Kernel Predicting Neural Shadow Maps

XUEJUN HU, Tsinghua University, China
JINFAN LU, Tsinghua University, China
KUN XU*, Tsinghua University, China

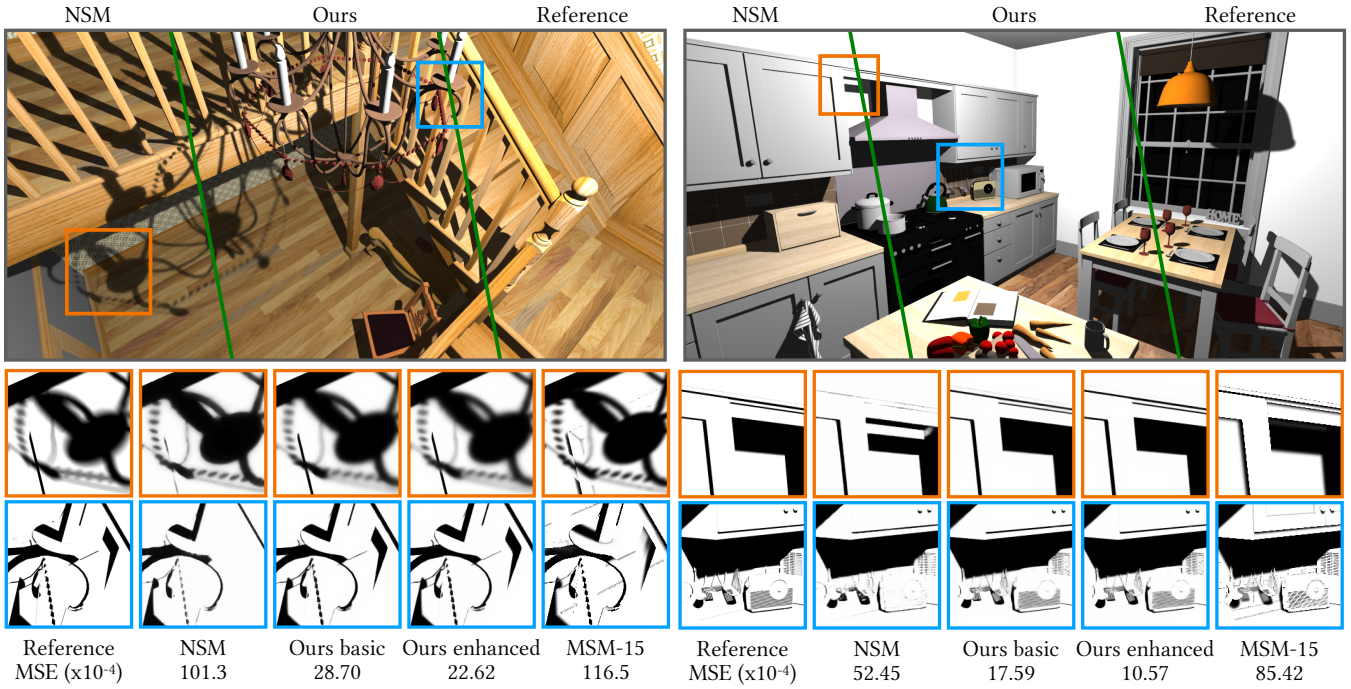

Fig. 1. Visual and numerical comparisons with state-of-the-art shadow mapping methods, including Neural Shadow Mapping (*NSM*) [Datta et al. 2022] and Moment Shadow Mapping (*MSM-15*) [Peters and Klein 2015]. We present two versions of our method (*ours-basic* and *ours-enhanced*), both consistently outperforming previous state-of-the-art methods in both visual quality and numerical measures.

Existing neural shadow mapping methods [Datta et al. 2022] have shown to be promising in generating high quality soft shadows. However, it demonstrates limited generalizability to new scenes. In this paper, we present a novel neural method, named *kernel predicting neural shadow mapping* to address this issue. Specifically, we explicitly model soft shadow values as pixelwise local filtering from nearby base shadow values (i.e., the classic hard shadow values) in the screen space, where the local filter weights are predicted through a trained neural network. We use dilated filters as the representation of our local filters to maintain a balance between computational efficiency and receptive field of a local filter. We further enhance shadow quality by replacing the classic shadow map algorithm [Williams 1978] with moment shadow maps [Peters and Klein 2015] to generate the base shadows values. With carefully designed filters, input features, and loss functions with temporal regularization, our method runs in real-time framerates (i.e., >100 fps for 2048 × 1024 resolution), produces temporally-stable soft shadows with good generalizability, and consistently beats state-of-the-art methods in both visual qualities and numeric measures. Code and model weights are available at https://github.com/Hoosus/KPNSM.

CCS Concepts: • **Computing methodologies** → **Visibility**; *Machine learning*; *Rasterization*.

Additional Key Words and Phrases: Shadow Mapping, Kernel Prediction, Neural Networks

ACM Reference Format:
Xuejun Hu, Jinfan Lu, and Kun Xu. 2025. Kernel Predicting Neural Shadow Maps. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25), August 10–14, 2025, Vancouver, BC, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3721238.3730645

*Kun Xu is the corresponding author.

Authors' Contact Information: Xuejun Hu, Tsinghua University, Beijing, China, huxj23@mails.tsinghua.edu.cn; Jinfan Lu, Tsinghua University, Beijing, China, lujf22@mails.tsinghua.edu.cn; Kun Xu, Tsinghua University, Beijing, China, xukun@tsinghua.edu.cn.

This work is licensed under a Creative Commons Attribution 4.0 International License.
SIGGRAPH Conference Papers '25, Vancouver, BC, Canada
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1540-2/25/08
https://doi.org/10.1145/3721238.3730645

1 Introduction

Shadows augment rendering with crucial lighting and spatial cues. As tracing shadow rays can be expensive, modern graphics involves shadow mapping as an essential category of techniques for fast and realistic shadows. Long since its establishment [Williams 1978], shadow mapping received improvements for anti-aliasing [Reeves et al. 1987], soft shadows [Fernando 2005], efficient prefiltering [Annen et al. 2007, 2008b; Donnelly and Lauritzen 2006; Peters and Klein 2015] and shadow map cascading [Engel 2006; Zhang et al. 2006]. While staying intuitive and lightweight, these methods rely on heuristic parameters and can easily cause artifacts such as limited contact hardening or light leaking.

Recently, Datta et al. [2022] proposed *neural shadow mapping* (NSM) that directly predicts soft shadows from a few carefully designed screen-space features (i.e., G-buffers). Input features are selected from feature combinations of view-space or emitter-space depths, directions, normals, their dot-products, and emitter size. To achieve efficient inference for real-time applications, a simplified U-Net structure [Ronneberger et al. 2015] is used and is trained through pairs of pre-generated screen-space features and ground truth shadows. While it is able to produce more accurate shadows than classic shadow map methods in complex scenes, its limited generalization capability significantly restricts its practicality. In particular, a dedicated network must be trained for each individual scene.

