 times compared to the HDE solver.

In our assessment of spatial filters for real-time GI denoising, we compare the performance of our method with the widely adopted À-Trous wavelet transforms used in the industry. In the case of

2K ray-traced images, our method demonstrates nearly double the speed compared to the wavelet method. Additionally, our method inherently prevents visual artifacts, reducing the need for additional effort in artifact handling during practical applications.

Moreover, we also test the DoF and denoising performance with the filtering radius varying from 1 to 64. In the DoF test, we use a uniform CoC value for each pixel and increase the CoC radius from 1 to 64. Similarly, in the denoising test, we use a uniform radius originally related to the roughness in the domain function and increase it. The statistics of the performance in frames per second (FPS) are shown in Figure 12. As the diagram indicates, our method maintains high performance in constant cost.

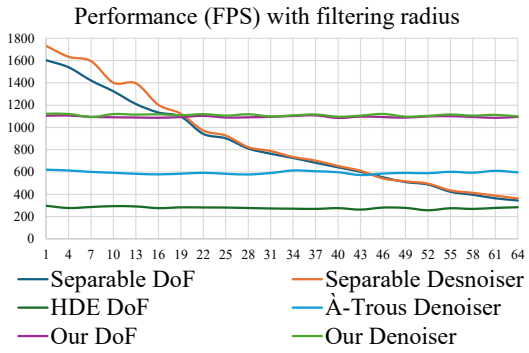


Figure 12: *Frame rates of different methods varying with the filtering radius (*1 *to* 64*) for DoF and denoising.*

Therefore, our method presents a promising solution for these significant real-time post-processing effects. Its high-performance capabilities effectively save computational resources, allowing for improved illuminations and simulations.

6. Conclusions

In this paper, we propose a novel approach to efficiently approximate edge-preserving filters for large kernels. Our method leverages upsampling on MIPs in a dual 3×3 space and incorporates machine learning techniques to achieve high-quality edge preservation. By employing our sampling pattern and weight model, the entire process can converge in an efficient downsampling and upsampling pipeline. Furthermore, we transform our pipeline and weight template model into a lightweight neural network, enabling the training of unknown weights using a ground-truth dataset to further enhance filtering quality. Our experimental results demonstrate that our method produces high-quality edge-preserving filtered results that closely resemble the corresponding ground truths in real-time high-performance scenarios. Additionally, our method is applicable to three significant use cases: DoF, denoising, and screen-space particle rendering, in a generalized manner. Moreover, our proposed solution is developer-friendly, as it is easy to implement on GPUs and exposes the same controlling parameters as the original bilateral filters.

One primary limitation of our method lies in the challenge of approximating domain functions that are not bell-shaped (Gaussian-like) kernels. In future work, our first priority is to expand the support for different types of domain functions. We also see significant potential for implementing our method on mobile GPUs, although we have not yet tested it on mobile devices due to limitations in our experimental setup. Additionally, we aim to explore and extend the applicability of our method to more practical use cases.

7. Acknowledgement

The authors would like to express their gratitude to all anonymous reviewers for their valuable comments, which significantly improved the manuscript and demonstrations, and thank Takahiro Harada and Karen Prairie for their helpful proofreading. This research was supported by NSFC (62332015, 62072449, and 62272298), and also by AMD Shanghai-Khronos3D.

References

[AA21] AHMED A. A., AHMED S.: A real-time car towing management system using ml-powered automatic number plate recognition. *Algorithms 14*, 11 (2021), 317. 2

[AMD20] AMD, INC.: *AMD FidelityFX Denoiser*, November 2020. https://github.com/GPUOpen-Effects/FidelityFX-Denoiser/tree/master/docs/. 7

[AS20] ALAIN M., SMOLIC A.: A spatio-angular binary descriptor for fast light field inter view matching. In *2020 IEEE International Conference on Image Processing (ICIP)* (Abu Dhabi, United Arab Emirates, 2020), IEEE, pp. 2636–2640. 2

[Bjo15] BJORGE M.: Bandwidth-efficient rendering. In *SIGGRAPH 2015 Xroads of Discovery* (New York, NY, USA, 2015), ACM, ACM. 2

[BK08] BARSKY B. A., KOSLOFF T. J.: Algorithms for rendering depth of field effects in computer graphics. In *Proceedings of the 12th WSEAS International Conference on Computers* (Stevens Point, Wisconsin, USA, 2008), ICCOMP'08, WSEAS, p. 999–1010. 2

