Supplemental Material for: Ragged Neighborhood Attention for Spatiotemporal Neural Denoising of Deep Monte Carlo Renderings

XIANYAO ZHANG*, DisneyResearch|Studios, Switzerland
GERHARD RÖTHLIN*, DisneyResearch|Studios, Switzerland
TUNÇ OZAN AYDIN, DisneyResearch|Studios, Switzerland
FARNOOD SALEHI, DisneyResearch|Studios, Switzerland
MARIOS PAPAS†, DisneyResearch|Studios, Switzerland

This supplemental document contains implementation details, RaNA operator variants, training configuration, ablations, timing results, and additional workflow and failure case discussion supporting the main manuscript.

ACM Reference Format:
Xianyao Zhang, Gerhard Röthlin, Tunç Ozan Aydın, Farnood Salehi, and Marios Papas. 2026. Supplemental Material for: Ragged Neighborhood Attention for Spatiotemporal Neural Denoising of Deep Monte Carlo Renderings. *ACM Trans. Graph.* 1, 1 (May 2026), 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

S1 Overview

This document provides additional details supporting the main manuscript. Section S2 discusses down-sampling and up-sampling operators used in RaNAD. Section S3 describes the RaNA operator variants used throughout the network. Section S4 presents full implementation details including: deep image representation and CUDA kernel algorithms; architecture configurations and training hyperparameters; input transform equations (log1p, reciprocal, sinusoidal encoding); submodule visualizations; and training loss formulations including our modified FLIP for stable optimization. Section S5 provides additional methodology details including multi-scale reconstruction visualization. Section S6 provides additional quantitative comparisons and ablations.

The supplemental material contains, beyond this document, public scene examples, a supplemental video, and two supplemental image viewers. For reproducibility, the supplemental includes representative public scene examples with noisy deep inputs, RaNAD-denoised deep outputs, and reference deep renders. The supplemental video compares RaNAD-1f, RaNAD-cf, and KPAL-cf side by side to show the temporal stability gains of RaNAD-cf over RaNAD-1f, with KPAL-cf included for context. The supplemental image viewers provide the 32spp and 128spp flattened denoising results from DACNN, RaNAD-1f, RaNAD-cf, OIDN, KPAL-1f, and KPAL-cf, along with quantitative metrics, for datasets A1 and B, respectively.

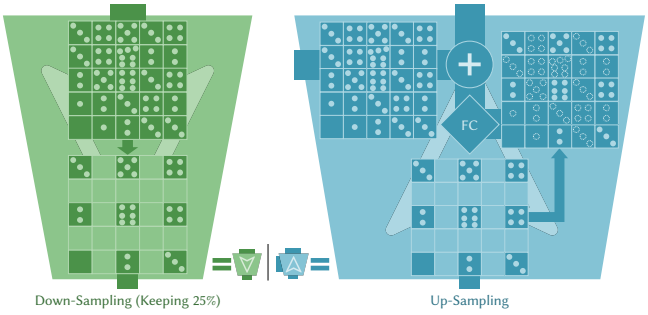


Fig. S1. Down-sampling (left) and up-sampling operators (right).

S2 Down- and Up-Sampling Operators

Figure S1 summarizes the down- and up-sampling operators. The down-sampling operator implements bin-level dropout. Its responsibility is to reduce the average bin density in the image, and fulfills a similar role as max-pooling or average-pooling. After dropping bins, the remaining bins are used as queries for attention layers to obtain sparser embeddings. In other words, the down layer itself changes the bin layout, while the neighboring RaNA layers perform feature aggregation on that new layout. This is why the backbone's resizing path is related to, but not identical to, the explicit aggregation attention used in reconstruction.

The reason dropout is used instead of max or mean pooling, like for KPAL [Vogels et al. 2018], is that it is unclear which bins should be pooled together. We experimented with the following options; none fully removes low-sample artifacts while maintaining consistently high quality.

- Random Pattern: Purely random, every bin has a P_{keep} probability to not be dropped
- Regular Pattern: View all bins in the image as a linear sequence and take uniform strides proportional to $\frac{1}{P_{\text{keep}}}$
- Pixel Regular pattern: Pixels are dropped in a regular grid pattern; that is, all bins from those pixels will be dropped.

Changing resolution in deep images therefore remains an open problem. Our current down-sampling operators are effective overall,

*Both authors contributed equally to this work.
†Corresponding author.

Authors' Contact Information: Xianyao Zhang, DisneyResearch|Studios, Zürich, Switzerland, xianyao.zhang@disneyresearch.com; Gerhard Röthlin, DisneyResearch|Studios, Zürich, Switzerland, gerhard@disneyresearch.com; Tunç Ozan Aydın, DisneyResearch|Studios, Zürich, Switzerland, tunc@disneyresearch.com; Farnood Salehi, DisneyResearch|Studios, Zürich, Switzerland, farnood.salehi@disneyresearch.com; Marios Papas, DisneyResearch|Studios, Zürich, Switzerland, marios.papas@disneyresearch.com.

ACM 1557-7368/2026/5-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

but they can emphasize preserved bins over discarded ones. With uniform patterns such as checkerboard down-sampling, this can produce grid-like aliasing artifacts in very noisy regions. These artifacts fade as sample count increases and become imperceptible at moderate to high spp levels.

The up-sampling operator merges the features from a lower scale with the shortcut connection on an upper scale. It up-scatters the data from the lower scale into the shape of the upper scale, which gives a tensor with the bin layout of the upper scale, where all bins that were in the lower scale are filled in, and the rest is zero. These up-scattered values are added with the encoder features from the shortcut connection. The up layer therefore restores a finer bin layout deterministically before the subsequent RaNA blocks continue feature propagation at that finer scale.

S3 RaNA Variants

Below we explain some RaNA variants used in the RaNAD architecture. Figure S4 shows the effective integer-lattice neighborhoods induced by the radii used in RaNA operators and the corresponding down-sampling semantics.