To address this problem, we propose a novel shadow generation method named *kernel predicting neural shadow mapping*. Our method is based on the assumption that soft shadow values can be approximated as a weighted average of base shadow values (i.e., hard shadows) of its screen-space neighboring pixels [MohammadBagher et al. 2010; Zheng and Saito 2011]. Instead of directly predicting soft shadow values as done in NSM [Datta et al. 2022], we predict screen-space kernel weights and then compute the soft shadow values through filtering using the predicted pixelwise kernel weights. To maintain a balance between computational efficiency and representation capacity, we choose to use dilated filters [Dammertz et al. 2010] as the representation of our filtering kernels. Kernel predicting can provide better regularization than predicting values, leading to less approximation errors and better generalizability [Bako et al. 2017].

We train a U-Net based network to predict the locally optimal kernel weights. We generally follow NSM [Datta et al. 2022] to design the input screen-space features, network structures, and loss functions to enforce a real-time inference speed, with a few modifications that could further improve shadow quality and temporal consistency. In addition, by combining our method with *moment shadow maps* (MSM) [Peters and Klein 2015], i.e., applying screen-space filtering on MSM generated shadows instead of on hard shadows, the shadow quality can be further improved with very small overhead. Our method is temporally stable, robust in handling unseen scenes, and runs in real-time. Experiments demonstrate that its rendering quality is superior to state-of-the-art classic and neural shadow map methods [Datta et al. 2022; Peters and Klein 2015] in terms of both visual qualities and quantitative measures.

2 Related Works

The original shadow mapping method [Williams 1978] contains two rendering passes. The first pass renders the scene from the light's perspective, and generates an emitter-space depth map which stores the closest depth value as seen from the light. In the second pass, we render the scene as usual and determine whether each pixel is in shadow by testing whether the queried depth (i.e., its distance from the rendered pixel to the light) is larger than the stored depth. As the most widely used real-time shadow rendering method, shadow mapping is able to generate hard shadows under point and directional lights in a very efficient way.

Filtering-based methods filter depth-based visibility to reduce aliasing. *Percentage closer filtering* (PCF) [Reeves et al. 1987] samples the depths within a fixed-size neighborhood window around the current pixel in the shadow map and computes the averaged shadow test, i.e., the percentage of sampled depths which are smaller than the queried depth. Another class of methods are based on prefiltered depth statistics, including *variance shadow maps* [Donnelly and Lauritzen 2006], *convolution shadow maps* [Annen et al. 2007], *exponential shadow maps* [Annen et al. 2008b], and *moment shadow maps* (MSM) [Peters and Klein 2015]. Instead of heavily sampling on the depth map, those methods rely on depth statistics to compute an approximation of the averaged shadow test. Thanks to the use of hardware texture operations like mipmapping and anisotropic filtering, those computations can be made very efficient which mostly require only one texture lookup.

Dozens of works of shadow map variants have been proposed to approximately render soft shadows. *Percentage closer soft shadows* (PCSS) [Fernando 2005] extends PCF by automatically determining the filtering kernel size through blocker search and penumbra size estimation. By incorporating blocker search and acceleration structures such as mipmapping or *summed-area table* (SAT) for real-time area queries, pre-filtering methods have also been extended to soft shadows [Annen et al. 2008a; Dong and Yang 2010; Peters et al. 2016].

While the main stream of shadow map variants perform filtering in the emitter space, some opt for filtering in the screen space [MohammadBagher et al. 2010; Robison and Shirley 2009; Zheng and Saito 2011]. The screen space methods can generate soft shadows of similar quality as the emitter space PCSS method, and can better utilize image-space parallelization.

Note that all those shadow map variants are proposed before the era of deep learning. The used filtering kernels and their sizes are manually designed (or derived) through simple geometric assumptions between emitter, blocker, and receiver, i.e., they are usually assumed to be parallel. However, such assumptions can hardly be satisfied in real complex scenes, leading to inaccurate shadows.

Recently, Datta et al. [2022] proposed the first neural method for soft shadow generation, named *neural shadow mapping* (NSM). They directly predict soft shadows from a few carefully designed screen-space features (i.e., G-buffers), which are selected from feature combinations of view-space or emitter-space depths, directions, normals, dot-products, and emitter size. While it is able to produce more accurate shadows than classic shadow map methods in complex scenes, it has limited capacity of generalization to unseen

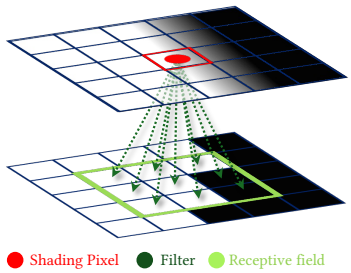

Fig. 2. **Screen space shadow filtering.** We assume that soft shadow value at every pixel can be well approximated by a local filtering in screen space on the hard shadow values.

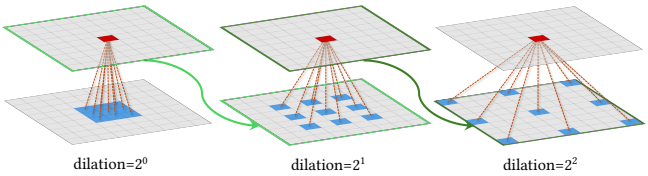

Fig. 3. **Illustration of dilated filters.** A dilated filter [Dammertz et al. 2010] is a multi-pass filter with doubling strides, which provides large receptive field with a small number of parameters.

scenes. In contrast, while our method is also a machine-learning based approach for computing shadows, we predict filtering kernels instead of shadow values. The design of filter kernel prediction provides us with the following advantages. First, the filtered shadow values will be bounded by the values of nearby hard shadows by definition, improving the stability of training. Second, our method leads to smaller approximation errors, as demonstrated by the experiments. Third, our method has better generalizability so that it performs better when being trained on multiple scenes or when being tested on unseen scenes.

3 Screen-Space Shadow Filtering

The goal of our method is to compute approximated soft shadows from an area light source in an efficient way. As illustrated in Fig. 2, our basic assumption is that the soft shadow value e_i at every pixel i in screen space can be well approximated by a local filtering on the hard shadow values generated by the classic shadow mapping algorithm [Williams 1978]:

$$e_i = \sum_{j \in N(i)} w_{ij} s_j, \tag{1}$$

where j iterates over all pixels in the local neighborhood $N(i)$, s_j denotes the binary hard shadow value, and w_{ij} denotes the normalized filtering weights, i.e., satisfying $\sum_j w_{ij} = 1$.

We opt for filtering shadows in the screen space instead of in the emitter space. This is because the crucial steps in our neural network computations, i.e., image (layer) convolutions, would be more natural and efficient to be performed in the screen space, avoiding the need of complicated coordinate transformations between the two spaces. Note that the idea of screen space filtering has already been explored by existing works on screen-space percentage-closer soft shadows [MohammadBagher et al. 2010; Zheng and Saito 2011]. Different from those works which utilize manually designed pixel-wise anisotropic bilateral filters, we allow arbitrary kernel weights which are predicted through supervised training and can be much closer to the optimal values.

Dilated Filters. Now we discuss the appropriate representation of our shadow filters. On the one hand, the representation of the filter should be as compact as possible in order to restrict the computation cost within a reasonable range. On the other hand, the receptive field of the filter (or the size of the filter) should be as large as possible in order to scale well for shadows with a large penumbra.

