

Log-linear Kernel Cluster Distillation for Lightweight Neural Monte Carlo Denoising

YE YUAN, ZIJUAN YUAN, JUNYUAN HE, ZIREN WANG, CHANGYUN ZHI, CHENXI ZHOU, BOSHAN LIU, YANWEN GUO, and JIE GUO^{✉*}, State Key Lab for Novel Software Technology, Nanjing University, China

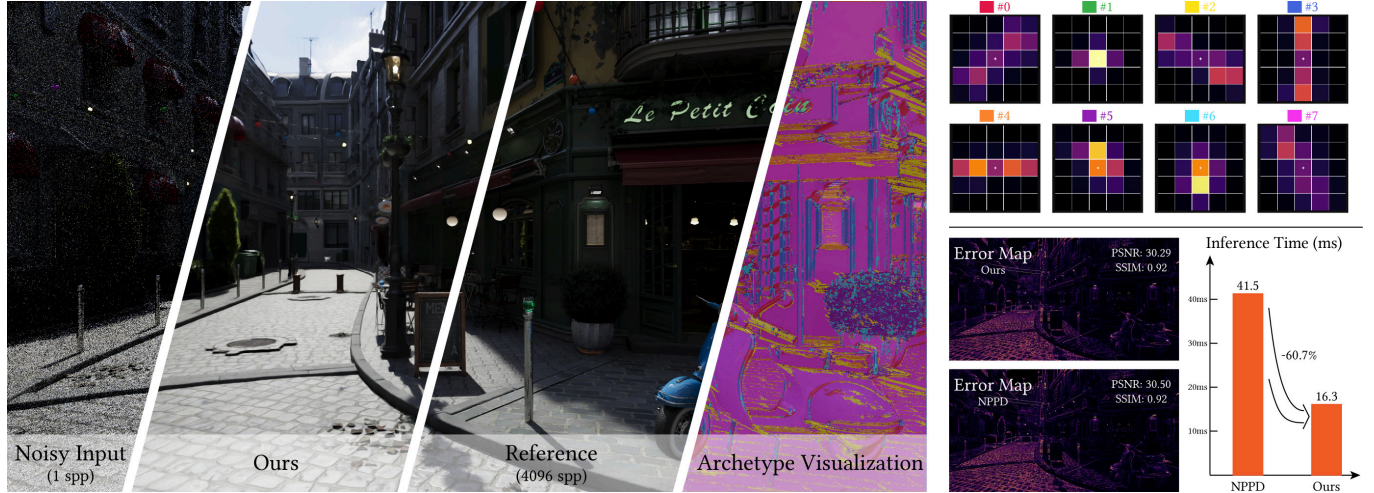


Fig. 1. The proposed *Log-Linear Kernel Cluster Distillation* framework aims to compress the cumbersome output head of kernel-prediction Monte Carlo denoisers by identifying sparse *archetypes* (shown in the upper-right panel) in a high-dimensional *denoising kernel space*. By predicting the weights of these archetypes and the weights of residual offsets, we can significantly improve the efficiency of existing kernel-prediction denoisers (e.g., NPPD [Balint et al. 2023]) while preserving high image quality. *Bistro* © Amazon Lumberyard.

Kernel prediction networks have gained widespread popularity for Monte Carlo denoising, due to their ability to preserve fine details without hallucinating non-existent features. However, the requirement to explicitly predict per-pixel spatially varying kernels imposes a heavy burden on memory and computation. We observe that the predicted kernels are not arbitrary: despite their high dimensionality, they exhibit recurring spatial patterns and concentrate on a compact region of the probability simplex. This suggests that kernel prediction networks can be compressed in the space of filtering behavior rather than only in the space of network weights. Motivated by this observation, we propose *Log-linear Kernel Cluster Distillation*, a distillation framework that clusters denoising kernels predicted by a high-quality teacher model into representative archetype patterns and trains a lightweight student model to predict archetype mixtures refined by residual offsets through a log-linear parameterization. The student model is supervised heterogeneously by both rendered outputs and normalized kernels,

encouraging it to reproduce the teacher model’s filtering strategy while retaining a compact prediction head. Our formulation decouples kernel expressiveness from output dimensionality, reducing end-to-end runtime by 60.7% relative to the 15M NPPD teacher at comparable image quality.