Self Attention. Self-attention is used in the core network, although locally, this also implements cross-attention of all bins in each pixel with all bins in the neighborhood. This is used with distinct query, key and value tensors derived from the same feature via a fully connected layer in the core network.

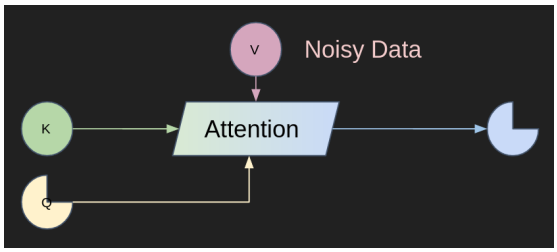


Fig. S2. RaNA for down-sampling aggregation in the reconstruction module. Full circles represent high-resolution data and the three-quarter disks represent low-resolution data. The output of this operation has a lower resolution.

Aggregation RaNA. In the first stage of the reconstruction block, the aggregation attention receives query embeddings with fewer bins than key and value (Figure S2). As the output of the layer has the same bin layout as the query, aggregation attention can be used for down-sampling the data, e.g. by keeping fewer pixels on the image plane. In practice, keys come from backbone features and values come from the raw linear image data. In the reconstruction module, this is the attention-based resizing step: sparse queries define which bins remain at the coarser scale, while attention performs the actual aggregation from the denser key–value set.

Denoise RaNA. The denoise attention, on the other hand, receives the query at full-resolution and the key and value can be sparser (Figure S3). This means a full-resolution output from low-resolution aggregated input. In the current multi-scale reconstruction block,

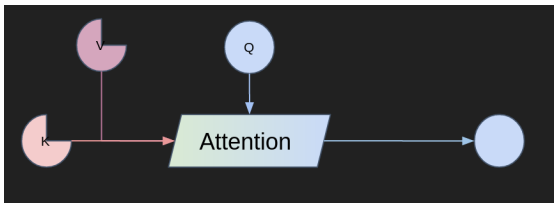


Fig. S3. RaNA for denoising at a lower scale. Full circles represent high-resolution data and the three-quarter disks represent low-resolution data. The output of this operation is a high-resolution image.

there can be multiple levels of sparsity and blurriness for K and V, which can reconstruct different levels of detail. These reconstructions are then combined through the scale mixer.

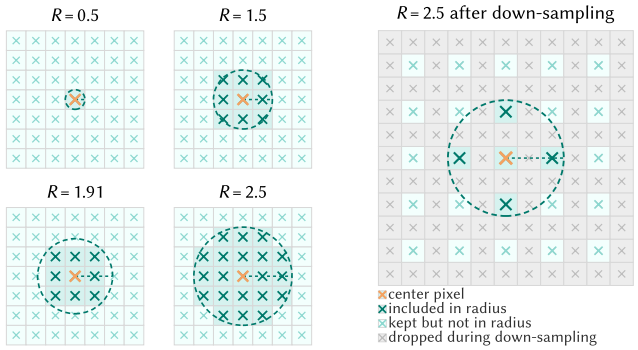


Fig. S4. Effective RaNA neighborhoods and down-sampling semantics. The top panels mark integer pixel centers with crosses and dashed Euclidean radius circles: $R = 0.5$ selects only the center pixel; $R = 1.5$ and $R = 1.91$ select the same 8-connected neighborhood; and $R = 2.5$ reaches a wider set of integer centers. The bottom panel keeps the full-resolution lattice visible after down-sampling: the same background shades distinguish pixels included in the radius from kept pixels outside the radius, while dim fills mark pixels dropped during down-sampling.

S4 Implementation Details

S4.1 Deep Image Representation

We represent deep images using TensorFlow's `RaggedTensor`, which supports variable-length dimensions. A deep image is stored as a tensor with shape $N \times H \times W \times B \times C$, where the bin dimension B is ragged (variable per pixel).

For CUDA kernel interfaces, we flatten this to two components:

- A dense tensor of shape $[N_{\text{bins}}, C]$ containing all bin values concatenated in row-major order.
- A splits tensor of shape $[H \times W + 1]$ encoding cumulative bin counts, such that bins for pixel (x, y) span indices $[\text{splits}[y \cdot W + x], \text{splits}[y \cdot W + x + 1])$.

This representation enables $O(1)$ random access to any pixel's bins with two memory reads.

S4.2 CUDA Kernel Implementation

This section provides the full mathematical details of our custom CUDA kernels for the RaNA operator.

S4.2.1 Online Softmax for Forward Pass. The key challenge in implementing RaNA on GPU is avoiding materialization of the full attention matrix. We employ an online softmax algorithm inspired by FlashAttention-2 [Dao 2023], computing attention incrementally as we iterate over key-value pairs.

For each query q, we maintain running statistics: the maximum dot product m and the exponential sum $\ell = \sum_j \exp(s_j - m)$. When processing a new key k_j with dot product $s_j = q \cdot k_j$, we update:

$$m' = \max(m, s_j), \tag{S1}$$

$$\ell' = \ell \cdot e^{m-m'} + e^{s_j - m'}, \tag{S2}$$

$$o' = o \cdot e^{m-m'} + e^{s_j - m'} \cdot v_j, \tag{S3}$$

where o is the running output accumulator. The final output is o/ℓ. The forward pass also stores $\log(\ell) + m$ (the `logsumexp`) for each query bin, enabling efficient gradient computation without re-reading keys.

