Neural Shadow Mapping

Sayantan Datta
sayantan.datta@mail.mcgill.ca
McGill University
Montreal, QC, Canada
Meta Reality Labs
Redmond, WA, USA

Derek Nowrouzezahrai
derek@cim.mcgill.ca
McGill University
Montreal, QC, Canada

Christoph Schied
Zhao Dong
cschied@nvidia.com
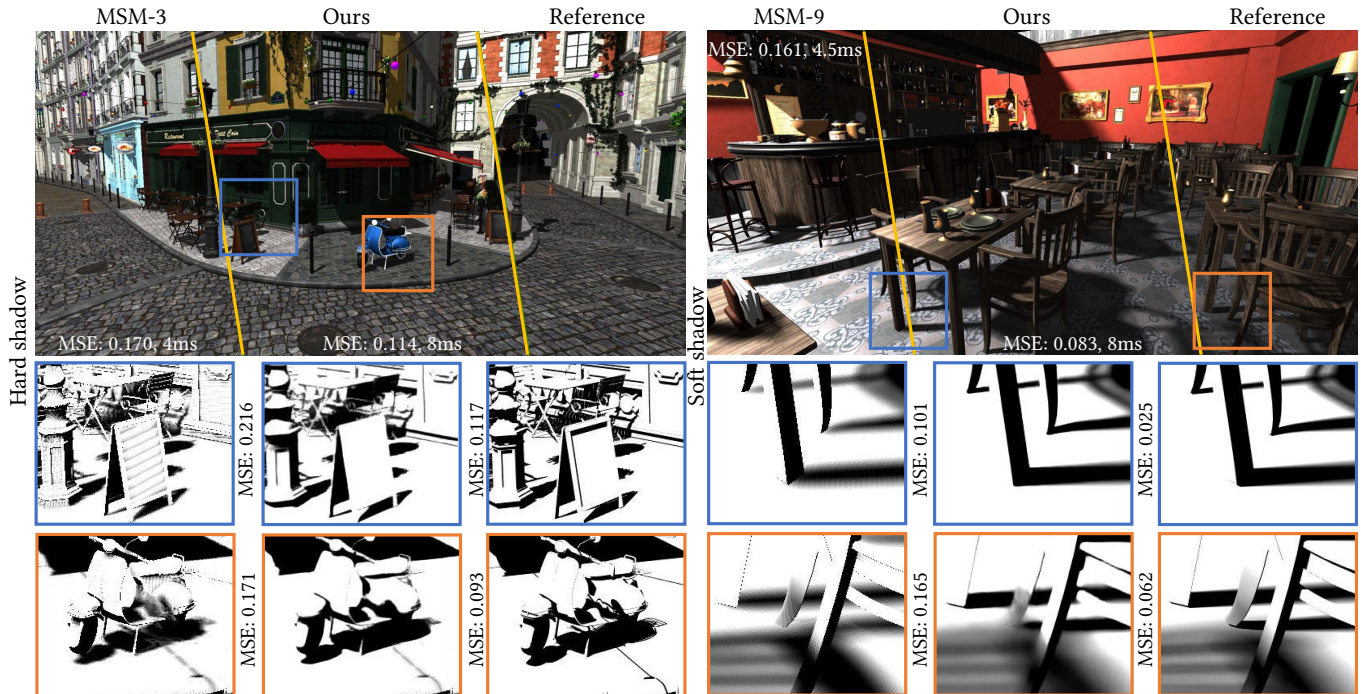zhaodong@fb.com
Meta Reality Labs
Redmond, WA, USA

Figure 1: Our hard and soft shadowing method approaches the quality of offline ray tracing whilst striking a favorable position on the performance-accuracy spectrum. On the high-performance end, we produce higher quality results than $n \times n$ Moment Shadow Maps (MSM-n). We require only vanilla shadow mapping inputs to generate visual (and temporal) results that approach ray-traced reference, surpassing more costly denoised interactive ray-traced methods.

ABSTRACT

We present a neural extension of basic shadow mapping for fast, high quality hard and soft shadows. We compare favorably to fast pre-filtering shadow mapping, all while producing visual results on par with ray traced hard and soft shadows. We show that combining memory bandwidth-aware architecture specialization and careful temporal-window training leads to a fast, compact and easy-to-train neural shadowing method. Our technique is memory bandwidth conscious, eliminates the need for post-process temporal anti-aliasing or denoising, and supports scenes with dynamic view, emitters and geometry while remaining robust to unseen objects.

CCS CONCEPTS

• **Computing methodologies** → **Visibility**; *Machine learning*; *Rasterization*.

KEYWORDS

Shadow mapping, neural networks

1 INTRODUCTION

Shadows provide important geometric, depth and shading cues. Real-time hard and soft shadow rendering remains a challenge, especially on resource-limited systems. Pre-filtering based methods [Annen et al. 2007; Donnelly and Lauritzen 2006; Peters and Klein 2015] are fast but approximate. They are prone to light leaking artifacts, reduced shadow contrast, and limited contact hardening. Interactive ray-tracing [Keller and Waechter 2009] coupled with post-process denoising [Chaitanya et al. 2017; Schied et al. 2017] and upscaling [Xiao et al. 2020] can deliver high quality dynamic shadows, but even the fastest GPU ray-tracers fall short of the performance demands of interactive graphics. Low ray-tracing hardware adoption and the added engineering complexity of integrating GPU ray tracers into rasterization-based pipelines is another limitation. Pre-computation based methods [Mildenhall et al. 2020; Sloan et al. 2002] do not generally support dynamic objects or near-field light transport, and require significant memory.

We propose a machine learning-based method that generates high quality hard and soft shadows for dynamic objects in real-time. Our approach does not require ray-tracing hardware, has high performance (< 6ms), requires little memory (< 1.5MBs), and is easy to deploy on commodity low-end GPU hardware. We use the output of "vanilla" rasterization-based shadow mapping (i.e., no cascades, etc.) to hallucinate temporally-stable hard and soft shadows. We design a compact neural architecture based on the statistics of penumbra sizes in a diversity of scenes. The network admits rapid training and generalizes to unseen dynamic objects. We demonstrate improved quality over state of the art in high-performance pre-filtering based methods while retaining support for dynamic scenes and approaching reference-quality results.