To make a balance between the compactness and capability of the shadow filters, we resort to dilated filters [Dammertz et al. 2010] as our representation. Dilated filters, also known as À-trous filters, are composed of a series of small kernels, applied sequentially with doubling strides. This multi-pass design ensures coverage of a large filter region, while keeping a small computational budget with only a few small filters per pixel. Please see an illustration in Fig. 3.

In our method, we use 4 5-by-5 kernels with dilations of 1, 2 and 4 and 8, respectively. It achieves a square receptive field of 61-by-61 pixels. In total, each pixelwise dilated filter contains $4 \times 5 \times 5 = 100$ adjustable parameters. In contrast, a naive dense filter with the same receptive field contains a much larger amount of parameters, i.e., $61 \times 61 = 3721$ parameters.

4 Method

As illustrated in Fig 4, our rendering method consists of 4 steps:

(1) **Pre-processing**. Following the classic shadow mapping algorithm [Williams 1978], we render the scene from the light's perspective and record the emitter-space depth map in the first pass, and then render the scene from the viewpoint to record the hard shadow image in the second pass. We refer to the generated hard shadow image as the *base shadow image*, which will be used for filtering later.
(2) **Input features gathering.** During rendering in the second pass, besides recording the hard shadow image, we will also collect a few screen-space feature maps as G-buffers, which will be later used as inputs to the neural network.
(3) **Kernel Prediction.** After that, we feed the screen-space features into our kernel predicting network which is composed of a modified U-Net [Ronneberger et al. 2015] to predict pixel-wise dilated filters.
(4) **Post-processing filtering.** Finally, we obtain soft shadow values by filtering the hard shadow image with the predicted dilate filters in the screen-space using Eq. 1.

The above 4 steps would be performed one time for each light source if we have multiple light sources.

In the rest of this section, we will introduce the details of the input features in Section 4.1, and describe the network structure and the loss function to train the network in Section 4.2. In section 4.3, we will also show that we could use more sophisticated shadow mapping variants such as moment shadow maps [Peters and Klein

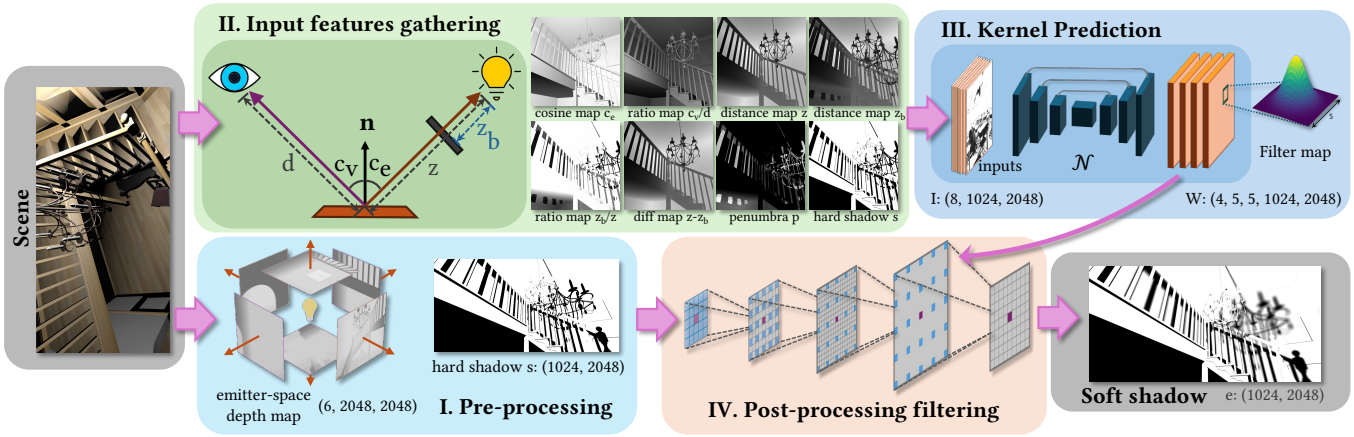

Fig. 4. Pipeline of our method.

2015] to generate the base shadow to enhance the shadow quality of final results.

4.1 Input features gathering

Generally, we follow NSM [Datta et al. 2022] to construct our inputs to the network with a few modifications. Specifically, similar to NSM, we also include the following 4 feature maps:

- a cosine feature map c_e, which are dot products of receiver normals and emitter directions;
- a ratio map c_v/d, which are dot products of receiver normals and viewing directions, divided by the distance from the viewpoint to the receiver point;
- a ratio map z_b/z and a difference map $z - z_b$, where z and z_b are distances from the emitter to the receiver and to the blocker, respectively.

Besides, we also include 4 additional feature maps as input:

- the two depth maps z (distance from the emitter to the receiver) and z_b (distance from the emitter to the blocker);
- Since the size of penumbra could be a useful hint for the size of the filter, we separately create a penumbra width map p that are estimated from light radius and depth information and use it as a input feature map. This is different from NSM which adds the light radius to the cosine feature map c_e.
- the rendered hard shadow image s.

In total, the input to our network contains all the above 8 feature maps, i.e., $I = \{c_e, c_v/d, z, z_b, z_b/z, z - z_b, p, s\}$. All the input feature maps can be assembled in the second pass which renders the scene from the viewpoint. All the features can be obtained by simple computations and queries with respect to depths, normals, and viewing and emitter directions. Below, we explain how to compute the estimated penumbra width map p.

By assuming that the blocker, the receiver and the light source are parallel, we follow PCSS [Fernando 2005] to estimate the emitter-space penumbra width as:

$$p_{pcss} = \frac{z - z_b}{z_b} \cdot R_e, \tag{2}$$

where z and z_b are distances from the emitter to the receiver and to the blocker, respectively, and R_e is the light radius. By transforming p_{pcss} to the screen-space, i.e., by first projecting the penumbra area to the local tangent plane and then projecting it to screen-space, we obtain the screen-space penumbra width as:

$$p = \sqrt{\left|\frac{c_v}{c_e}\right|} \cdot p_{pcss} = \sqrt{\left|\frac{c_v}{c_e}\right|} \cdot \frac{z - z_b}{z_b} \cdot R_e. \tag{3}$$

Note that this input is only non-zero in shadowed regions.

4.2 Kernel predicting networks

Similar to NSM, we use a modified U-Net [Ronneberger et al. 2015] $\mathcal{N}$ to predict our dilated shadow filters. The kernel predicting process can be formulated as:

$$W = \mathcal{N}(I), \tag{4}$$

where I are the input features defined in Sec. 4.1, and $W = \{w_{ij}\}$ denote the predicted filter weight map.

4.2.1 Compact network structure. Our U-Net structure has 4 downsampling layers. Each downsampling layer has 64, 64, 128, and 256 channels after it, respectively. The features are downsampled with maxpooling and upsampled using the transposed convolutions. We change skip connections from concatenation to addition to reduce memory consumption and computation. To further improve its performance on high-resolution inputs, we replace the outer downsampling and upsampling layers with pixel-shuffle downsampling and bilinear upsampling layers, respectively. Since the kernel prediction is smooth, this brings us considerable speedup with only little amount of quality loss. In total, our network has 1.81M trainable parameters.