[BYA15] BADRI H., YAHIA H., ABOUTAJDINE D.: Fast edge-aware processing via first order proximal approximation. *IEEE transactions on visualization and computer graphics 21*, 6 (2015), 743–755. 2

[Com86] COMOLLI J.-L.: Technique and ideology: Camera, perspective, depth of field [parts 3 and 4]. *Narrative, apparatus, ideology 24* (1986), 421–43. 2

[CPD07] CHEN J., PARIS S., DURAND F.: Real-time edge-aware image processing with the bilateral grid. *ACM Trans. on Graph. (TOG) 26*, 3 (2007), 103–es. 2

[CTY20] CHEN B.-H., TSENG Y.-S., YIN J.-L.: Gaussian-adaptive bilateral filter. *IEEE Signal Processing Letters 27* (2020), 1670–1674. 2

[CYS15] CHENG Z., YANG Q., SHENG B.: Deep colorization. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)* (USA, 2015), ICCV '15, IEEE Computer Society, p. 415–423. 2

[Dem04] DEMERS J.: Depth of field: A survey of techniques. In *GPU Gems* (Boston, MA, USA, 2004), Addison-Wesley Professional, p. u390. 2

[DHB24] DELIOT T., HEITZ E., BELCOUR L.: Transforming a non-differentiable rasterizer into a differentiable one with stochastic gradient estimation. *Proc. ACM Comput. Graph. Interact. Tech. 7*, 1 (may 2024). URL: https://doi.org/10.1145/3651298, doi:10.1145/3651298. 6

[DSHL10] DAMMERTZ H., SEWTZ D., HANIKA J., LENSCH H. P. A.: Edge-avoiding À-trous wavelet transform for fast global illumination filtering. In *Proceedings of the Conference on High Performance Graphics* (Goslar, DEU, 2010), HPG '10, Eurographics Association, pp. 67–75. 2, 9

[Fat09] FATTAL R.: Edge-avoiding wavelets and their applications. *ACM Trans. Graph. 28*, 3 (jul 2009). 2

[GCB*17] GHARBI M., CHEN J., BARRON J. T., HASINOFF S. W., DURAND F.: Deep bilateral learning for real-time image enhancement. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 1–12. 3

[GO11] GASTAL E. S. L., OLIVEIRA M. M.: Domain transform for edge-aware image and video processing. *ACM Trans. Graph. 30*, 4 (jul 2011). 2

[Gru11] GRUEN H.: An optimized diffusion depth of field solver (ddof). In *Game Developers Conference 2011* (San Francisco, CA, USA, 2011), AMD. 2, 7

[Ham08] HAMMON E. J.: Practical post-process depth of field. In *GPU Gems 3* (Boston, MA, USA, 2008), Addison-Wesley Professional, p. u390. 2

[HDL11] HANIKA J., DAMMERTZ H., LENSCH H.: Edge-optimized À-trous wavelets for local contrast enhancement with robust denoising. *Computer Graphics Forum 30*, 30 (2011), 1879–1886. 2, 9

[JAFF16] JOHNSON J., ALAHI A., FEI-FEI L.: Perceptual losses for real-time style transfer and super-resolution, April 2016. cite arxiv:1603.08155. 6

[JB11] JANSEN J., BAVOIL L.: *Fast rendering of opacity mapped particles using DirectX 11 tessellation and mixed resolutions*. Tech. rep., NVIDIA, 2011. 5

[Kaw03] KAWASE M.: Frame buffer postprocessing effects in double-s.t.e.a.l (wreckless). In *Game Developers Conference 2003* (San Francisco, CA, USA, 2003), BUNKASHA PUBLISHING CO.,LTD. 2

[KLB13] KANG X., LI S., BENEDIKTSSON J. A.: Feature extraction of hyperspectral images with image fusion and recursive filtering. *IEEE Transactions on Geoscience and Remote Sensing 52*, 6 (2013), 3742–3752. 2

[LDOD06] LEFOHN A., DAVIS U. C., OWENS J., DAVIS U. C.: *Interactive depth of field using simulated diffusion*. Tech. rep., Pixar, 2006. 2

[LFY*21] LIU C., FENG Y., YANG C., WEI M., WANG J.: Multi-scale selective image texture smoothing via intuitive single clicks. *Signal Processing: Image Communication 97* (2021), 116357. 3