Additional Key Words and Phrases: real-time ray tracing, Monte Carlo denoising, kernel prediction, knowledge distillation

ACM Reference Format:

Ye Yuan, Zijuan Yuan, Junyuan He, Ziren Wang, Changyun Zhi, Chenxi Zhou, Boshan Liu, Yanwen Guo, and Jie Guo. 2026. Log-linear Kernel Cluster Distillation for Lightweight Neural Monte Carlo Denoising. In *SIGGRAPH Asia 2026 Conference Papers (SA Conference Papers ’26)*, December 01–04, 2026, Kuala Lumpur, Malaysia. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3829340.3842233>

*Corresponding author.

Authors’ Contact Information: Ye Yuan, 502024330061@smail.nju.edu.cn; Zijuan Yuan, 522025320206@smail.nju.edu.cn; Junyuan He, 221240065@smail.nju.edu.cn; Ziren Wang, 221220149@smail.nju.edu.cn; Changyun Zhi, 241220044@smail.nju.edu.cn; Chenxi Zhou, chenxizhou@smail.nju.edu.cn; Boshan Liu, 231220035@smail.nju.edu.cn; Yanwen Guo, ywguo@nju.edu.cn; Jie Guo, guojie@nju.edu.cn, State Key Lab for Novel Software Technology, Nanjing University, Nanjing, China.



This work is licensed under a Creative Commons Attribution 4.0 International License. *SA Conference Papers ’26, Kuala Lumpur, Malaysia*
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2842-6/2026/12
<https://doi.org/10.1145/3829340.3842233>

1 Introduction

Real-time ray tracing is now a standard rendering technique in modern interactive graphics applications, yet frame-rate budgets severely restrict sample counts, resulting in substantial Monte Carlo noise [Christensen and Jarosz 2016; Pharr et al. 2023]. Consequently, neural Monte Carlo denoising has recently become a core component of the real-time rendering pipeline. Among neural approaches, kernel prediction [Bako et al. 2017; Balint et al. 2023; Gharbi et al. 2019; Vogels et al. 2018] has emerged as the dominant paradigm: instead of regressing pixel colors directly [Chaitanya et al. 2017],

the network predicts per-pixel filtering weights that combine neighboring samples, which preserves radiance consistency and keeps the output free of hallucinated structures.

Deploying kernel-prediction denoisers under strict real-time budgets, however, is limited by the output of the prediction head rather than by the backbone alone. An $n \times n$ predicted kernel entails n^2 output channels per pixel, and modern pyramid denoisers typically predict several kernel families across multiple pyramid levels, resulting in a cumbersome output head. For instance, NPPD [Balint et al. 2023] produces 221 channels across its five levels, of which 214 are kernel coefficients. Since each output head is produced by a 1×1 convolution from the decoder’s feature maps, the decoder channel width at every level is lower-bounded by the output head’s dimensionality. This bound is largely decoupled from backbone capacity, making the output head a structural performance bottleneck for kernel-prediction networks.

In this paper, we first revisit this bottleneck by analyzing the denoising kernels and their spanned space (typically forming a simplex) produced by a trained kernel-prediction network. Applying archetypal analysis [Alcacer et al. 2025; Cutler and Breiman 1994] to this denoising kernel space, we observe that the normalized kernels actually concentrate on a compact region of the simplex: a small set of archetypes and their convex combinations reconstruct the observed kernels with negligible error across scenes. This suggests that the prediction head can output compact mixture weights over a shared set of archetypes rather than full kernel coefficients.

Motivated by this observation, we propose *Log-linear Kernel Cluster Distillation*, a practical distillation framework that compresses the output head of kernel-prediction denoisers while preserving the denoising quality of a large teacher model (e.g., NPPD). Instead of predicting n^2 output channels for an $n \times n$ kernel, a lightweight student model only needs to predict two low-dimensional sets of mixture weights for the generated archetypes and residual offsets, respectively. A log-linear expansion module then fuses these two sets to reconstruct the full denoising kernels before filtering. Consequently, the output dimensionality reduces to the sum of the archetype and residual counts, a dramatic reduction from the quadratic footprint of direct kernel prediction.