4.2.2 Loss functions. Following NSM [Datta et al. 2022], to encourage visually convincing and temporally consistent shadow, our training loss is defined as a weighted combination of an L1 loss $\mathcal{L}_1$, a VGG loss $\mathcal{L}_{VGG}$ and a temporal loss $\mathcal{L}_t$:

$$\mathcal{L} = \mathcal{L}_1 + \beta_v \mathcal{L}_{VGG} + \beta_t \mathcal{L}_t(V, V'), \tag{5}$$

where the balancing weights are empirically set as $\beta_v = 0.1$ and $\beta_t = 1.0$, respectively. The L1 loss measures the L1 difference between our produced soft shadow image and the ground truth image, and the VGG loss measures the L2 difference between their VGG feature maps. The temporal loss is included to reduce flickering when moving the camera or the light. Different from NSM's temporal loss which directly measures pixelwise differences between current view and perturbed view, we align the two views through a warping $\mathcal{W}$ derived from the motion vector before computing their differences:

$$\mathcal{L}_t(V, V') = \mathcal{L}_1(V, \mathcal{W}(V')) + \beta_v \mathcal{L}_{VGG}(V, \mathcal{W}(V')), \quad (6)$$

where V' denotes a perturbed view. We align the two views with motion vector $\mathcal{W}$ and mask out pixels that fail the reprojection tests.

4.3 Enhancement with Moment Shadow Map

Up to this point, we have used the hard shadow image generated by the classic shadow mapping algorithm [Williams 1978] as our base shadow image and compute our soft shadows by filtering on it. In fact, instead of using the hard shadow image, we could filter on higher quality shadow images generated by some more sophisticated shadow mapping variants, which we refer to as *base shadow generator*. We have selected moment shadow maps (MSM) [Peters and Klein 2015] with bilinear filtering as our base shadow generator. This is because MSM produces soft shadows with the best quality among those filtering based shadow map variants and only incurs a little overhead.

Our rendering pipeline only needs a few minor modifications in the pre-processing step and the post-processing filtering step. The input features gathering step and the kernel prediction step are not changed. In particular, in the pre-processing step, we run MSM [Peters and Klein 2015] instead of the classic shadow mapping algorithm, and we record the shadow image generated by MSM instead of the hard shadow image as the base shadow image. Note that the input feature maps in the second step will also use this new base shadow image instead of the hard shadow image. In the post-processing filtering step, similarly, we simply filter on the new base shadow image instead of filtering on the hard shadow image to compute the final soft shadow outputs.

Results demonstrate that this enhancement strategy with MSM is rather effective, leading to significantly lower errors. We believe it is probably due to two reasons. First, since MSM has utilized more emitter-space information while we mainly make use of screen space features, combining them is probably helpful. Second, the shadow results of MSM is smoother and has less artifact than the hard shadow image generated by classic shadow map, thus it is a better choice as a filtering input.

Note that this is another key difference between our method and NSM [Datta et al. 2022]. Thanks to our kernel predicting scheme, our method can be combined with any sophisticated and efficient shadow generator to further enhance the shadow quality. In contrast, NSM uses a value predicting scheme and does not hold this desired property.

In our implementation, we always use an MSM implementation with 4 order depth moments and bilinear interpolation as our base shadow generator.

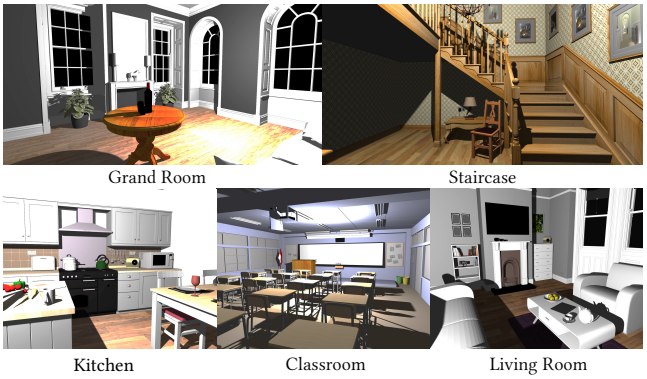

Fig. 5. Preview of our training scenes.

5 Dataset and Implementation

Dataset. We have gathered a dataset of 5 scenes from the rendering resources website [Bitterli 2016] for training. Fig. 5 shows the 5 scenes.

We generate 6000 rendering settings for each scene. Each rendering setting contains a randomly sampled camera position, a randomly sampled light position, and a randomly selected light radius. The camera position and the light position are not allowed to be too close to scene geometries in order to avoid degenerated cases. For each rendering setting, we collect a pair of input feature maps and a ground truth soft shadow image. As in NSM [Datta et al. 2022], we generate ground truth shadows using hardware ray tracing and screen-space multi-sampling anti-aliasing (MSAA) while leaving the inputs aliased, letting the network learn to produce anti-aliased outputs. For each rendering setting, we also generate a perturbed rendering setting where the positions of the camera and the light are both slightly changed. The perturbed setting is used to compute the temporal loss.

Implementation. Our method is implemented with a mixed use of Falcor renderer [Kallweit et al. 2022], PyTorch, and TensorRT[1]. Specifically, the rendering part is implemented inside the Falcor renderer. In the first rendering pass when we render the scene from the light's perspective, we use cubemaps for generating the emitter-space depth maps to ensure coverage of the entire scene. We also generate extra depth moment textures and perform Gaussian blur on them when moment shadow maps are selected as the base shadow generator. In the second rendering pass when we render the scene from the viewpoint, all input feature maps are computed through shaders and stored in GPU buffers, which will be fed into the kernel predicting network later.

Our kernel predicting network is trained using PyTorch with Adam optimizer [Kingma and Ba 2017]. The learning rate is initially set to 10^{-3}, and is reduced by 10x after 20000 and 40000 iterations. After training, we export the models to TensorRT for accelerated inference. Besides, we use half precision during inference for further acceleration. During training, the dilated filters are implemented using the *unfold* operations in Pytorch. During

[1] https://developer.nvidia.com/tensorrt

inference, we implement dilated filtering with compute shaders for optimal performance.

6 Results and Ablations

We test our method on a PC equipped with an AMD Ryzen 9 7950X CPU and an NVIDIA RTX 4090 GPU. All scenes are always rendered with a resolution of 2048×1024. The emitter-space depth maps (i.e., shadow maps) always use a resolution of $6 \times 2048 \times 2048$.

6.1 Comparisons and Results

We validate two versions of our method: a basic version that uses the classic shadow mapping as the base shadow generator, which we refer to as *ours-basic*, and an enhanced version that uses moment shadow map as the base shadow generator, which we refer to as *ours-enhanced*.