S4.2.2 Backward Pass Gradient Computation. The backward pass uses a key/value-centric iteration order to minimize atomic operations. The softmax gradient follows the standard derivation:

$$\frac{\partial \mathcal{L}}{\partial s_{ij}} = p_{ij}\left(\frac{\partial \mathcal{L}}{\partial o_i} \cdot v_j - \sum_k p_{ik} \frac{\partial \mathcal{L}}{\partial o_i} \cdot v_k\right), \tag{S4}$$

where $p_{ij} = \exp(s_{ij})/\sum_k \exp(s_{ik})$ are the attention weights. We recompute p_{ij} on-the-fly using the stored `logsumexp` values:

$$p_{ij} = \exp\left(s_{ij} - \text{logsumexp}_i\right). \tag{S5}$$

The gradients for query, key, and value are:

$$\frac{\partial \mathcal{L}}{\partial q_i} = \sum_j \frac{\partial \mathcal{L}}{\partial s_{ij}} \cdot k_j, \tag{S6}$$

$$\frac{\partial \mathcal{L}}{\partial k_j} = \sum_i \frac{\partial \mathcal{L}}{\partial s_{ij}} \cdot q_i, \tag{S7}$$

$$\frac{\partial \mathcal{L}}{\partial v_j} = \sum_i p_{ij} \cdot \frac{\partial \mathcal{L}}{\partial o_i}. \tag{S8}$$

By iterating over key-value pairs (instead of queries), gradients $\partial\mathcal{L}/\partial k_j$ and $\partial\mathcal{L}/\partial v_j$ can be written directly without atomics, while only query gradients require atomic accumulation.

S4.3 Input Transforms

We apply the following transforms to input layers before feeding them into the network. These transforms normalize the wide dynamic range of rendering data to improve network training. Table S1 summarizes the transforms applied to each input layer.

Log1p Transform. For HDR radiance values (diffuse and specular), we apply a `log1p` transform to compress the dynamic range:

$$\text{log1p}(x) = \log(1 + x). \tag{S9}$$

The output is then clipped to $[0, 6]$ to prevent extreme outliers.

Table S1. Input feature transforms. Frame ID is only used in the cross-frame model.

Source	In	Transform Pipeline	Out
Diffuse	3	`log1p` → `clip[0,6]`	3
		`unpremult` → `log1p` → `clip[0,6]`	3
Specular	3	`log1p` → `clip[0,6]`	3
		`unpremult` → `log1p` → `clip[0,6]`	3
Albedo	3	`unpremult` → `clip[0,6]`	3
Normal	3	`unpremult` → `clip[-1,1]`	3
Depth	1	`reciprocal` → `sineEnc(24)`	24
		`reciprocal`	1
Alpha	1	`clip[0,1]`	1
		`overToAdd`	1
	14	*Single-frame total*	*45*
Frame ID	1	`oneHot(7)`	7
	15	***Cross-frame total***	***52***

Unpremultiply Alpha. Layers stored in premultiplied form are converted to straight alpha:

$$\text{unpremult}(x, \alpha) = \begin{cases} x/\alpha & \text{if } \alpha > \epsilon \\ x & \text{otherwise} \end{cases}, \tag{S10}$$

where ϵ is a small threshold to avoid division by zero; in the small-alpha case, the original value is kept to preserve the invertibility of the transform.

Reciprocal Depth. Depth values span a large range from near to far planes. We apply a reciprocal transform:

$$\text{reciprocal}(z) = \frac{1}{1+z}, \tag{S11}$$

which maps infinite depth to 0 and near-plane depths closer to 1.

Sinusoidal Positional Encoding. To further enhance depth representation, we apply sinusoidal positional encoding [Vaswani et al. 2017] with exponentially-spaced frequencies:

$$\text{sineEnc}(z, L) = [\sin(2\pi f_1 z), \cos(2\pi f_1 z), \ldots, \sin(2\pi f_L z), \cos(2\pi f_L z)], \tag{S12}$$

where frequencies $f_\ell = f_{\min} \cdot (f_{\max}/f_{\min})^{(\ell-1)/(L-1)}$ are spaced exponentially between $f_{\min}$ and $f_{\max}$. In our configuration, we use $L = 12$ frequency pairs (24 output channels) with $f_{\min} = 10^{-5}$ and $f_{\max} = 1$.

Alpha Transforms. Alpha values are clipped to $[0, 1]$ and converted from over-compositing to additive form [Vicini et al. 2019]. In over-compositing, each bin's alpha a_i represents its opacity, and bins occlude those behind them. We convert to additive alpha b_i, which represents each bin's actual contribution to the final composited result:

$$b_i = a_i \cdot \prod_{j<i} (1 - a_j), \tag{S13}$$

where the product term is the transmittance of all bins in front of bin i. Such a transformation is helpful because the additive alpha reflects the sample count for each bin, and subsequently the reliability of each bin's noisy values.

S4.4 Architecture Configuration

Table S2 summarizes the architectural parameters for both our single-frame and cross-frame RaNAD models. We use fewer reconstruction scales for less noisy layers: 3 scales for alpha and 2 scales for depth, compared to 4 scales for diffuse and specular. The cross-frame model uses a smaller backbone with fewer U-Nets and attention heads to accommodate the increased memory requirements of processing multiple frames.

Table S2. Architecture and training-input configuration for single-frame and cross-frame RaNAD models.

Parameter	Single-Frame	Cross-Frame
Number of U-Nets	5	4
Scales per U-Net	3	3
U-Net dropout rate	0.75	0.75
Attention heads	8	6
Query/Key depth	32	32
Value depth	32	32
Embedding depth	256	192
Reconstruction scales (diffuse)	4	4
Reconstruction scales (specular)	4	4
Reconstruction scales (alpha)	3	3
Reconstruction scales (depth)	2	2
Reconstruction dropout rate	0.5	0.5
Temporal window size	—	7
Training crop size	128×128	112×112
Training batch size	1	1

The next two figures detail the cross-frame submodules omitted from the main paper for brevity. Figure S5 shows the temporal mixing module inserted between frame-parallel U-Nets, and Figure S6 shows the two-stage temporal-then-spatial reconstruction used for each output layer.

S4.5 Training Hyperparameters

We train our RaNAD models on a single NVIDIA RTX A6000 GPU (48 GiB VRAM). All models use AdamW [Loshchilov and Hutter 2019] with AMSGrad [Reddi et al. 2018] and a global gradient clipping norm of 0.01. Training is patch-based rather than full-image based, with crop sizes and batch sizes listed in Table S2. We train for 1048576 iterations. The learning rate follows a piecewise-constant schedule (Figure S7), with a warm-up phase followed by staged decay.