We show that careful feature engineering, application and memory aware architecture design, combined with a novel temporal stability loss results in a system with many favorable properties: apart from compactness and high-performance, our output precludes the need for post-process temporal anti-aliasing (TAA), further reducing the renderer's bandwidth requirements. We parameterize our network by emitter size, allowing us to encode both hard and soft shadow variation into a single trained net. We demonstrate its effectiveness on several scenes with dynamic geometry, camera, and emitters. Our results are consistently better than workhorse interactive methods, and they also rival much slower (and more demanding, system- and hardware-wise) interactive ray-tracing and denoising-based pipelines. Finally, we discuss scene dependent optimizations that further reduce our network size and runtime.

2 RELATED WORK

Shadow mapping [Williams 1978] and its variants are efficient shadowing methods for point and directional lights in dynamic scenes. Shadow map resolution and projection leads to shadow aliasing artifacts, with solutions (e.g., depth biasing [Dou et al. 2014; King 2004]) leading to secondary issues and trade-offs. Modern shadow mapping relies on delicately engineered systems that combine many *cascaded maps* [Engel 2006; Zhang et al. 2006]. Here, we refer readers to a comprehensive survey [Eisemann et al. 2011].

Filtering-based methods prefilter (in emitter-space) depth-based visibility to reduce aliasing. One simple such method weights nearby depth samples [Reeves et al. 1987]; this *percentage closer filtering* remains a commonly used technique in interactive applications, with a recent variant that modulates the filter size based on the relative blocker and receiver positions is used to approximate soft shadows [Fernando 2005]. More recently, a new class of filtering methods replace binary depth samples with statistical proxies, allowing for more sophisticated pre-filtering [Annen et al. 2007, 2008; Donnelly and Lauritzen 2006] and coarse approximations of soft shadows. Moment shadow maps [Peters and Klein 2015] are the state of the art of these methods, but it can suffer from banding, aliasing, light leaking in scenes with high depth complexity.

Screen-space methods treat G-buffers, including screen-projected shadow map data, leveraging image-space locality and GPU parallelization for efficient filtering in a deferred shading pipeline. Here, accurate filtering here requires the determination of an potentially-anisotropic filter kernel (due to perspective distortion), and so depends non-linearly on the viewing angle [Zheng and Saito 2011] and pixel depths [MohammadBagher et al. 2010]. Our method similarly treats image-space G-buffer data, but we instead *learn* compositional filters from data. High-fidelity soft shadows also benefit from occluder depth estimates from both the emitter and shade point, of which only the former is readily available from the shadow map and the latter can be approximated using min- [2018] or average-filtering [2010] of the projected shadow map. Again, we rely on learning compositions of convolution and pooling layers to model (the effects of) these depth estimates.

Ray tracing hardware opens up an exciting new avenue for dynamic hard and soft shadows. These methods, however, remain power-inefficient and typically require post-process denoising (traditional [Schied et al. 2017] or machine learning-based [Chaitanya et al. 2017; Munkberg and Hasselgren 2020]) and TAA [Edelsten et al. 2019; Xiao et al. 2020] to attain modest interactivity.

3 OVERVIEW

Overall, our approach is straightforward; we generate a set of screen-space buffers using a G-buffer and a shadow mapping pass before passing them as inputs to our network. The output of the network is compared against ray-traced shadows as target during training. Although straightforward, simply using a UNet [Nalbach et al. 2016; Ronneberger et al. 2015] without our proposed training
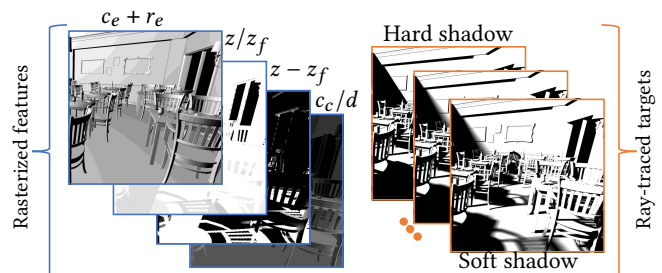

Figure 2: Visualizing supervised learning pairs. The network inputs are the rasterization buffers modulated by the size of emitter (r_e). The targets are generated using ray-tracing according to the corresponding emitter size.
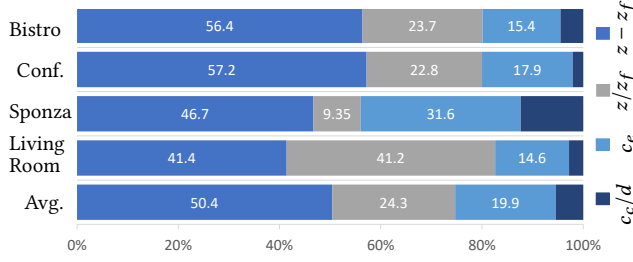
Figure 3: Relative sensitivity of the selected features for various scenes.

and optimization methodology yields a network that is temporally unstable, bandwidth limited, heavy (>25MBs) and too slow (>100ms) for real-time use. As such, our methodology is focused on making conscious choices to preserve memory bandwidth while having minimal impact on quality.

We train our network using screen-space buffers as features and corresponding ray-traced shadows as targets. The approach allows for easy integration into the rendering pipeline while providing room for integration (possible future work) into supersampling [Xiao et al. 2020] and neural-shading [Nalbach et al. 2016]. Our approach is also suitable for tiled rendering - popular among mobile devices. We develop a methodology to select a compact set of features that preserve necessary information without increasing memory bandwidth. We then show a simple technique to encode shadows with variable emitter size in a single network. We design a loss function to enhance the temporal stability of network without using historical buffers, thus further reducing bandwidth requirements. We provide several network architecture optimizations aimed at reducing memory and compute requirements. While our general architecture supports flexible emitter sizes, we show a recipe to further optimize our network for a fixed emitter size, enabling further trimming of network layers.