We compare our method against three state-of-the-art works, including neural shadow mapping (NSM) [Datta et al. 2022], moment shadow maps (MSM) [Peters and Klein 2015] and percentage-closer soft shadows (PCSS) [Fernando 2005]. Since the official implementation of NSM is unavailable, we reimplement it by ourselves. For fair comparisons, we employ a larger multi-layer decoder in the NSM implementation to ensure it has an equal or larger number of parameters than ours. We have also provided two versions of NSMs: *NSM-joint* which is trained using all 5 training scenes, and *NSM-single* which is trained separately using each scene. In contrast, both versions of our method are trained using all 5 scenes. For moment shadow maps being compared, we use a parameter setting of order 4, a 15×15 filtering window and a fixed $\sigma = 3.0$. Note that the MSM that serves as the base shadow generator of our method uses a more simplified setting. For PCSS, we use 64 samples for blocker search and 128 samples for shadow estimation. A collection of visual comparisons are presented in Figure 6. Additionally, we report the averaged statistics over all tested scenes, including rendering errors measured in MSE and SSIM, along with the running time in Table 1.

From the results we observe that both versions of our method consistently outperform other methods across all metrics, producing accurate hard and soft shadows without noticeable artifacts. In contrast, NSM exhibits incorrect predictions in certain regions and suffers from aliasing artifacts near short seams. MSM suffers from improper blurring and light leakage, while PCSS is prone to incorrect blurring and aliasing. Besides nice rendering quality, our method runs in real-time, i.e., achieving >100 fps for a rendering resolution of 2048×1024. For rendering a frame of a typical indoor scene, ours-enhanced takes 0.9ms for two rasterization passes, 6.4ms for network inference and 1.6ms for post-processing filtering, which is 8.9ms in total.

To demonstrate the generalizability of our method, we test our method on five unseen scenes. We provide visual results and quantitative error numbers in Fig. 7. In general, our method generates the best results in terms of both visual quality and error numbers. In contrast, NSM and MSM exhibit more frequent incorrect shadows or artifacts. Take the pink room scene as example (Fig. 7 top), NSM produces artifacts in the top-right corner and MSM generates overblurred shadows on the shelf, while our method is able to deal with both cases well. Our superior generalizability comes from extra regularization from the kernel prediction strategy, as the final soft shadows are always bounded by local hard shadows. The kernel prediction strategy is also less sensitive to domain gap than direct prediction of final values.

Furthermore, we compare our method with hardware ray tracing. Two variants of hardware ray tracing are compared, including one without denoising and one with denoising (i.e., SVGF) [Schied et al. 2017]. The comparison is performed in an equal-time setting and the results are given in Fig. 8. We also provide numerical error numbers measured in MSE and *Perception-based Image Quality Evaluator (PIQE)* [N et al. 2015]. Generally, our results have the best visual quality. Although the ray-traced results have lower MSE error numbers, they are visually and perceptually worse due to noticeable noises and visible temporal flickering. While SVGF reduces ray tracing noises effectively, it introduces additional biases and blurring, resulting in lower quality than our method across both metrics.

Table 1. Statistics of our method and other comparing methods. We report averaged rendering errors measured in MSE and SSIM of the 5 scenes in the training dataset for all methods. The averaged running time for rendering a single frame is also reported.

	Ours basic	Ours enhanced	NSM single	NSM joint	MSM-15	PCSS
MSE($\times 10^{-4}$)	14.59	**9.67**	45.88	64.38	65.49	28.58
SSIM	0.973	**0.979**	0.906	0.915	0.892	0.960
time	8.8ms	8.9ms	5.0ms		3.7ms	9.0ms

Table 2. Ablations on varying number of layers and varying number of filters in a dilated filter. We also provide the size of the receptive field for each configuration of the dilated filter.

MSE ($\times 10^{-4}$)		number of filters to receptive field		
		3 to 29	4 to 61	5 to 125
number of layers	3	10.24	9.72	10.76
	4	10.10	**9.67**	9.94
	5	10.19	10.20	9.81

Table 3. Ablations on the temporal loss.

Temporal error ($\times 10^{-2}$)	Ours basic	Ours enhanced
w/ temporal loss	**3.72**	**3.68**
w/o temporal loss	4.01	3.91

6.2 Analysis and Ablation Studies

6.2.1 Varying sizes of light sources. In Fig. 9, we further test our method in rendering the same scene with varying sizes of light sources. Our method generates more accurate and visually convincing shadows across different emitter sizes with smooth transitions, whereas NSM produces perceptually inferior results with much larger errors. Again, the results verify the superior robustness of our kernel prediction over NSM's value prediction.