S4.5.1 Training Stability Techniques. We apply the following techniques consistently to both single-frame and cross-frame RaNAD to improve training stability and prevent early softmax collapses [Zhang et al. 2021]:

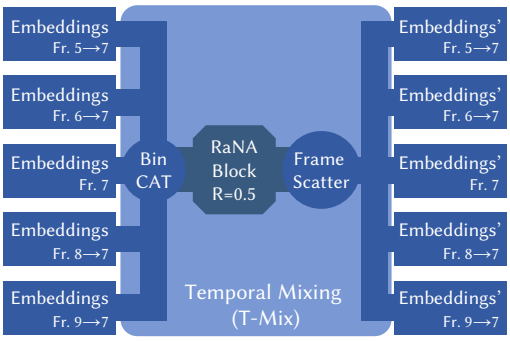


Fig. S5. Temporal mix (T-Mix) module. A RaNA block attends only along the temporal neighborhood of each pixel, enabling information exchange across frames without immediately mixing unrelated spatial neighborhoods.

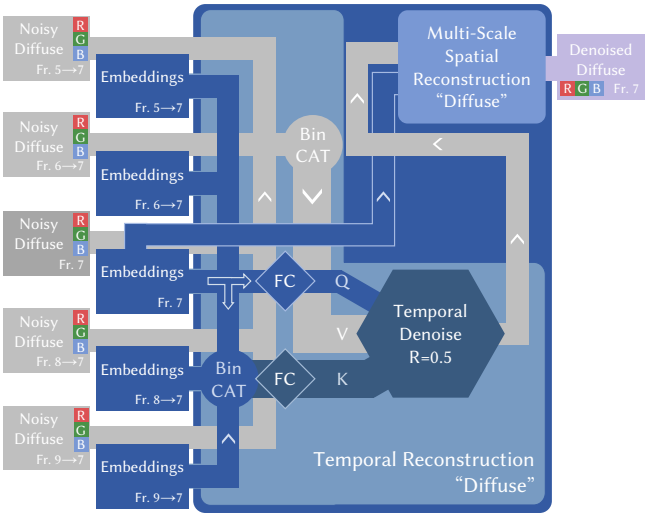


Fig. S6. Spatiotemporal reconstruction, using the diffuse layer as an example. The noisy diffuse layers from all frames first pass through a temporal reconstruction submodule, producing a temporally denoised center-frame diffuse layer with the help of the learned embeddings. This intermediate result is then refined by a spatial reconstruction submodule to produce the final denoised diffuse layer of the center frame.

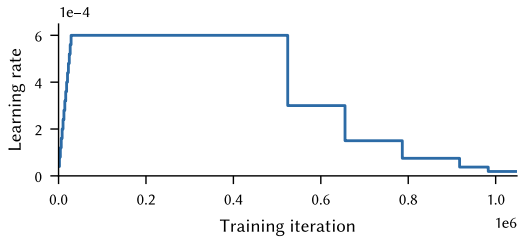


Fig. S7. Learning rate schedule used for training RaNAD models.

- Gradient clipping with norm threshold 0.01.

- L_2 regularization on FC layer weights with $\lambda = 2 \times 10^{-7}$, decayed by $1/\sqrt{2}$ after 384K iterations.
- Variance-scaling initialization (0.1× scale) for FC layer weights.

We also employ staged training of the multi-scale reconstruction: training starts with only downsampled data and gradually phases in per-scale denoising via sigmoid crossfading, fully replacing downsampling after 16384 iterations. The learned scale-mixing results are then phased in similarly, fully replacing the outputs from uniform mixing weights after 49152 iterations. The models are trained end-to-end thereafter.

S4.6 Training Losses

Table S3 summarizes the losses used for each output layer. Bin-wise losses are computed per bin before compositing, while composited losses operate on the alpha-over composited result.

Table S3. Training losses for each output layer. Bin-wise losses are computed per bin before compositing, while composited losses operate on the alpha-over composited result. Weights are shown in parentheses where applicable.

Layer	Bin-wise Loss	Composited Loss
Diffuse	SMAPE	ꟻLIP* (×10)
Specular	SMAPE	ꟻLIP* (×10)
Alpha	L1 (×10)	—
Depth	log1p ∘ SMAPE	—

S4.6.1 SMAPE Loss. For radiance and depth layers, we use the Symmetric Mean Absolute Percentage Error (SMAPE) [Flores 1986]:

$$\mathcal{L}_{\text{SMAPE}}(r, p) = \frac{|r - p|}{|r| + |p| + \varepsilon}, \tag{S14}$$

where r is the reference, p is the prediction, and $\varepsilon = 10^{-3}$ prevents division by zero. SMAPE provides scale-invariant gradients, making it well-suited for HDR content where pixel values span many orders of magnitude.

For depth, we apply a $\texttt{log1p}$ transform before SMAPE to further compress the range:

$$\mathcal{L}_{\text{depth}}(r, p) = \mathcal{L}_{\text{SMAPE}}(\text{log1p}(r), \text{log1p}(p)). \tag{S15}$$

S4.6.2 Modified FLIP for Training. While we use ꟻLIP [Andersson et al. 2020] as a perceptual loss, we modify it for gradient stability during backpropagation. The original ꟻLIP computes:

$$\text{ꟻLIP}_{\text{orig}} = \Delta_{\text{color}}^{1-\Delta_{\text{feature}}}, \tag{S16}$$

where Δ_{color} and Δ_{feature} are color and edge/point detection errors, respectively. This formulation can produce unstable gradients when $\Delta_{\text{feature}} \approx 1$ or when errors are very small.