[MH22] MEUNIER S., HARADA T.: *Weighted À-Trous Linear Regression (WALR) for Real-Time Diffuse Indirect Lighting Denoising*. Tech. rep., Advanced Micro Devices, Dec. 2022. 3, 9

[MPS19] MOYNIHAN M., PAGÉS R., SMOLIC A.: Spatio-temporal upsampling for free viewpoint video point clouds. In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISIGRAPP 2019, Volume 5: VISAPP* (Prague, Czech Republic, 2019), Trémeau A., Farinella G. M., Braz J., (Eds.), SciTePress, pp. 684–692. 2

[NKZN08] NAGAHARA H., KUTHIRUMMAL S., ZHOU C., NAYAR S. K.: Flexible depth of field photography. In *Computer Vision – ECCV 2008* (Berlin, Heidelberg, 2008), Forsyth D., Torr P., Zisserman A., (Eds.), Springer Berlin Heidelberg, pp. 60–73. 2

[PHK11] PARIS S., HASINOFF S. W., KAUTZ J.: Local laplacian filters: edge-aware image processing with a laplacian pyramid. *ACM Trans. Graph. 30*, 4 (2011), 68. 2

[RDS*15] RUSSAKOVSKY O., DENG J., SU H., KRAUSE J., SATHEESH S., MA S., HUANG Z., KARPATHY A., KHOSLA A., BERNSTEIN M., BERG A. C., FEI-FEI L.: ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV) 115*, 3 (2015), 211–252. 6

[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* (New York, NY, USA, 2017), HPG '17, Association for Computing Machinery. 7

[SR19] SHAHDOOSTI H. R., RAHEMI Z.: Edge-preserving image denoising using a deep convolutional neural network. *Signal Processing 159* (2019), 20–32. 3

[SSC*17] SCHAFFNER M., SCHEIDEGGER F., CAVIGELLI L., KAESLIN H., BENINI L., SMOLIC A.: Towards edge-aware spatio-temporal filtering in real-time. *IEEE Transactions on Image Processing 27*, 1 (2017), 265–280. 2

[SSD*09] SOLER C., SUBR K., DURAND F., HOLZSCHUCH N., SILLION F.: Fourier depth of field. *ACM Transactions on Graphics (TOG) 28*, 2 (2009), 1–12. 2

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. In *Proceedings of the Sixth International Conference on Computer Vision* (USA, 1998), ICCV '98, IEEE Computer Society, p. 839. 3

[vdLGS09] VAN DER LAAN W.-J., GREEN S., SAINZ M.: Screen space fluid rendering with curvature flow. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2009), I3D '09, ACM, pp. 91–98. 1, 3, 7, 10

[WYS*12] WU S., YU K., SHENG B., HUANG F., GAO F., MA L.: Accurate depth-of-field rendering using adaptive bilateral depth filtering. In *Proceedings of the First International Conference on Computational Visual Media* (Berlin, Heidelberg, 2012), CVM'12, Springer-Verlag, p. 258–265. URL: https://doi.org/10.1007/978-3-642-34263-9_33, doi:10.1007/978-3-642-34263-9_33. 6

[WYS*20] WANG Z., YE X., SUN B., YANG J., XU R., LI H.: Depth upsampling based on deep edge-aware learning. *Pattern Recognition 103* (2020), 107274. 3

[XRW19] XU T., REN X., WU E.: The power of box filters: Real-time approximation to large convolution kernel by box-filtered image pyramid. In *SIGGRAPH Asia 2019 Technical Briefs* (New York, NY, USA, 2019), SA '19, Association for Computing Machinery, p. 1–4. 2, 3, 6, 7, 9

[XRY*15] XU L., REN J. S. J., YAN Q., LIAO R., JIA J.: Deep edge-aware filters. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37* (Lille, France, 2015), ICML'15, JMLR.org, p. 1669–1678. 3

[ZCO10] ZHANG Y., COHEN J., OWENS J. D.: Fast tridiagonal solvers on the gpu. *SIGPLAN Not. 45*, 5 (Jan. 2010), 127–136. 2

[ZLJ*19] ZHU F., LIANG Z., JIA X., ZHANG L., YU Y.: A benchmark for edge-preserving image smoothing. *IEEE Transactions on Image Processing 28*, 7 (2019), 3556–3570. 3