In summary, our contributions are:

- A structural analysis of the kernel-prediction output head, identifying it as a decoder-width bottleneck that is largely independent of backbone compression, and an empirical analysis showing that normalized kernels concentrate within a compact region of the simplex admitting an archetype-based representation.
- A log-linear kernel expansion method that reconstructs full denoising kernels from two groups of predicted weights over offline-generated archetypes and residual offsets, reducing the output dimension from the quadratic coefficient count of direct kernel prediction to two low-dimensional mixture dimensions.
- A distillation framework built on this expansion, which integrates an offline two-stage archetype generation procedure, a kernel-specific distillation loss, and a freeze-then-finetune

schedule. It reduces end-to-end runtime by 60.7% relative to the 15M NPPD teacher at comparable image quality.

2 Related Work

2.1 Kernel Prediction for Monte Carlo Denoising

To meet strict real-time budgets, non-neural denoising filters have long been the mainstream solution for interactive Monte Carlo path tracing [Li et al. 2012; Schied et al. 2017; Sen and Darabi 2012; Zwicker et al. 2015]. Recently, the NVIDIA Real-time Denoisers (NRD) [NVIDIA 2022] has become a widely adopted industrial pipeline for high-efficiency spatio-temporal denoising. While these non-neural methods deliver consistent runtime performance, they inevitably suffer from over-smoothing and detail loss in complex scenes, motivating the development of neural alternatives [Yan et al. 2025].

Among neural denoising methods, kernel prediction has emerged as a dominant paradigm due to its effectiveness in preserving high-frequency details under severe noise. The seminal KPCN [Bako et al. 2017] established this paradigm by demonstrating that deep neural networks can learn spatially varying kernels from noisy radiance and G-buffer features, laying the foundation for this field. Subsequent researches further improved the kernel prediction pipeline. Early efforts optimized network structures, parameter efficiency and multi-task fusion for real-time performance [Cai et al. 2021; Fan et al. 2021; Meng et al. 2020; Thomas et al. 2022]. Neural partitioning pyramids (NPPD) [Balint et al. 2023] mitigated low-frequency artifacts through a hierarchical kernel filtering design. Online denoising frameworks [Choi et al. 2024] enhanced generalization and temporal stability without relying on external pre-training datasets. Multi-branch multi-scale networks [Kaźmierczyk et al. 2025] further enabled efficient joint denoising and upscaling. However, dense full-resolution kernel outputs remain costly, motivating compression of the output representation.

2.2 Knowledge Distillation and Kernel Compression

Knowledge distillation compresses deep neural networks by transferring knowledge from a teacher model to a smaller student [Hinton et al. 2015]. For dense prediction, Liu et al. [2023] proposed structured knowledge distillation to transfer pair-wise and holistic structural information. Hu et al. [2023] leveraged auxiliary unlabeled data to mitigate the teacher–student capacity gap in lightweight depth estimation. In the context of natural image denoising, Young et al. [2021] introduced feature alignment and feature matching losses for efficient nature denoising, whereas Zou et al. [2022] adapted the distillation target to the student architecture. Recently, within real-time Monte Carlo denoising, Kong et al. [2025] proposed render-aware knowledge distillation, combining local-to-global distillation, render-aware parameter generation, and auxiliary unlabeled data.

These methods mainly reduce the backbone cost or improve the supervision strategy, while keeping the output space unchanged. For kernel-prediction denoisers, the number of output channels scales with the kernel size, making the dense per-pixel kernel representation a significant computational burden. An alternative direction is to represent spatially varying kernels in a compact form. Existing works have explored several such parameterizations. Specifically,

Niklaus et al. [2017] factorized each adaptive 2D kernel into two 1D kernels, reducing the number of predicted weights from quadratic to linear in the kernel size. Carbajal et al. [2021] represented dense non-uniform blur kernels using image-adaptive basis motion kernels and pixel-wise mixing coefficients. In natural image denoising, Basis Prediction Networks [Xia et al. 2020] similarly used image-level basis kernels and per-pixel mixing coefficients to avoid predicting full per-pixel denoising kernels. Closest to our setting, Işık et al. [2021] derive filter weights implicitly from affinities between learned per-pixel embeddings. These methods demonstrate spatially varying kernels are highly redundant and can be compressed through structured output parameterization. Building upon this insight, our work applies archetypal analysis to denoising kernels and leverages the representative kernel archetypes to compress the kernel-prediction output space. Our teacher-derived components are shared within each kernel family and combine archetype and residual mixtures through a log-linear expansion.