3.1 Supervised training pairs

We use supervised learning to train our neural network. The training examples are generated using rasterization and ray-tracing for features and targets respectively. The rasterization pipeline includes a G-buffer pass followed by a shadow mapping pass. Together they generate the following screen-space buffers:

- view-space depth d and normal $\mathbf{n}$,
- emitter-to-occluder depth z and emitter-space normal $\mathbf{n_e}$,
- pixel-to-emitter distance z_f, the emitter radius (size) r_e for spherical sources, and
- dot products $\{c_e, c_c\}$ of $\mathbf{n}$ with the emitter direction and $\mathbf{n}$ with the viewing direction.

The ray-tracing pass generates converged images of hard and soft shadows using a brute force Monte-Carlo sampling and a mild Gaussian filter. An 8x multi-sample anti-alising (MSAA) is also applied to the ray-traced targets. We do not however use MSAA in the rasterization pipeline as we expect the network to implicitly learn anti-aliasing from the target images.

3.1.1 Softness control. We train a single network to predict a range of shadows with varying softness. Note that the same input from the rasterization is used to generate both hard and soft shadows. The softness is controlled (on a continuous scale) using a scalar parameter indicating the size of the emitter. For training, the emitter sizes are encoded as integer textures between 0 to 4, where 0 indicates a point light and 4 indicates the largest emitter size (diameter 50 cm). Rather than passing the scalar values to the network as an additional constant screen-space buffer, a more bandwidth efficient approach is to add the scalar as a dc-offset to an already existing buffer. We choose the cosine texture (c_e) to add the emitter-size (r_e) dc-offset to. The network targets are also changed corresponding to the selected emitter size. See figure 2. During inference, the network accepts a scalar between 0 and 4 and intrinsically interpolates across discrete emitter values the network is trained with.

3.2 Feature selection

While dumping the content of the rasterization pass through a network works, it is bandwidth inefficient and adds a 2.5ms penalty to the cost of evaluating the network. The inefficient technique involves adding a feature extraction network, cascaded before the main network and training the two network end to end. The feature extraction network consisting of several layers of 1x1 convolutions, compresses all 15 channels of rasterizer output down to 4. A 1x1 convolution layer acts as fully connected layer across channels without performing any convolution across pixels. We tested a 2-layer deep 1x1 convolution network, recorded an overall error of 6.64×10^{-3} across a suite of scenes involving hard and soft shadows.

Our approach eliminates the need for a feature extraction network by systematic evaluation and selection of the rasterization output buffers. We first introduce the notion of *sensitivity*, a metric we use to quantify the importance of a feature. *Sensitivity* measures a change in the network output due to a small perturbation in the input. Intuitively, sensitivity is lower if a channel's contribution in explaining the output variation is lower. Absolute sensitivity S_i for the ith input channel f_i is given by

$$S_i = \mathbb{E}\left[\frac{(\phi(f_i + \epsilon_i) - \phi(f_i))}{0.1\sigma_i}\right], \quad \epsilon_i \sim \mathcal{N}(0, 0.1\sigma_i) \qquad (1)$$

where ϕ is the network, the random perturbation texture ϵ_i is obtained by sampling a normal distribution. The standard deviation σ_i corresponding to the ith channel is empirically estimated by aggregating all pixels in the dataset for that channel. The formula is repeated several times to reduce sampling noise. To compare the sensitivities across different training instances, we compute **relative sensitivity** as $s_i = S_i / \sum_i S_i$.

Armed with relative sensitivity as our yardstick, our problem is thus selecting a subset of features from a set of features $U = \{d, \mathbf{n}, z, \mathbf{n_e}, z_f, c_e, c_c\} + \{z - z_f, z/z_f, c_c/d, \mathbf{n} \cdot \mathbf{n_e}\}$. The first set of buffers are obtained directly from the rasterization while the second set is a composition from the first set. We take a tiered approach for selecting the best features. In the first pass, we train our network with all buffers in set U and reject buffers with low relative sensitivity. We repeat the process until all buffers have sensitivity higher than 1.5%. Refer to supplemental material, section 1.0.2 for more details. Our final set of buffers is as shown in figure 3. We obtain
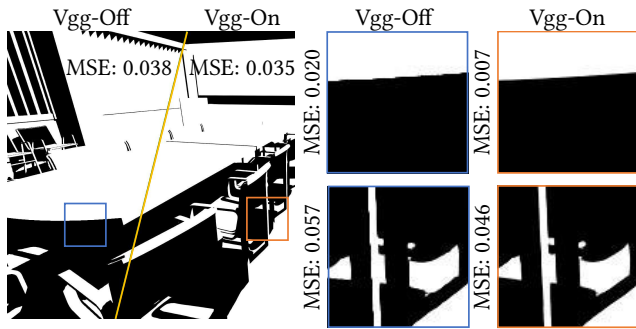
Figure 4: Effect of VGG loss on the final output. VGG loss produces sharper edges for geometry and shadow silhouettes.

nearly the same error (6.67×10^{-3}) as having a feature extraction while saving an extra 2.5ms.

3.3 Loss function and temporal stability

The loss function plays two main role in defining the characteristics of our network. It shapes the network to better fit the hard edges for shadow silhouette and geometry, essentially performing post-process anti-aliasing. Second, our loss function improves temporal stability without using any historical buffers for training and inference. Our approach not only saves memory bandwidth but also enables easier integration into tiled renderers.

We achieve the first objective using a weighted combination of per-pixel difference and VGG-19 [2015] perceptual loss. The effect of VGG loss on the final output is shown in figure 4 and clearly shows the anti-aliasing effect of VGG-19 on hard edges.