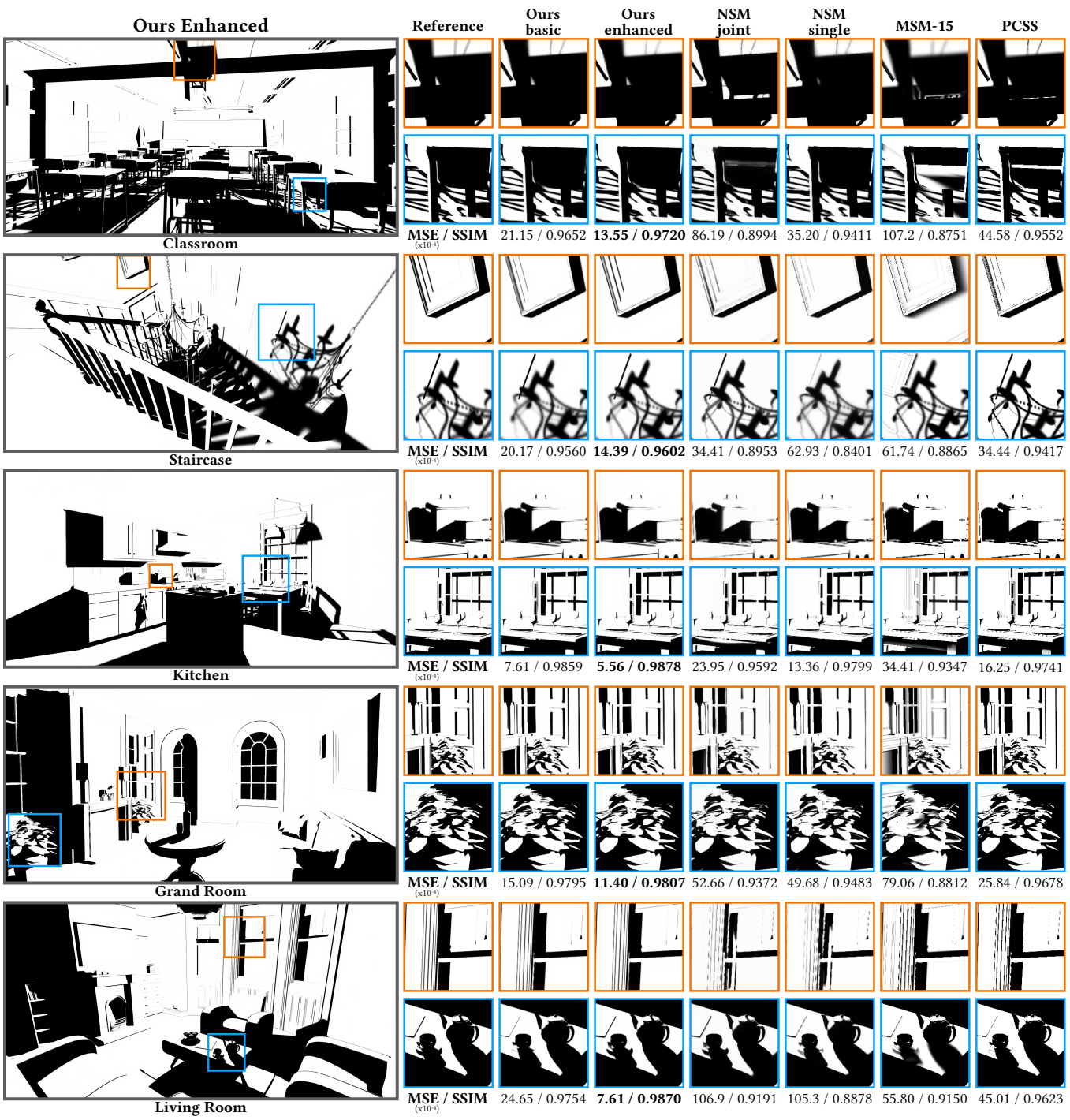

Fig. 6. Comparing our method with other state-of-the-art techniques. *Ours-basic* and *Ours-enhanced* are variants of our models that use either classic shadow mapping or moment shadow map as the base shadow generator. *NSM-joint* and *NSM-single* are variants of Neural Shadow Mapping [Datta et al. 2022] trained on all five scenes or on a single scene. MSM-15 is 4-order moment shadow map with 15×15 prefiltering kernel and $\sigma = 3.0$.

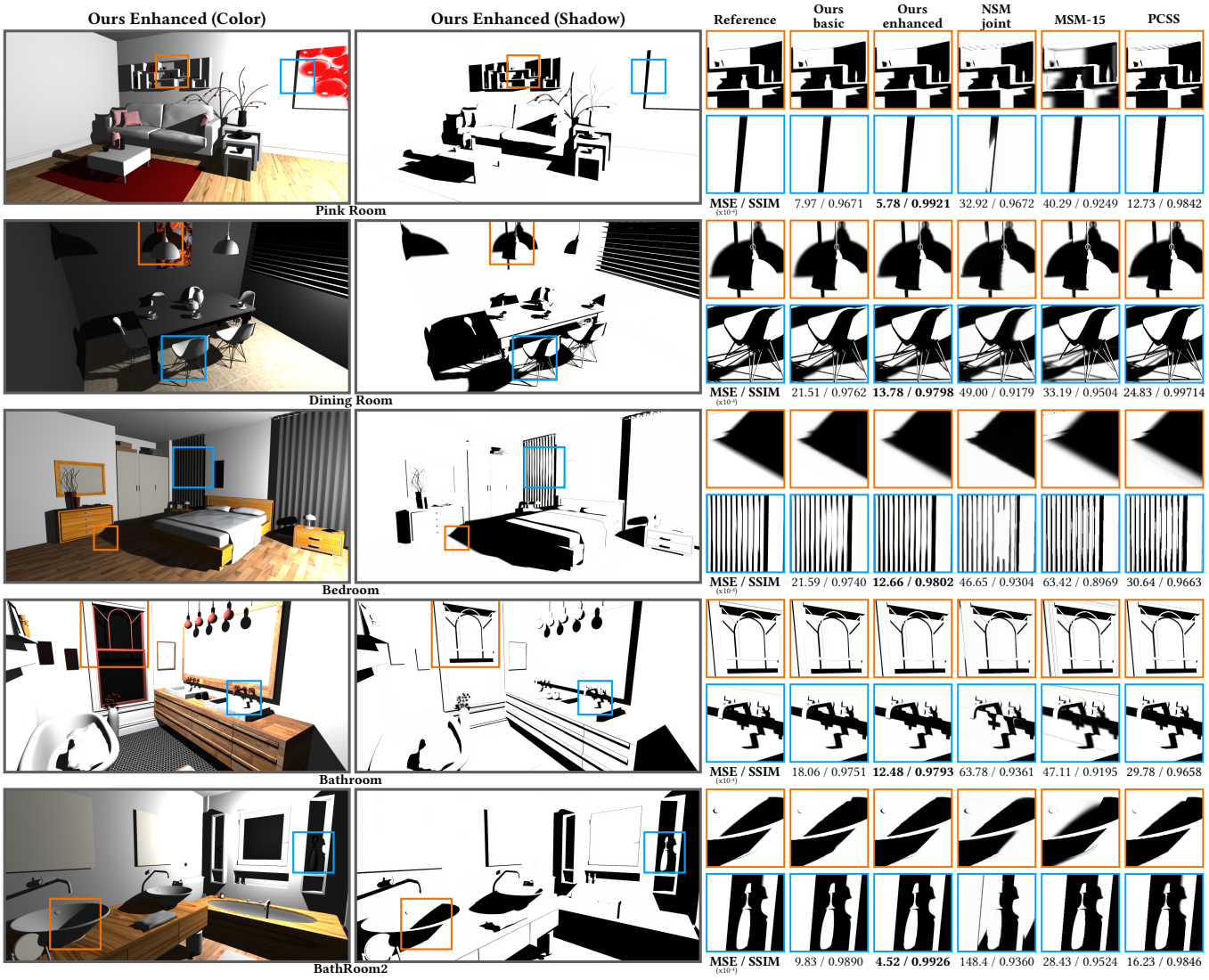

Fig. 7. Evaluation on five unseen scenes. Neural Shadow Mapping (*NSM-joint*) [Datta et al. 2022] and our variants (*Ours-basic* and *Ours-enhanced*) are trained on 5 training scenes and tested in new scenes. We also include results from 4-order moment shadow map (*MSM-15*) and PCSS for comparison.

6.2.2 Base shadow generator. Our method can accept shadows generated by any base shadowing technique. So far, we have referred to two shadow generators: classic shadow mapping and moment shadow mapping with bilinear interpolation. In the last column of Fig. 9, we explore another promising option: PCF with hardware-accelerated bilinear interpolation. However, it performs worse than moment shadow mapping, likely because it suffers more from shadow map aliasing.

6.2.3 Network structure. We study the impact of the number of dilated kernels and network layers in Table 2. Both factors directly affect the receptive field size: one at the network level and the other at the filter level. We observe that a small network receptive field can negatively impact accuracy and that the receptive fields of the network and filter should be balanced for optimal results. Our choice of a 4-layer network with 4 filters achieves the best performance among all tested configurations.

6.2.4 Temporal Loss. To demonstrate the effectiveness of our temporal loss in Eq. 6, we collect additional video sequences and assess the temporal consistency between frames. Specifically, we measure the *temporal error*, which is defined as the average Euclidean color difference between consecutive frames after warping by the motion vector:

$$E_{temporal} = \sqrt{\frac{1}{(T-1)HW}\sum_{t=1}^{T-1}\sum_{i=0}^{H-1}\sum_{j=0}^{W-1}\left(e^{t}_{i,j} - \mathcal{W}(e^{t-1})_{i,j}\right)^2}, \tag{7}$$

Table 4. Ablations on the choices of how to consider the penumbra width in input features. The results demonstrate that our choice of using the screen-space penumbra width as an input feature map is superior to other choices in both versions of our method (ours basic and ours enhanced).