3 Denoising Kernel Analysis

3.1 Performance Bottleneck of Kernel Prediction

Kernel prediction methods [Bako et al. 2017; Munkberg and Hasselgren 2020; Vogels et al. 2018] reconstruct denoised radiance by estimating spatially varying filtering kernels rather than directly regressing pixel colors. Given a noisy radiance image, the network predicts a *denoising kernel* $\mathbf{k}_p$ over a neighborhood $\mathcal{N}(p)$ at each pixel p , producing the reconstructed color as

$$\hat{\mathbf{c}}_p = \sum_{q \in \mathcal{N}(p)} k_{p,q} \mathbf{c}_q, \quad \sum_{q \in \mathcal{N}(p)} k_{p,q} = 1. \quad (1)$$

These methods are output-channel intensive: an $n \times n$ filter requires n^2 output channels per pixel. Recent denoisers [Balint et al. 2023; Thomas et al. 2022] predict several distinct filters and blend weights across multiple resolution levels. Taking NPPD [Balint et al. 2023] as an example, its five-level pyramid denoising predictor predicts 5×5 spatial kernels at all five levels, 5×5 temporal kernels at the finest level, 4×4 upsampling kernels at the four coarser levels, 1-channel blend weight for previous noisy color, 1-channel blend weight for previous output and 5-channel split weights for pyramid partition.

We refer to each such filter type as a *kernel family* f , and denote the set of kernel families at level ℓ as $\mathcal{F}_\ell$. The total output dimension at level ℓ is then

$$D_\ell = \sum_{f \in \mathcal{F}_\ell} D_f + B_\ell, \quad (2)$$

where D_f is the number of kernel weights in family f and B_ℓ is the number of scalar channels like blend/partition weights at level ℓ .

Figure 2 shows that kernel prediction at the finest pyramid level dominates the runtime cost, as this level combines the highest spatial resolution with the largest output dimension D_0 . Varying the spatial and temporal kernel size at this level primarily impacts the prediction cost rather than the filtering cost, confirming that convolution at the finest layer constitutes the computational bottleneck.

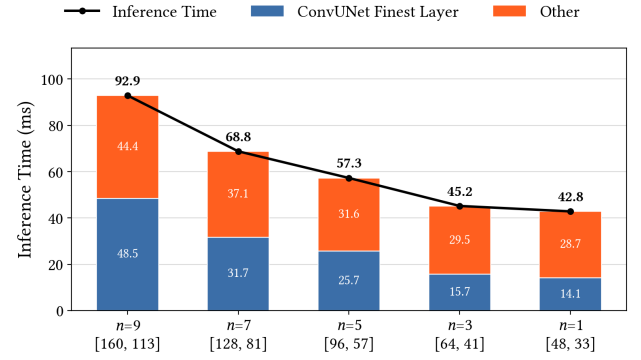


Fig. 2. Runtime breakdown of NPPD on an NVIDIA RTX 4090 GPU (FP32) under varying kernel size (n) at the finest pyramid level, with kernel fusion and operator-level optimizations disabled to isolate individual kernel costs. Stacked bars show average CUDA time for finest-level prediction (**ConvUNet Finest Layer**) and all remaining components (**Other**). Bracketed numbers $[C_0, C_1]$ are the output channel counts at the finest and second levels. Convolution at the finest level forms the primary computational bottleneck, especially when the kernel size n is large.

3.2 Archetypal Analysis for Denoising Kernel Space

In this section, we will analyze the distribution of denoising kernels on a probability simplex, and try to find a compact subspace that is well approximated by a small set of archetypes. In this way, we only need to predict compressed weights of dimension $D'_\ell \ll D_\ell$ in the kernel space.

Kernel prediction methods normalize filter weights to sum to one for energy conservation, constraining each predicted kernel of family f to lie in the probability simplex

$$\mathcal{K}_f = \{ \mathbf{k} \in \mathbb{R}^{D_f} \mid \sum_i k_i = 1, k_i \geq 0 \}, \quad (3)$$

which we refer to as the *denoising kernel space*.