Existing methods typically improves temporal stability using historical buffers to better support the network during inference [Edelsten et al. 2019; Xiao et al. 2020] and also to reshape the loss function [Holden et al. 2019; Kaplanyan et al. 2019] during training. In our case, we do not use historical buffers but use random perturbations of the input buffers for reshaping the loss landscape during training. Temporal instabilities arise due to shadow-map aliasing, where shadow map texels do not align one-to-one with screen pixels. As such small, movement in camera or emitter can cause large changes in depth comparisons, especially around shadow silhouettes. Inspired from noise-to-noise training [Lehtinen et al. 2018], we train our network to learn from pairs of noisy inputs, in addition to the traditional supervised learning pair. Our network intrinsically learns to damp sharp changes due to small perturbations with minimal impact on overall quality as shown in figure 5. At each training iteration, we perturb the camera and emitter position by a small value proportional to the distance from the scene and size of emitter. For each perturbation, we collect the input buffers for training. The target is evaluated for only one of the perturbations. We evaluate the network on each perturbations and minimize the differences in the perturbed outputs as an additional loss function. All network instances evaluating the perturbed inputs share the same weights while backpropagation is only enabled through one instance, as

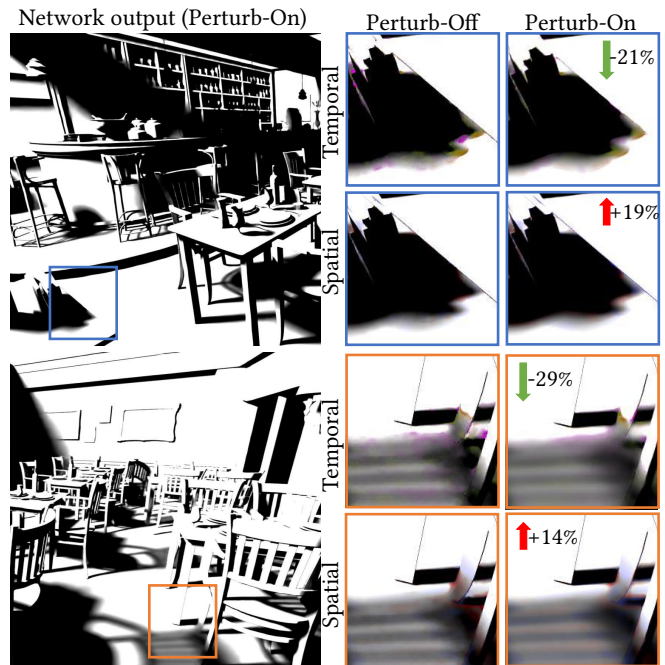$$\mathcal{L} = L(x_0, \tilde{x}) + \sum_{i=1}^{p} L(x_0, x_i), \qquad (2)$$

Figure 5: Comparing the temporal and spatial effect of perturbation loss. The application of perturbation loss reduces temporal instability while causing an increase in spatial blur as shown in the cutouts. We measure temporal instability by comparing the network output between consecutive frames while we measure spatial error by comparing the network output with reference. Temporal instability and spatial errors are represented using false colors purple/gold and red/blue respectively.

where $L(y, \tilde{y}) = \alpha \cdot |y - \tilde{y}| + (1 - \alpha) \cdot VGG19(y, \tilde{y})$, and x_i and $\tilde{x}$ are the network outputs and target. Only one network output-x_0 has backpropagation enabled through it. We set $\alpha = 0.9$ and the number of perturbations $p = 3$.

3.4 Temporal stability measurement

Several techniques exist for measuring perceptual similarity with respect to a single reference frame [Andersson et al. 2020; Wang et al. 2003] or reference video [Mantiuk et al. 2021]. These techniques measure the spatio-temporal difference which indicates the overall reconstruction quality across screen-space and time. Since we sacrifice spatial quality for temporal stability, using these metric may not indicate a reduction in temporal instability due perturbation loss, even when there is a clear visual improvement in temporal stability. Thus we formulate our own metric to measure only the temporal changes without considering spatial similarity with reference. To measure flickering, we find the motion-vector adjusted per-pixel temporal difference [Yang et al. 2020]. Since flickering can be quantified as an abrupt change in the pixel intensities between frames, we penalize the large differences more by passing the temporal pixel difference through an exponential. We aggregate the
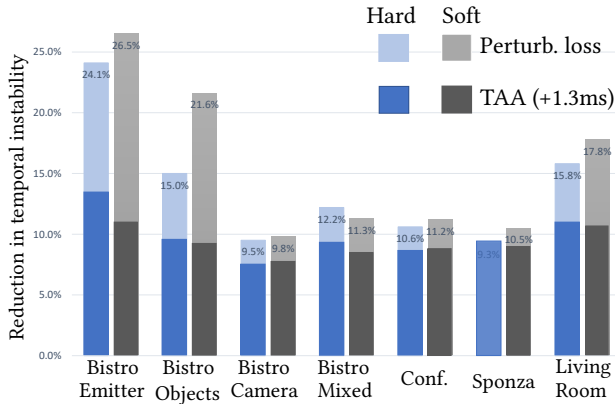
Figure 6: Comparing reduction in temporal instability between *Perturbation Loss* and TAA. Reference is trained without perturb. loss or TAA. TAA is implemented as an additional pass after network (trained without perturbation loss) evaluation requiring extra 1.3ms.

result across all pixels and frames, with

$$E = \frac{1}{P} \sum_{p,t} \{ \exp(\alpha D_t(p)) - 1 \} , \qquad (3)$$

where $D_t(p) = |I_t(p) - I_{t-1}(m(p))|$ is a per-pixel difference between two consecutive frames at time t and $m(p)$ abstracts away the motion-vector adjusted lookup at pixel p in the previous frame. We set $\alpha = 3$, which controls the penalty for large changes in intensity through time, and the normalizing factor P is the total number of pixels. We reject pixels that fail depth and normal comparison with its reprojection.