Our training-friendly variant uses a weighted sum:

$$\text{ꟻLIP}_{\text{train}} = \Delta_{\text{color}} + \lambda \cdot \Delta_{\text{feature}}, \tag{S17}$$

with $\lambda = 0.1$ to balance the contributions. Additionally:

- Color error uses linear normalization instead of piecewise power mapping.
- Feature errors are summed ($\Delta_{\text{edge}} + \Delta_{\text{point}}$) instead of using max.
- All operations are differentiable with bounded gradients.

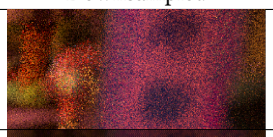
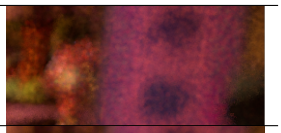
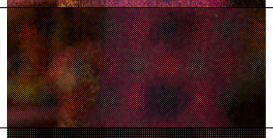
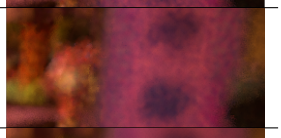
These modifications preserve the perceptual motivation of ꟻLIP while ensuring stable optimization.

S5 Additional Methodology Details

S5.1 Multi-Scale Reconstruction Visualization

Table S4 visualizes the intermediate outputs of our 4-scale reconstruction module. The middle column shows the down-sampling outputs at each scale, with increasing sparsity as bins are progressively dropped. The rightmost column shows the per-scale denoised outputs, all at full resolution but with different effective kernel sizes.

Table S4. Flattened output of the down-sampling and denoising stages of a 4-scale reconstruction module with 50% dropout. The middle column shows each scale's down-sampling output with increasing sparsity. The rightmost column shows the per-scale denoised outputs at full resolution. © Disney

Scale	Downsampled	Per-Scale Denoised
0		
1		
2		
3		

S6 Additional Quantitative Results

This section presents additional quantitative comparisons that complement the main paper results.

S6.1 Dataset B Quantitative Comparisons

Figure S8 and Figure S9 provide the additional Dataset B metrics corresponding to the Dataset A1 comparisons in the main paper.

S6.2 Comparison with Intel Open Image Denoise

To contextualize the quality of KPAL (our flat denoising baseline), we compare it against Intel Open Image Denoise (OIDN v2.4.1), a widely-used open-source denoiser. Figure S10 and Figure S11 show that KPAL-1f consistently outperforms OIDN across both test datasets, establishing it as a high-quality baseline for flat image denoising.

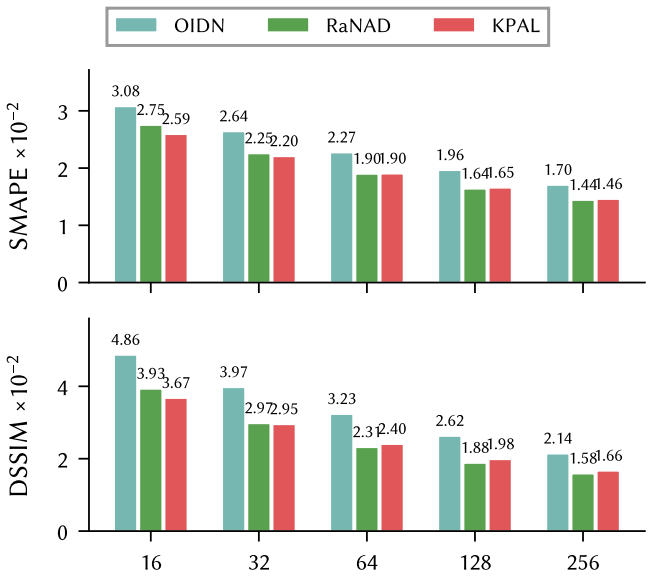


Fig. S8. Single-frame flattened denoising comparison on Dataset B. Lower is better for all metrics.

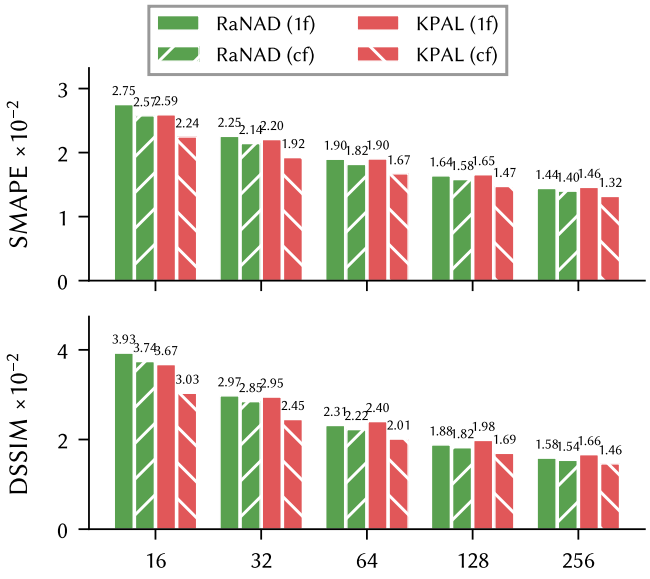


Fig. S9. Cross-frame flattened denoising comparison on Dataset B. Hatched bars indicate cross-frame versions, and lower is better for all metrics.

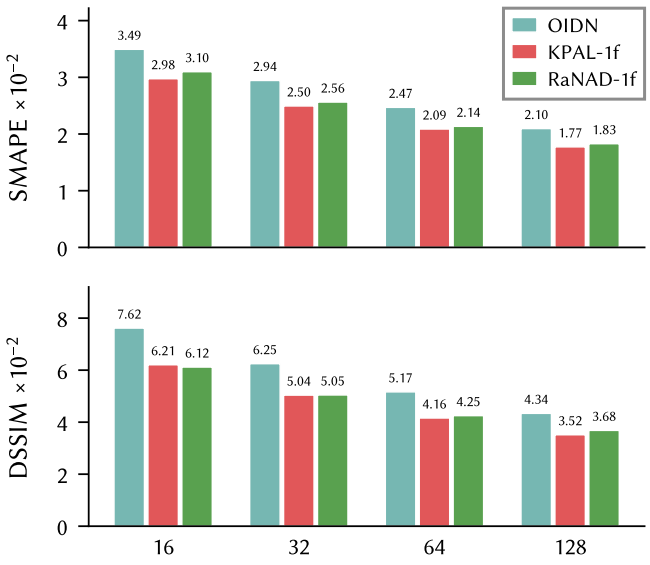


Fig. S10. Comparison of flat denoising methods on Dataset A1 across 16–128 spp. KPAL-1f outperforms OIDN, and RaNAD-1f matches KPAL-1f quality. Lower is better for all metrics.