We analyze the distribution of trained kernels in $\mathcal{K}_f$ via *archetypal analysis* (AA) [Cutler and Breiman 1994]. For each kernel family f , AA learns a set of archetypes $\mathcal{A}_f = \{ \mathbf{a}_{f,1}, \dots, \mathbf{a}_{f,K_f} \}$ and represents every observed kernel as a convex combination of them,

$$\mathbf{k}_f \approx \sum_{i=1}^{K_f} \alpha_{f,i} \mathbf{a}_{f,i}, \quad (4)$$

s.t. $\mathbf{a}_{f,i} \in \mathcal{K}_f, \sum_{i=1}^{K_f} \alpha_{f,i} = 1, \alpha_{f,i} \geq 0.$

where each archetype $\mathbf{a}_{f,i}$ is itself a convex combination of the observed kernels.

Since convex combinations of normalized kernels preserve normalization, all archetypes and their resulting convex combinations reside within $\mathcal{K}_f$. Unlike hard clustering or unconstrained linear representations, AA expresses kernels as convex mixtures of normalized observations and therefore preserves $\mathcal{K}_f$ by construction. Further comparisons are provided in the supplementary material.

We assess AA as a compressed parameterization of $\mathcal{K}_f$ along two axes: a qualitative inspection of the learned archetypes, and a

quantitative measurement of the reconstruction error in Eq. (4) as a function of the archetype count K_f .

Figure 3 shows the qualitative result on 5×5 spatial kernels collected from trained NPPD [Balint et al. 2023]. The kernels partition into a small number of clusters, each aligned with a distinct archetype that concentrates weight on a localized spatial pattern. The four most frequent archetypes alone contain the majority of kernels, and their representative members deviate from the assigned archetype by only a small residual, suggesting that a few archetypes already account for most of the distribution.

In summary, the visualization of denoising kernels from NPPD confirms that these kernels actually concentrate within a compact subregion of $\mathcal{K}_f$, which is well represented by a small set of archetypes. We present our distillation framework based on this observation, in which the student model predicts archetype weights in $\mathcal{K}_f$ together with learned residuals, and recovers the full kernels by combining them.

4 Log-linear Kernel Cluster Distillation

This section introduces our log-linear kernel cluster distillation framework (Fig. 4), which combines offline kernel cluster generation with a lightweight kernel-prediction student. Given a pretrained teacher denoiser, we first collect its normalized pyramid kernels and generate *archetypes* $\mathbf{A}_f$ and *residual offsets* δ_f for each kernel family (Sec. 4.1). During training, the teacher and the student process the same inputs, while the student predicts archetype weights α_f and residual offset weights β_f over these generated components. We then use a log-linear expansion to reconstruct the full normalized kernels before the standard pyramid filtering step (Sec. 4.3). Finally, we train the student with image-space supervision and kernel-level distillation (Sec. 4.4).

4.1 Kernel Cluster Generation

The above analysis shows that normalized kernels in $\mathcal{K}_f$ concentrate around a small set of archetypes. While archetypes provide a compact representation of $\mathcal{K}_f$, a single archetype cannot fully capture the individual kernels within its cluster. We therefore generate two sets of components for each kernel family f . The archetypes $\mathbf{A}_f$ encode the dominant filtering behavior within each kernel cluster, and the residual offsets δ_f capture the finer variation around the archetype. Both the archetypes and residual offsets share the same dimension D_f as the original kernel of family f . We represent these components in the logarithm space, enabling the log-linear expansion of Sec. 4.3 to combine them additively prior to softmax normalization.

Stage 1: Archetypes. Following Sec. 3.2, we apply archetypal analysis to the kernel samples $\{\mathbf{k}_{f,n}\}$ predicted by the pretrained teacher model, obtaining $M_{a,f}$ archetypes for family f . We then take the element-wise logarithm of each archetype to form the archetype matrix $\mathbf{A}_f \in \mathbb{R}^{M_{a,f} \times D_f}$.