Figure 6 contrasts the effect on temporal stability between our loss and TAA (1 last frame): the improvement in temporal stability with our perturbation loss is strongest in scenes with dynamic emitters and non-negligible, albeit smaller with dynamic view.
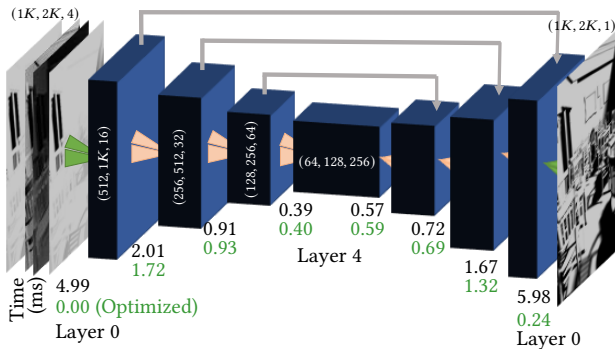
3.5 Network architecture and optimizations

The original UNet [Ronneberger et al. 2015] architecture is too slow (>100ms) to fit into a real-time graphics pipeline. As such, we start with trimming the network down. Our generic network has 5 layers, however, each layer composed of one 3x3 convolution and one 1x1 convolution layers as opposed to the standard double 3x3 convolution. A major departure from the original UNet is using bi-linear interpolation instead of expensive transpose convolutions for upscaling. We also use algebraic sum instead of a concatenation layer for merging the skip connections on the decoder side. A positive side effect of using a sum layer is the reduction in the number of hidden units on the decoder side. With these modifications we reduce the network size from 25MB to just 2.5MB, while the runtime is minimized to 28ms. Quantizing the network to half-precision further reduces the size to 1.5MB and 17ms runtime.

Other modifications for improving temporal stability without affecting performance includes using Average-pool instead of Max-pool and removing the skip connection in the first layer. Replacing max-pool with average-pool reduces extremities during processing and smooths out the output. As raw shadow map depth values are prone to aliasing noise, removing the first skip connection ensures the noisy input does not affect the output directly.

At this stage, we analyze the performance and error of our network before optimizing it further. Our validation error is 6.67×10^{-3} over an ensemble test scenes. From figure 7, we see that the first layer (combined encoder-decoder) requires more time compared to the rest of the layers combined. Moving from inner (#4) to outer layers (#0), the resolution is quadrupled while the number of channels is halved. Consequently, the effective cached memory bandwidth doubles as we move from inner to outer layers; however, with increasing resolution, memory operations are also more prone to cache misses. In practice, we see more than doubling of runtime as we move towards the outer layers. Refer supplemental section 3.

We further optimize by changing the first layer which consumes disproportionately more time. A naive approach is to replace the first layer with a downsampler on the encoder side and upsampler on the decoder side of UNet . However, simple downsampling and upsampling loses information contained in the input and also produces less sharp output. Instead, we flatten a 2x2 square of pixels into 4 separate channels and use the restructured buffer as the input
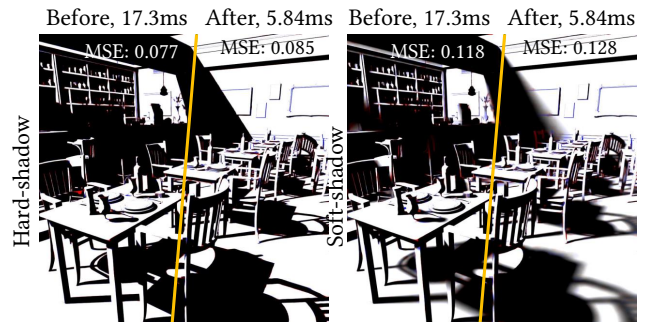

Figure 7: Layer-wise performance optimization for a 1024×2048 input.

Figure 8: Figure showing the effect of all optimizations in section 3.5.
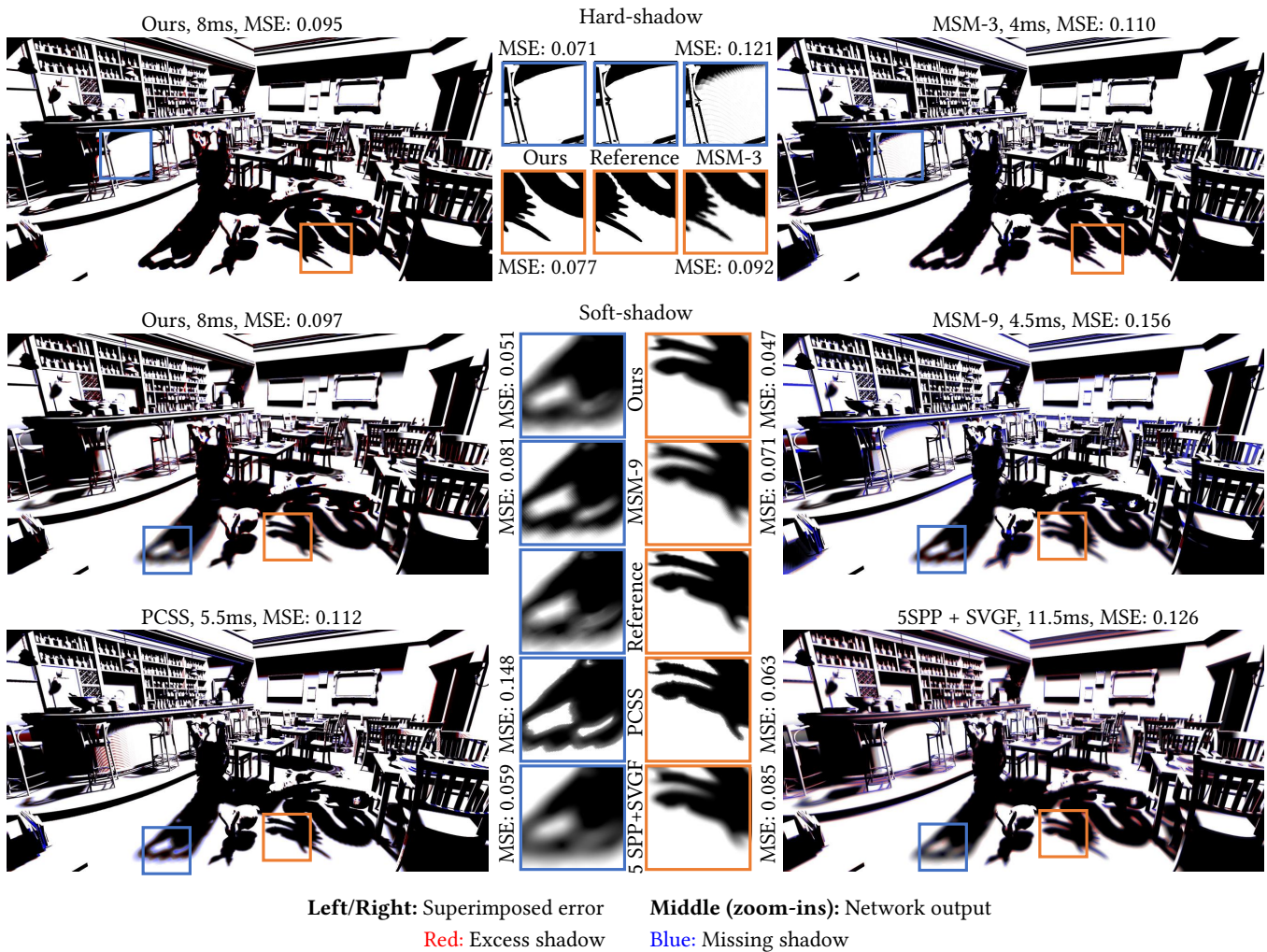
Figure 9: Comparing our network's ability to generalize to unseen objects (BUDDHA, BUNNY, DRAGON) with other competing techniques (MSM, PCSS, Raytracing & Denoising) for hard and soft shadows. MSM-3 and MSM-9 are Moment Shadow Map variants using 3×3 and 9×9 prefiltering kernels.