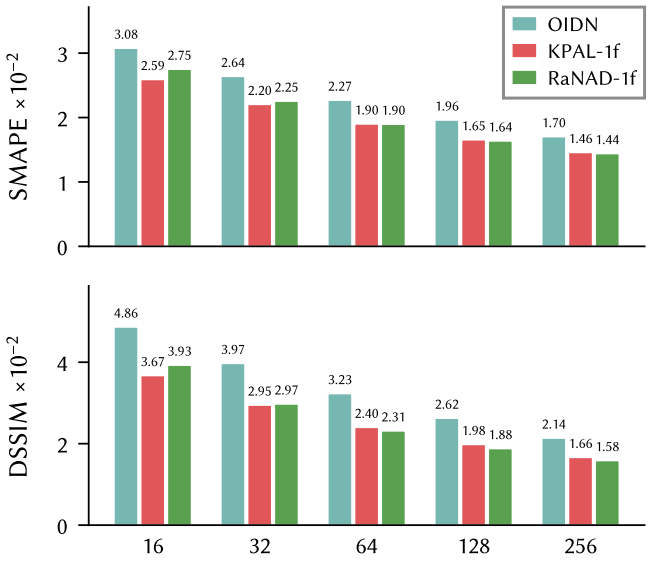


Fig. S11. Comparison of flat denoising methods on Dataset B. KPAL-1f significantly outperforms OIDN, while RaNAD-1f achieves comparable quality despite being trained for deep images. Lower is better for all metrics.

S6.3 Bin Preservation Analysis

Deep images enable clipping away bins from the foreground or the background without introducing artifacts. We evaluate the deep image denoisers' capability of preserving depth separation by progressively moving a virtual near or far clip plane through the scene, removing a percentage of bins from the denoised and reference images, then flattening the results and computing metrics. Figure S12 shows how flat denoising metrics change with respect to the percentage of bins preserved. RaNAD-1f produces lower errors at all bin preservation percentages.

S6.4 Deep-Z vs Deep-ObjectID Comparison

RaNAD models are inherently capable of denoising both deep-Z and deep-ObjectID images. Figure S13 confirms the versatility of

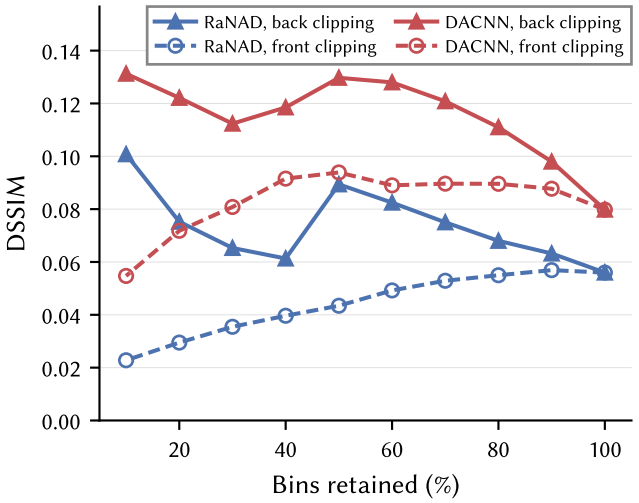


Fig. S12. Bin preservation metrics showing denoising quality with respect to percentage of bins preserved. Back clipping removes distant bins (moving the far clip plane closer); Front clipping removes nearby bins (moving the near clip plane farther). RaNAD-1f maintains lower error across all retention levels.

RaNAD—denoising single-frame deep-Z or deep-ObjectID images produce almost the same flat denoising metrics. Cross-frame deep-ObjectID experiments are omitted because matching temporal deep-ObjectID evaluation windows were not available in our current data setup; this is a data limitation rather than an architectural limitation of RaNAD.

S6.5 Multi-Scale Reconstruction Ablation

Figure S14 compares single-scale reconstruction (with a large radius $R = 8.5$) against our multi-scale reconstruction module (4 scales). Multi-scale reconstruction provides slight quality improvements while reducing computation cost by up to 20% across our test scenes, especially for images with many bins.

S6.6 Ablation: Single-U-Net vs. Multi-U-Net

We compare the chained Multi-U-Net backbone against a capacity-matched single-U-Net variant on the flattened single-frame evaluation. The capacity-matched single-U-Net slightly outperforms the chained Multi-U-Net on both datasets. On Dataset A1, averaged over 32–128 spp, it is about 3.6% lower in SMAPE and 6.4% lower in DSSIM. Dataset B shows the same trend, with about 1.3% lower SMAPE and 2.1% lower DSSIM across 32–256 spp, so the Multi-U-Net should be understood mainly as the building block chosen for the later spatiotemporal extension. Figure S15 and Figure S16 report the flattened-image metrics for Datasets A1 and B, respectively.