Stage 2: Residual offsets. For each sample $\mathbf{k}_{f,n}$, let $\mathbf{a}_{f,n}$ be the archetype to which Stage 1 assigns the largest weight for that sample. The log-space residual is

$$r_{f,n} = \log(\mathbf{k}_{f,n} + \epsilon) - \log(\mathbf{a}_{f,n} + \epsilon), \quad \epsilon = 10^{-6}. \quad (5)$$

We then apply a second archetypal analysis with $M_{\delta,f}$ archetypes to the residuals $\{r_{f,n}\}$ to form the residual offset matrix $\delta_f \in \mathbb{R}^{M_{\delta,f} \times D_f}$.

We fit AA independently for each kernel family using teacher samples pooled across its pyramid levels. The samples are collected from 128 training scenes, with 16 frames per scene at 256p, yielding 320M spatial, 80M temporal, and 250M upsampling samples. The constrained optimization details are provided in the supplementary material. We set $(M_{a,f}, M_{\delta,f})$ to (8, 4) for spatial kernels and (4, 4) for both temporal and upsampling kernels.

4.2 Distillation Network Design

Teacher. Our distillation framework does not depend on a specific teacher architecture. It only assumes that, for each kernel family f , the teacher predicts kernels in the denoising kernel space $\mathcal{K}_f$ defined in Sec. 3.2. In the current implementation, we instantiate the teacher with NPPD [Balint et al. 2023], whose kernel prediction module satisfies this requirement and provides a strong baseline for Monte Carlo denoising.

Student. The student follows the same pyramid filtering structure as the teacher. We group the predicted filters into three kernel families: spatial kernels, temporal kernels, and upsampling kernels. The same family definition is used across pyramid levels. Empirically, sharing one set of archetypes and residual offsets within each family is sufficient to represent kernels from different levels.

The architectural change is at the kernel-prediction decoder’s output. Instead of predicting all D_f kernel weights for each active family, the student predicts low-dimensional mixture weights over the shared family-level archetypes and residual offsets.

Let $\mathcal{F} = \{\text{spatial, temporal, upsampling}\}$ denote the three kernel families, and let $\mathcal{F}_\ell \subseteq \mathcal{F}$ be the families used at pyramid level ℓ . For each $f \in \mathcal{F}_\ell$, the student predicts archetype mixture weights $\alpha_{\ell,f} \in \mathbb{R}^{M_{a,f}}$ and residual mixture weights $\beta_{\ell,f} \in \mathbb{R}^{M_{\delta,f}}$, both non-negative and summing to one. The log-linear kernel expansion then combines these weights with $\mathbf{A}_f$ and δ_f to produce the full pyramid kernels $\mathbf{k}_{\ell,f} \in \mathcal{K}_f$.

For the teacher, the decoder output dimension at pyramid level ℓ is D_ℓ in Eq. (2). Under the log-linear expansion, each D_f is replaced by $M_{a,f} + M_{\delta,f}$. Therefore, the student’s output dimension at level ℓ becomes

$$D_\ell^{\text{cluster}} = \sum_{f \in \mathcal{F}_\ell} (M_{a,f} + M_{\delta,f}) + B_\ell, \quad (6)$$

with the largest reduction at the finest level.

4.3 Log-linear Kernel Expansion

Given archetypes $\mathbf{A}_f$ and residual offsets δ_f from Sec. 4.1, and the student-predicted mixture weights $\alpha_{\ell,f}$ and $\beta_{\ell,f}$ from Sec. 4.2, the reconstructed kernel of family f at level ℓ is

$$\mathbf{k}_{\ell,f} = \text{softmax}(\alpha_{\ell,f}^\top \mathbf{A}_f + \beta_{\ell,f}^\top \delta_f) \in \mathcal{K}_f. \quad (7)$$

AA provides the offline initialization of the archetypes and residual offsets, whereas Eq. (7) defines the student’s forward mapping. At inference time, the student-predicted mixtures are combined in log space and normalized by softmax: the archetype mixture captures the overall filtering behavior, while the residual mixture refines