	Ours basic	R_e basic	$c_e + R_e$ basic	p_{pcss} basic	Ours enhanced	R_e enhanced	$c_e + R_e$ enhanced	p_{pcss} enhanced
MSE $(\times 10^{-4})$	**14.59**	15.23	15.18	14.80	**9.67**	10.63	10.04	9.68
SSIM	**0.9735**	0.9722	0.9722	0.9729	**0.9787**	0.9751	0.9766	**0.9787**

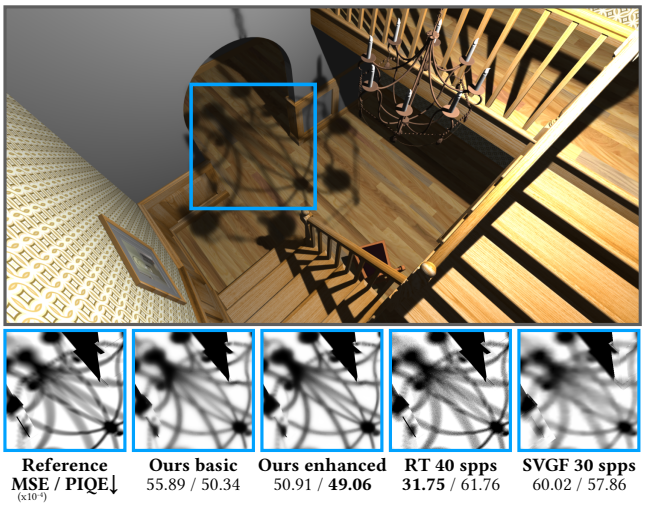

Fig. 8. Equal-time comparison with hardware ray tracing. *Ours-basic* and *Ours-enhanced* are variants of our models that use classic shadow mapping and moment shadow maps as the base shadow generators, respectively. *RT (40 spps)* and *SVGF (30 spps)* are ray tracing results where the latter is denoised with SVGF [Schied et al. 2017].

where T is the number of frames in the video and e^t is the soft shadow at frame t. We align the previous frame e^{t-1} to current frame with motion vector $\mathcal{W}$ before computing the error and mask out pixels that failed the reprojection tests.

We provide quantitative results in Table 3. The results confirm the effectiveness of our temporal loss in improving the temporal stability of predicted shadows.

6.2.5 Penumbra width as an input feature. The penumbra width is surely a crucial factor that affects the softness of the shadow and definitely should be fed into the kernel prediction network in some way. Our choice is to directly use the screen-space penumbra width p (Eq. 3) as an input feature map. We compare our choice to several other alternatives:

- R_e: using emitter radius as an input feature;
- $c_e + R_e$: adding emitter radius to the cosine feature map c_e as done in NSM [Datta et al. 2022].
- p_{pcss}: using emitter-space penumbra width as an input feature.

As shown in Table 4, our choice of considering the penumbra width is superior than other alternatives in terms of MSE and SSIM error numbers for both versions of our method. The results also verify that using estimated penumbra width is a better choice than using emitter size, and using screen-space penumbra width is a better choice than using the emitter-space one.

7 Conclusion and Future Directions

In this paper, we have presented kernel predicting neural shadow mapping. We assume that soft shadows can be approximated by pixelwise local filtering from the base shadow image in the screen space, where the local filter weights are predicted through a trained neural network. We use dilated filters as the representation of our local filters to maintain a balance between computational efficiency and representation capacity. We further enhance shadow quality by using moment shadow maps instead of the classic shadow map as our base shadow generator. With carefully designed filters, input features, and loss functions with temporal regularization, our method generates high-quality and temporally-stable results in real-time framerates. Our results are consistently superior to state-of-the-art methods in both visual qualities and numeric measures.

In the future, our method could be further improved and extended in several ways. First, similar to existing shadow mapping methods, since our main algorithm needs to be carried out once per light source, our computational cost is proportional to the number of lights. It is worthwhile to investigate more sophisticated solutions whose time complexity is independent of the number of lights. Second, existing screen-space works [MohammadBagher et al. 2010; Zheng and Saito 2011] already find the anisotropic properties of shadows and have utilize manually designed anisotropic bilateral filters to deal with it. While our method is able to model such anisotropy in an implicit way through supervised training, how to explicitly introducing such anisotropy into the shadow filter deserves future works. Last but not least, it would be also interesting to extend our work to real-time indirect illumination such as reflective shadow maps [Dachsbacher and Stamminger 2005].

Acknowledgments

We thank the anonymous reviewers for their valuable comments and insightful suggestions. This work is supported by the National Natural Science Foundation of China (Project No. 62372257).

References

Thomas Annen, Zhao Dong, Tom Mertens, Philippe Bekaert, Hans-Peter Seidel, and Jan Kautz. 2008a. Real-time, all-frequency shadows in dynamic scenes. *ACM Trans. Graph.* 27, 3 (Aug. 2008), 1–8. doi:10.1145/1360612.1360633

Thomas Annen, Tom Mertens, Philippe Bekaert, Hans-Peter Seidel, and Jan Kautz. 2007. Convolution shadow maps. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques* (Grenoble, France) *(EGSR'07)*. Eurographics Association, Goslar, DEU, 51–60.

Thomas Annen, Tom Mertens, Hans-Peter Seidel, Eddy Flerackers, and Jan Kautz. 2008b. Exponential shadow maps. In *Proceedings of Graphics Interface 2008* (Windsor, Ontario, Canada) *(GI '08)*. Canadian Information Processing Society, CAN, 155–161.