A depth-clipped bin-preservation analysis on Dataset A1 adds a small nuance to this picture. The capacity-matched single-U-Net remains lower in DSSIM for all back-clipping levels and for front clipping down to about 50% bins retained, but the chained Multi-U-Net becomes slightly better once only 40% or fewer front bins remain. This suggests that the Multi-U-Net may preserve a small advantage

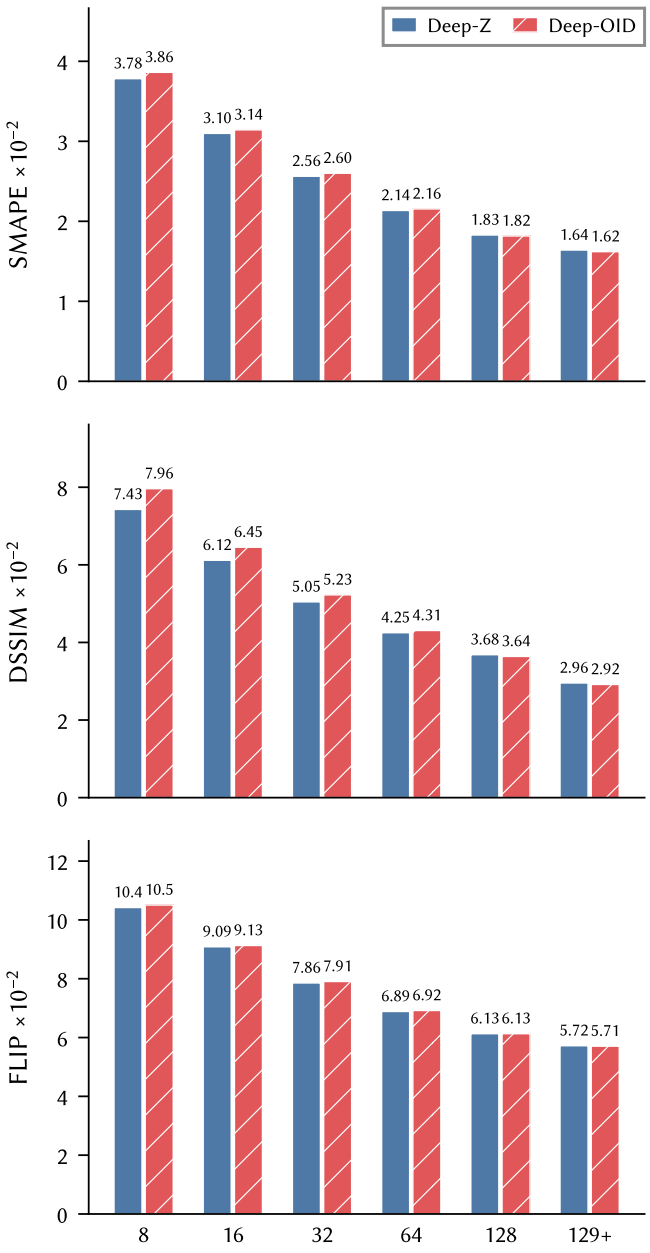


Fig. S13. Comparison of RaNAD-1f denoising metrics on deep-Z vs. deep-ObjectID images. The nearly identical results demonstrate that RaNAD handles both depth representations equally well.

only for very aggressive foreground removal, even though the overall flattened-image metrics still favor the single-U-Net. Figure S17 shows this depth-clipped comparison.

S6.7 Denoising Timings

Figure S18 shows CPU vs. GPU denoising times for single-frame RaNAD. In contrast to flat image denoising where run time scales with pixel count, RaNAD's execution time correlates strongly with

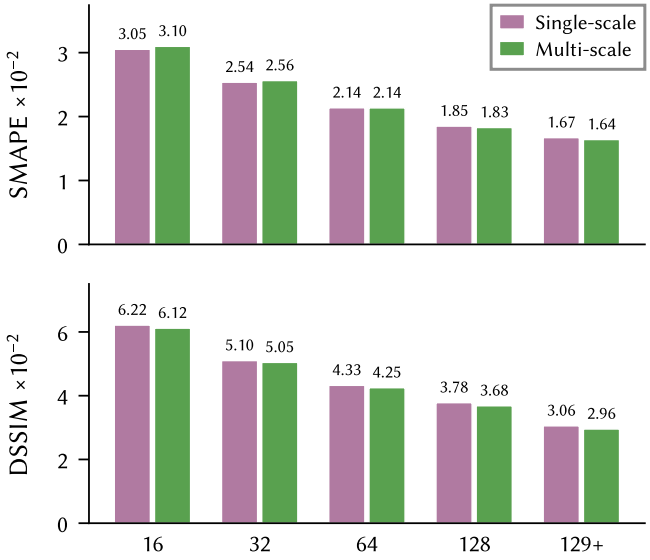


Fig. S14. Quality metrics comparing single-scale (R=8.5) vs. multi-scale (4 scales) reconstruction on Dataset A1. Multi-scale provides slight quality improvements while reducing computation cost by up to 20%. Lower is better for all metrics.

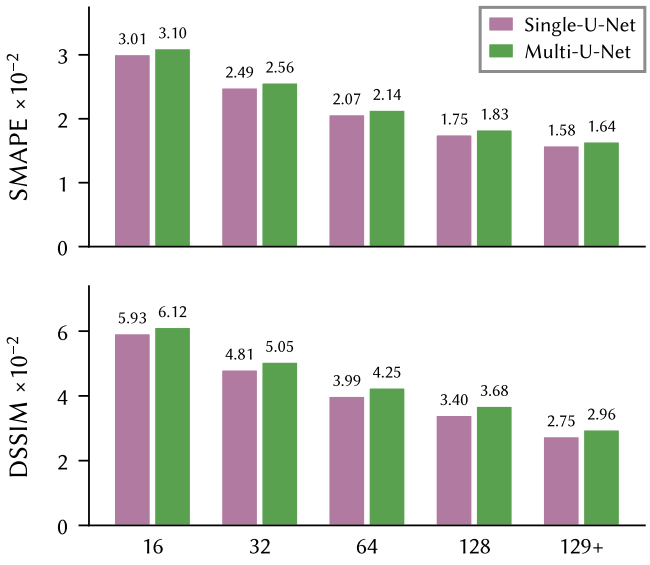


Fig. S15. Quality metrics comparing the capacity-matched single-U-Net and the chained Multi-U-Net backbone on Dataset A1. The single-U-Net yields slightly lower flattened-image error across most spp levels. Lower is better for both metrics.