for second layer. Thus we rearrange the input and change the buffer dimensions from $(h \times w \times ch)$ to $(h/2 \times w/2 \times 4ch)$. More concretely, instead of feeding the first layer with full resolution (1024×2048) input with 4 channels, we feed the second layer directly with quarter resolution (512×1024) input with 16 channels. We do the inverse on the decoder side; rearrange 4 output channels into a 2x2 pixel square. Note that the rearrangement of buffers does not add any extra temporary storage for the second layer while removing the first layer (Conv2D operations) completely. On the decoder side, to improve training convergence, we upscale the first output channel to full resolution using a bi-linear interpolation. We then add rest of the three channels to the interpolated output, filling in rest of the details. The performance of our optimized network is 5.8ms.

3.6 Network depth optimizations

Our optimizations so far are generic and apply across scene and emitter configurations. Below, we explore *scene specific* optimizations and tune our network architecture for compactness. Shallower networks have many pragmatic benefits: it has exponentially (power of 2) fewer parameters, is faster to train, and admits faster runtime inference. Instead of relying on adhoc architecture tuning, we will choose architectures based on their ability to capture the shadowing effects we target. Specifically, we will build a simple model to estimate the maximum penumbra size for a scene configuration, and then relate this size exactly to the depth of the network suited to reproducing them. We empirically validate our model.

To compute the world space penumbra size, our simplified model assumes a spherical occluder (or, conservatively, a bounding sphere around occluding geometry). When generating training samples, we additionally measure the minimum and maximum occluder

a. Model estimating penumbra
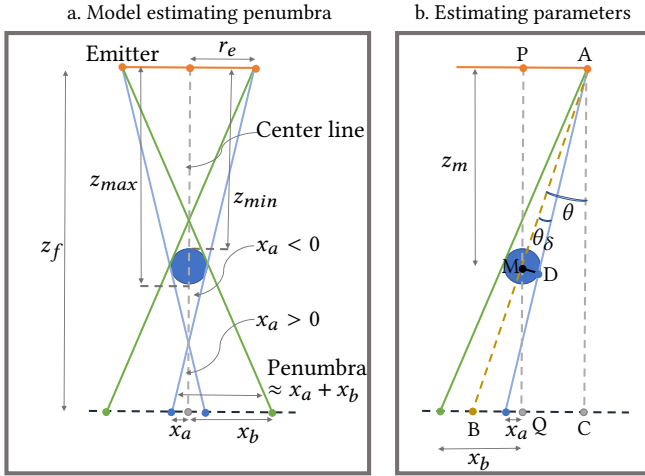
b. Estimating parameters

Figure 10: A simplified model to estimate the penumbra size at a given pixel. We assume our occluder is spherical in shape, forming a convex bounding-sphere around the occlusion geometry as shown in figure (a) on the left. Figure (b) shows a simplified diagram to estimate the parameters θ, θ_δ.

distances z_{max}, z_{min} (figure 10, a). We then estimate the penumbra width at a pixel as the sum of inner (x_a) and outer (x_b) penumbra (figure 10, b) as $\{x_a, x_b\} = z_f \tan(\theta \pm \theta_\delta) - r_e$. We derive the parameters θ, θ_δ in the supplemental, section 4.

After computing a histogram of penumbra sizes in screen-space for each pixel across all the training frames, we select the highest 95th percentile penumbra size as a conservative bound on the *receptive field* size requirements for our neural architecture.

We can modulate the per-layer convolutional layer parameters (kernel size, stride) and pooling operation parameters in order to meet the target receptive field requirements. If we set each convolutional layers to halve the spatial resolution, the *effective receptive field* of the network grows with $\times 2^l$ for an l-layer network. Exclusively using 3×3 kernels, we can solve for $l = \log_2(p_w/3)$, where p_w is the conservative screen-space penumbra *width*.

Table 1 provides an empirical validation of our technique. We train three networks with 3,5, and 7 layers using the same dataset. The dataset consist of scenes with mixture of penumbra sizes. The penumbra size estimates are computed using our model. During inference, networks with *receptive field* lower than the predicted penumbra size perform poorly as marked in red color. A more

Table 1: MSE for variable penumbra sizes and with 3/5/7-layer nets.

Predicted Penumbra Width (in pixels)	Network layers → Receptive field size (in pixels)		
	3 → 24	5 → 96	7 → 384
21	0.005	0.005	0.012
90	0.083	0.009	0.018
180	0.161	0.042	0.019

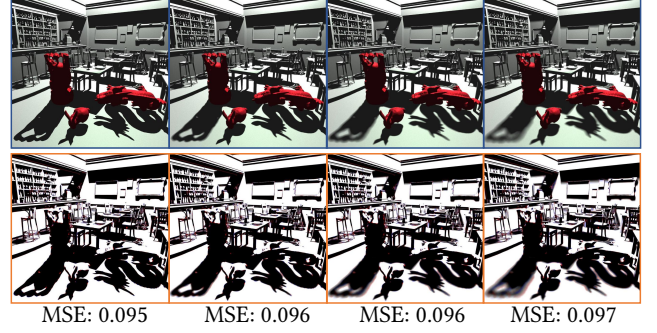MSE: 0.095　　MSE: 0.096　　MSE: 0.096　　MSE: 0.097

Figure 11: Generalization to untrained objects (red, first row) in a trained scene. Second row visualizes false color error w.r.t. reference. From left to right, emitter size increases linearly from 0 (hard shadow) to 50 cm diameter.

detailed analysis of the empirical verification is provided in the supplemental section 4.1.