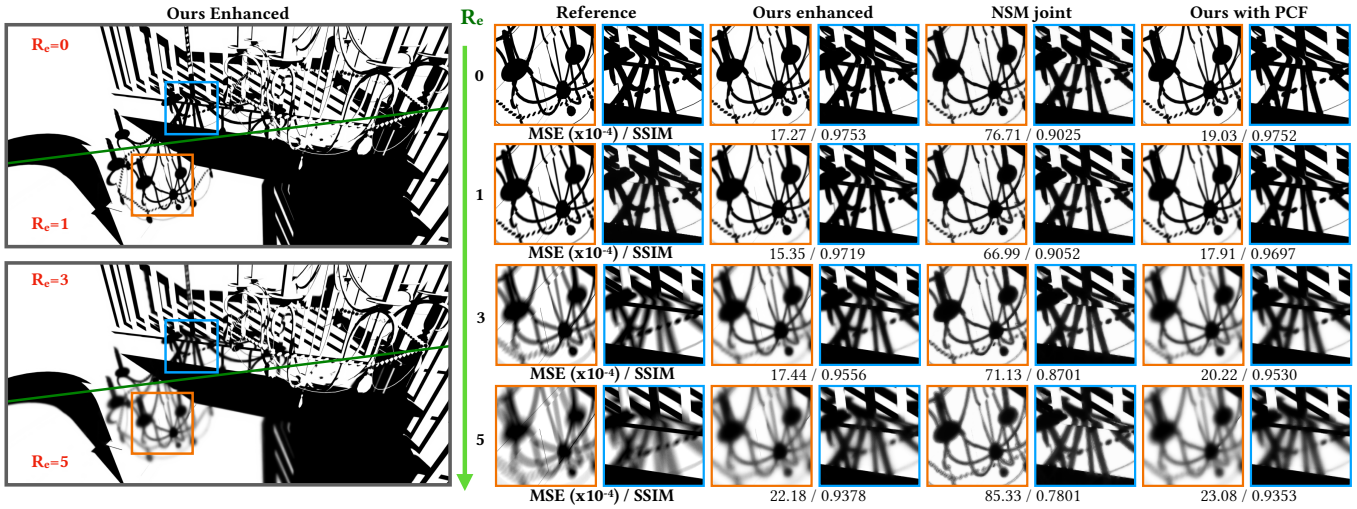

Fig. 9. Comparison of our method with Neural Shadow Mapping [Datta et al. 2022] under varying emitter sizes R_e, where $R_e = 0$ represents a point light source that produces purely hard shadows and $R_e = 5$ represents the largest emitter size. In the last column we compare with another base shadow generator: PCF with bilinear interpolation.

Steve Bako, Thijs Vogels, Brian Mcwilliams, Mark Meyer, Jan NováK, Alex Harvill, Pradeep Sen, Tony Derose, and Fabrice Rousselle. 2017. Kernel-predicting convolutional networks for denoising Monte Carlo renderings. *ACM Trans. Graph.* 36, 4, Article 97 (July 2017), 14 pages. doi:10.1145/3072959.3073708

Benedikt Bitterli. 2016. Rendering resources. https://benedikt-bitterli.me/resources/.

Carsten Dachsbacher and Marc Stamminger. 2005. Reflective shadow maps. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games* (Washington, District of Columbia) *(I3D '05)*. Association for Computing Machinery, New York, NY, USA, 203–231. doi:10.1145/1053427.1053460

Holger Dammertz, Daniel Sewtz, Johannes Hanika, and Hendrik P. A. Lensch. 2010. Edge-avoiding À-Trous wavelet transform for fast global illumination filtering. In *Proceedings of the Conference on High Performance Graphics* (Saarbrucken, Germany) *(HPG '10)*. Eurographics Association, Goslar, DEU, 67–75.

Sayantan Datta, Derek Nowrouzezahrai, Christoph Schied, and Zhao Dong. 2022. Neural Shadow Mapping. In *ACM SIGGRAPH 2022 Conference Proceedings* (Vancouver, BC, Canada) *(SIGGRAPH '22)*. Association for Computing Machinery, New York, NY, USA, Article 8, 9 pages. doi:10.1145/3528233.3530700

Zhao Dong and Baoguang Yang. 2010. Variance soft shadow mapping. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (Washington, D.C.) *(I3D '10)*. Association for Computing Machinery, New York, NY, USA, Article 18, 1 pages. doi:10.1145/1730804.1730990

William Donnelly and Andrew Lauritzen. 2006. Variance shadow maps. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games* (Redwood City, California) *(I3D '06)*. Association for Computing Machinery, New York, NY, USA, 161–165. doi:10.1145/1111411.1111440

Wolfgang Engel. 2006. *Shader X5: Advanced Rendering Techniques.* Charles River Media, Inc., USA.

Randima Fernando. 2005. Percentage-closer soft shadows. In *ACM SIGGRAPH 2005 Sketches* (Los Angeles, California) *(SIGGRAPH '05)*. Association for Computing Machinery, New York, NY, USA, 35–es. doi:10.1145/1187112.1187153

Simon Kallweit, Petrik Clarberg, Craig Kolb, Tom'aš Davidovič, Kai-Hwa Yao, Theresa Foley, Yong He, Lifan Wu, Lucy Chen, Tomas Akenine-Möller, Chris Wyman, Cyril Crassin, and Nir Benty. 2022. The Falcor Rendering Framework. https://github.com/NVIDIAGameWorks/Falcor https://github.com/NVIDIAGameWorks/Falcor.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG] https://arxiv.org/abs/1412.6980

Mahdi MohammadBagher, Jan Kautz, Nicolas Holzschuch, and Cyril Soler. 2010. Screen-space Percentage-Closer Soft Shadows. In *ACM SIGGRAPH 2010 Posters* (Los Angeles, California) *(SIGGRAPH '10)*. Association for Computing Machinery, New York, NY, USA, Article 133, 1 pages. doi:10.1145/1836845.1836987

Venkatanath N, Praneeth D, Maruthi Chandrasekhar Bh, Sumohana S. Channappayya, and Swarup S. Medasani. 2015. Blind image quality evaluation using perception based features. In *2015 Twenty First National Conference on Communications (NCC)*. 1–6. doi:10.1109/NCC.2015.7084843

Christoph Peters and Reinhard Klein. 2015. Moment shadow mapping. In *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games* (San Francisco, California) *(i3D '15)*. Association for Computing Machinery, New York, NY, USA, 7–14. doi:10.1145/2699276.2699277

Christoph Peters, Cedrick Munstermann, Nico Wetzstein, and Reinhard Klein. 2016. Beyond hard shadows: moment shadow maps for single scattering, soft shadows and translucent occluders. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (Redmond, Washington) *(I3D '16)*. Association for Computing Machinery, New York, NY, USA, 159–170. doi:10.1145/2856400.2856402

William T. Reeves, David H. Salesin, and Robert L. Cook. 1987. Rendering antialiased shadows with depth maps. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87)*. Association for Computing Machinery, New York, NY, USA, 283–291. doi:10.1145/37401.37435

Austin Robison and Peter Shirley. 2009. Image space gathering. In *Proceedings of the Conference on High Performance Graphics 2009* (New Orleans, Louisiana) *(HPG '09)*. Association for Computing Machinery, New York, NY, USA, 91–98. doi:10.1145/1572769.1572784

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi (Eds.). Springer International Publishing, Cham, 234–241.

Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty R. Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacker, Aaron Lefohn, and Marco Salvi. 2017. Spatiotemporal Variance-Guided Filtering: Real-time Reconstruction for Path Traced Global Illumination. In *ACM/EG Symposium on High Performance Graphics (HPG)*. 23–41. doi:10.1145/3105762.3105770

Lance Williams. 1978. Casting curved shadows on curved surfaces. In *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '78)*. Association for Computing Machinery, New York, NY, USA, 270–274. doi:10.1145/800248.807402

Fan Zhang, Hanqiu Sun, Leilei Xu, and Lee Kit Lun. 2006. Parallel-split shadow maps for large-scale virtual environments. In *Proceedings of the 2006 ACM International Conference on Virtual Reality Continuum and Its Applications* (Hong Kong, China) *(VRCIA '06)*. Association for Computing Machinery, New York, NY, USA, 311–318. doi:10.1145/1128923.1128975

Zhongxiang Zheng and Suguru Saito. 2011. Screen space anisotropic blurred soft shadows. In *ACM SIGGRAPH 2011 Posters* (Vancouver, British Columbia, Canada) *(SIGGRAPH '11)*. Association for Computing Machinery, New York, NY, USA, Article 75, 1 pages. doi:10.1145/2037715.2037799