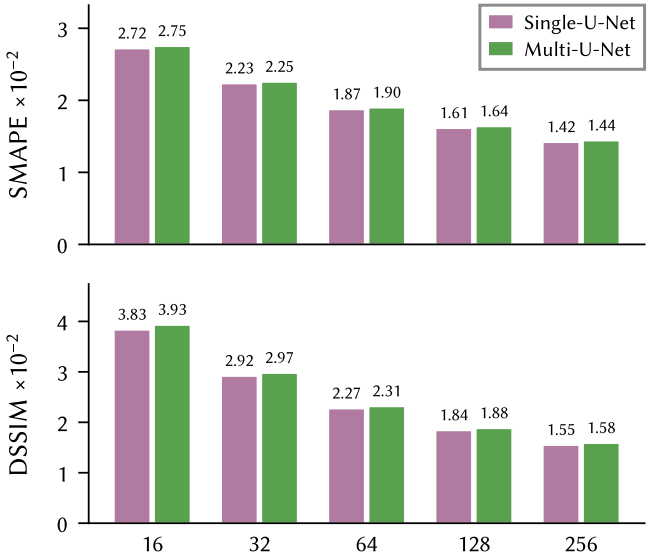


Fig. S16. Quality metrics comparing the capacity-matched single-U-Net and the chained Multi-U-Net backbone on Dataset B. The single-U-Net again yields slightly lower flattened-image error across the displayed spp levels. Lower is better for both metrics.

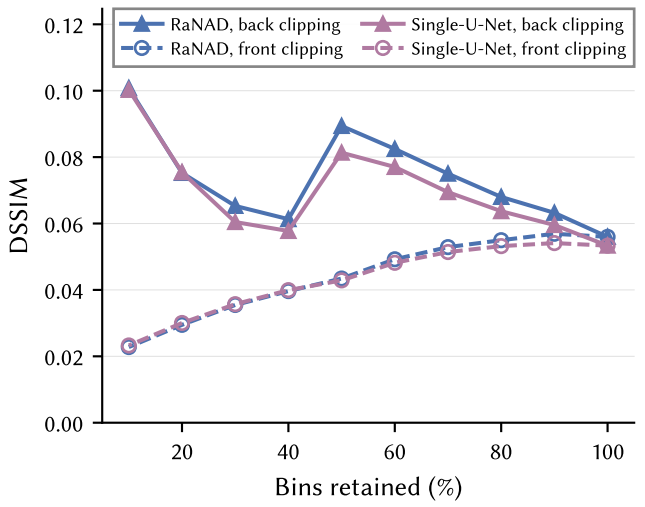


Fig. S17. Depth-clipped bin-preservation comparison for the capacity-matched single-U-Net and the chained Multi-U-Net on Dataset A1. The single-U-Net is lower for all back-clipping levels and for milder front clipping, while the Multi-U-Net becomes slightly better only once the foreground is clipped very aggressively. Lower is better.

the number of bins. GPU denoising is consistently 10–20× faster than CPU on our test hardware.

References

Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D. Fairchild. 2020. FLIP: A Difference Evaluator for Alternating Images. *Proc. ACM on Computer Graphics and Interactive Techniques* 3, 2, Article 15 (Aug. 2020), 23 pages. doi:10.1145/3406183

Tri Dao. 2023. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. arXiv:2307.08691. https://arxiv.org/abs/2307.08691 arXiv preprint.

Benito E Flores. 1986. A Pragmatic View of Accuracy Measurement in Forecasting. *Omega* 14, 2 (1986), 93–98. doi:10.1016/0305-0483(86)90013-7

Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations (ICLR)*. https://openreview.

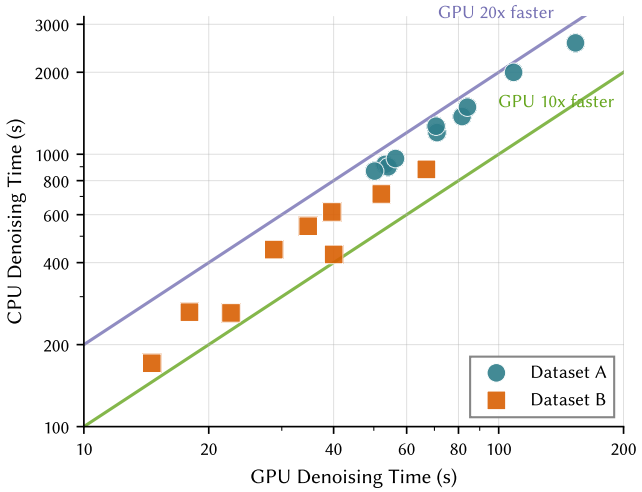


Fig. S18. Single-frame RaNAD denoising times on CPU vs. GPU for 19 test images from Datasets A1 and B. Reference lines indicate 10× and 20× GPU speedup. Timings measured on an AMD EPYC 7313P (16-core) CPU and NVIDIA GeForce RTX 3090 Ti GPU.

net/forum?id=Bkg6RiCqY7

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the Convergence of Adam and Beyond. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=ryQu7f-RZ

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

Delio Vicini, David Adler, Jan Novák, Fabrice Rousselle, and Brent Burley. 2019. Denoising Deep Monte Carlo Renderings. *Computer Graphics Forum* 38, 1 (2019), 316–327. doi:10.1111/cgf.13533

Thijs Vogels, Fabrice Rousselle, Brian McWilliams, Gerhard Röthlin, Alex Harvill, David Adler, Mark Meyer, and Jan Novák. 2018. Denoising with Kernel Prediction and Asymmetric Loss Functions. *ACM Trans. Graph.* 37, 4, Article 124 (July 2018), 15 pages. doi:10.1145/3197517.3201388

Xianyao Zhang, Marco Manzi, Thijs Vogels, Henrik Dahlberg, Markus Gross, and Marios Papas. 2021. Deep Compositional Denoising for High-quality Monte Carlo Rendering. *Computer Graphics Forum* 40, 4 (July 2021), 1–13. doi:10.1111/cgf.14337