4 RESULTS AND COMPARISONS

We demonstrate our method on a diversity of scenarios. We augment static environments (e.g., rooms) included in our training set to include untrained objects at runtime, illustrating an important use case for interactive settings like games (figure 11). We train a single network on the BISTRO INTERIOR scene with varying emitter sizes, emitter positions and camera trajectories and introduce the (untrained objects) BUDDHA, BUNNY, and DRAGON for validation.

Comparison. In the soft-shadowing regime, our comparisons focus on two classes of baseline methods: first, high-performance rasterization-based approximations such as Moment Shadow Maps (MSM) [Peters and Klein 2015] and Percentage Closer Soft Shadows (PCSS) [Fernando 2005] that align with our engineering (i.e., fully rasterization-based; no ray-tracing) and performance targets as our primary baseline; second, we use interactive GPU ray-tracing with post-process denoising as a more accurate "interactive" baseline, i.e., 5-SPP raytracing with SVGF [Schied et al. 2017]. Note that, unlike our method, neither MSM nor PCSS allow explicit control of penumbra style using emitter size; as such, we adjust the pre-filtering kernel size for MSM and PCSS to achieve a penumbra size that most closely matches reference renderings. We use kernel sizes of 3×3 for MSM and 9×9 for PCSS. Our PCSS baseline also includes a depth-aware post-filtering.

Our method consistently improves shadow quality at competitive performances (figure 9). Refer to our video to observe the temporal stability of our results. For hard shadows, we compare to 3×3 MSM, obtaining alias-free shadows without any light leaking. Please refer to the supplemental section 7 for more results, comparisons.

Runtime comparisons are measured at a resolution of 2k×1k on an AMD 5600X CPU and NVIDIA 2080TI GPU. The timings in all figures, both main paper and supplemental, exclude G-Buffer generation which consistently requires an additional 2-3 ms (depending on the scene) across all techniques. Each scene is trained on ≤ 400 images of resolution 2k×1k on a cluster for roughly 16 hours (75 training epochs).

5 LIMITATIONS

Our technique shares similar limitations to other screen-space methods. The unavailability of layered depth information, both in camera- [Ritschel et al. 2009] and emitter-space [Jesus Gumbau 2018] leads to an ill-posedness of the problem that results in approximation error. In camera space, the lack of peeled-depth data complicates the determination of mutual visibility between pixels. Similarly, computing the blur kernel size to soften shadow silhouettes relies on the distance between shading points and occluders, which is also unavailable in our setting. Our network and training methodology are effectively designed to compensate for this ill-posedness, bridging the visual gap in a diversity of object/scene arrangements by leveraging complex patterns inherent in the data. Figure 12 highlights a standard failure case and our supplemental includes additional discussion (section 8).

6 CONCLUSIONS

We presented a compact, fast neural method and training loss suited to temporally-coherent hard and soft shadows synthesis using only basic shadow map rasterized inputs. We showed that – with a careful, problem-specific architecture design and a new, simple temporal loss – a single small network can learn to hallucinate hard and soft shadows from varying emitter sizes and for a diversity of scenes. It is robust to the insertion of unseen objects, requires only a modest training budget, and precludes the need for any post-process denoising and/or TAA. Our approach yields stable hard and soft shadows with performance similar to workhorse interactive approximations and higher quality than (more expensive) GPU-raytracing and denoising (and TAA) alternatives.

Rasterization-based approaches for soft and hard shadows rely on heuristics and brittle manual tuning to achieve consistent, visually-desirable results. Our data-driven approach precludes such tuning, improving shadow quality at a modest cost, producing plausible and temporally-coherent soft shadows without any ray-tracing. Ours is a compact neural shading-based framework [Nalbach et al. 2016] suitable for low-power tiled-rendering systems, striking an interesting trade-off in a complex design space. We demonstrate benefits that largely offset the added training and integration complexity.

In the future, pursuing more aggressive neural architecture optimizations, including quantization and procedural architecture search, could likely further improve inference performance. When coupled with sparsification using, e.g., *lottery ticket-based* method [Frankle and Carbin 2018], we suspect that significant additional performance gains are possible, all without sacrificing quality.

REFERENCES

Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D. Fairchild. 2020. FLIP: A Difference Evaluator for Alternating Images. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 3, 2 (2020), 15:1–15:23.

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 P. Seidel, Eddy Flerackers, and Jan Kautz. 2008. Exponential Shadow Maps. In *Proceedings of Graphics Interface 2008* (Windsor, Ont., Canada) *(GI '08)*. Canadian Information Processing Society, CAN, 155–161.

Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. 2017. Interactive Reconstruction of Monte Carlo Image Sequences Using a Recurrent Denoising Autoencoder. *ACM Trans. Graph.* 36, 4, Article 98 (July 2017), 12 pages.

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.

Hang Dou, Yajie Yan, Ethan Kerzner, Zeng Dai, and Chris Wyman. 2014. Adaptive Depth Bias for Shadow Maps. In *Proceedings of the 18th Meeting of the ACM SIGGRAPH Symposium on I3D Graphics and Games* (San Francisco, California) *(I3D '14)*. Association for Computing Machinery, New York, NY, USA, 97–102.

Andrew Edelsten, Paula Jukarainen, and Anjul Patney. 2019. Truly Next-Gen: Adding Deep Learning to Games and Graphics. NVIDIA Sponsored Sessions (Game Developers Conference.

Elmar Eisemann, Michael Schwarz, Ulf Assarsson, and Michael Wimmer. 2011. *Real-Time Shadows* (1st ed.). A. K. Peters, Ltd., USA.

Wolfgang F Engel. 2006. *ShaderX5: Advanced Rendering Techniques*. C.R. Media.

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.

Jonathan Frankle and Michael Carbin. 2018. The Lottery Ticket Hypothesis: Training Pruned Neural Networks. *CoRR* abs/1803.03635 (2018). arXiv:1803.03635

Daniel Holden, Bang C. Duong, Sayantan Datta, and Derek Nowrouzezahrai. 2019. Subspace Neural Physics: Fast Data-Driven Interactive Simulation. In *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Los Angeles, California) *(SCA '19)*. Association for Computing Machinery, New York, NY, USA, Article 6, 12 pages.

Mateu Sbert Jesus Gumbau, Miguel Chover. 2018. *Screen Space Soft Shadow, GPU Pro 360 Guide to Rendering* (1st ed.). A. K. Peters, Ltd., USA.

Anton S. Kaplanyan, Anton Sochenov, Thomas Leimkühler, Mikhail Okunev, Todd Goodall, and Gizem Rufo. 2019. DeepFovea: Neural Reconstruction for Foveated Rendering and Video Compression Using Learned Statistics of Natural Videos. *ACM Trans. Graph.* 38, 6, Article 212 (nov 2019), 13 pages.

Alexander Keller and Carsten Waechter. 2009. Real-time precision ray tracing. US patent US20070024615A1.

Gary King. 2004. *Shadow mapping algorithms*. Nvidia. 354–355 pages.

Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. 2018. Noise2Noise: Learning Image Restoration without Clean Data. *CoRR* abs/1803.04189 (2018). arXiv:1803.04189

Amazon Lumberyard. 2017. Amazon Lumberyard Bistro, Open Research Content Archive (ORCA). http://developer.nvidia.com/orca/amazon-lumberyard-bistro http://developer.nvidia.com/orca/amazon-lumberyard-bistro.

Foliage-hard   Overlapping Mesh   Foliage-soft
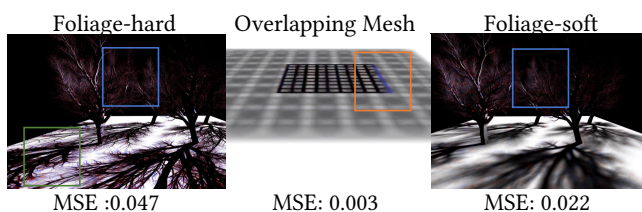
MSE :0.047   MSE: 0.003   MSE: 0.022

Figure 12: Limitations of our method with high-frequency/high-depth complexity. Orange, blue, and green boxes highlight visual artifacts due to limited emitter and camera depth information, and due to undersampled training scenarios (i.e., loss of fine details).

Rafał K. Mantiuk, Gyorgy Denes, Alexandre Chapiro, Anton Kaplanyan, Gizem Rufo, Romain Bachy, Trisha Lian, and Anjul Patney. 2021. FovVideoVDP: A Visible Difference Predictor for Wide Field-of-View Video. *ACM Trans. Graph.* 40, 4, Article 49 (July 2021), 19 pages.

Morgan McGuire. 2017. Computer Graphics Archive. https://casual-effects.com/data https://casual-effects.com/data.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *CoRR* abs/2003.08934 (2020). arXiv:2003.08934

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. https://doi.org/10.1145/1836845.1836987

Jacob Munkberg and Jon Hasselgren. 2020. Neural Denoising with Layer Embeddings. *Computer Graphics Forum* (2020).

Oliver Nalbach, Elena Arabadzhiyska, Dushyant Mehta, Hans-Peter Seidel, and Tobias Ritschel. 2016. Deep Shading: Convolutional Neural Networks for Screen-Space Shading. *CoRR* abs/1603.06078 (2016). arXiv:1603.06078

Christoph Peters and Reinhard Klein. 2015. Moment Shadow Mapping. In *Proceedings of the 19th I3D Symposium* (San Francisco, CA) *(i3D '15)*. Association for Computing Machinery, New York, NY, USA, 7–14.

William T. Reeves, David H. Salesin, and Robert L. Cook. 1987. Rendering Antialiased Shadows with Depth Maps. *SIGGRAPH Comput. Graph.* 21, 4 (Aug. 1987), 283–291.

Tobias Ritschel, Thorsten Grosch, and Hans-Peter Seidel. 2009. Approximating Dynamic Global Illumination in Image Space. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games* (Boston, Massachusetts) *(I3D '09)*. Association for Computing Machinery, New York, NY, USA, 75–82. https://doi.org/10.1145/1507149.1507161

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. *CoRR* abs/1505.04597 (2015). arXiv:1505.04597

Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty R. Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, Aaron Lefohn, and Marco Salvi. 2017. Spatiotemporal Variance-Guided Filtering: Real-Time Reconstruction for Path-Traced Global Illumination. In *Proceedings of High Performance Graphics* (Los Angeles, California) *(HPG '17)*. Association for Computing Machinery, New York, NY, USA, Article 2, 12 pages.

Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*.

Peter-Pike Sloan, Jan Kautz, and John Snyder. 2002. Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. *ACM Trans. Graph.* 21, 3 (July 2002), 527–536.

Z. Wang, Eero Simoncelli, and Alan Bovik. 2003. Multiscale structural similarity for image quality assessment. *Conference Record of the Asilomar Conference on Signals, Systems and Computers* 2, 1398–1402Vol.2.

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.

Lei Xiao, Salah Nouri, Matt Chapman, Alexander Fix, Douglas Lanman, and Anton Kaplanyan. 2020. Neural Supersampling for Real-Time Rendering. *ACM Trans. Graph.* 39, 4, Article 142 (July 2020), 12 pages.

Lei Yang, Shiqiu Liu, and Marco Salvi. 2020. A Survey of Temporal Antialiasing Techniques. *Computer Graphics Forum* 39, 2 (2020), 607–621. arXiv:https://onlinelibrary.wiley.com/olddoi/pdf/10.1111/cgf.14018

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.

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. https://doi.org/10.1145/2037715.2037799