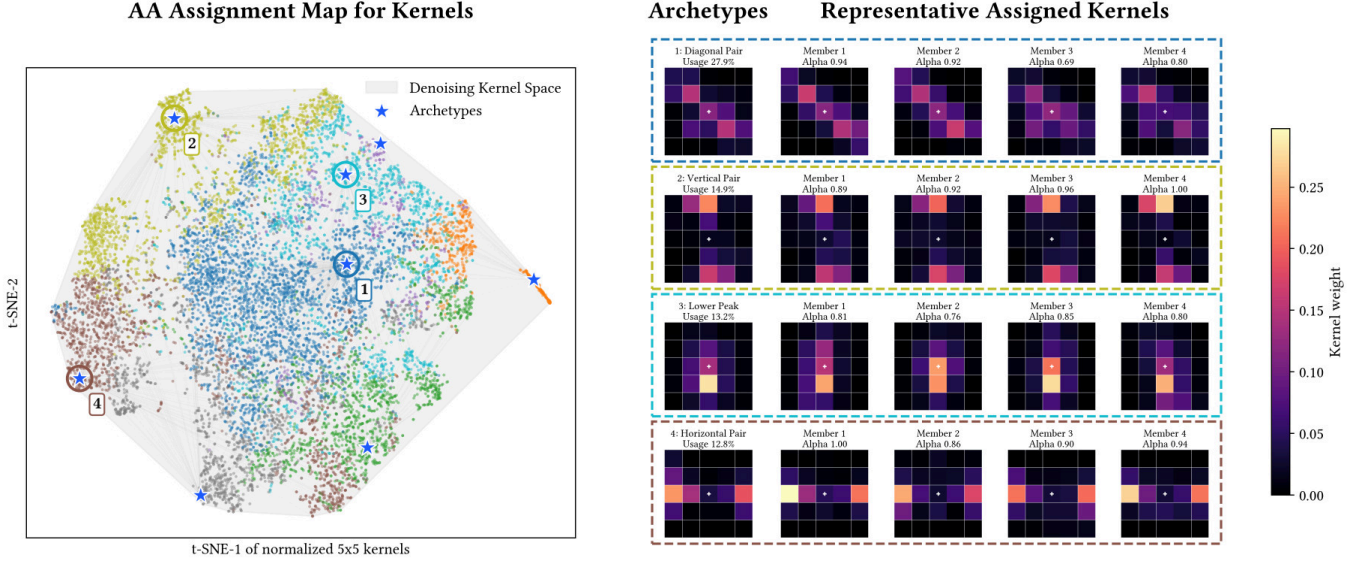


Fig. 3. Archetypal analysis groups 5×5 spatial kernels by filtering behavior. **Left:** t-SNE of 12k 5×5 spatial kernels collected from NPPD on the test set, colored by dominant archetype. Gray lines link samples to their archetypes (blue stars). **Right:** the four most frequent archetypes, each with four representative member kernels, annotated with usage share and dominant archetype weight $\alpha_{f,i}$.

element-wise variations. The student thus predicts only $M_{a,f} + M_{\delta,f}$ mixture weights per family, in place of the D_f kernel weights required by direct kernel prediction.

4.4 Training

Loss function. The student is trained with a combination of image-space supervision and teacher-guided distillation losses. The total objective is

$$\mathcal{L} = \lambda_s \mathcal{L}_{\text{sup}} + \lambda_o \mathcal{L}_{\text{kd-out}} + \lambda_k \mathcal{L}_{\text{kd-ker}}, \quad (8)$$

where λ_s , λ_o , and λ_k weight supervised reconstruction, output distillation, and kernel distillation, respectively. We follow the NPPD training objective [Balint et al. 2023] for $\mathcal{L}_{\text{sup}}$: a radiance-normalized SMAPE term,

$$\text{SMAPE}(\mathbf{u}, \mathbf{v}) = \frac{1}{|\Omega|} \sum_{i \in \Omega} \frac{|u_i - v_i|}{|u_i| + |v_i| + \epsilon}, \quad (9)$$

with $\epsilon=10^{-3}$, combined with the original spatial and temporal feature losses and the two-step BPTT temporal consistency loss [Balint et al. 2023].

Output distillation. $\mathcal{L}_{\text{kd-out}}$ compares the rendered student and teacher images under the same radiance normalization:

$$\mathcal{L}_{\text{kd-out}} = \text{SMAPE}(\mathcal{N}(\hat{\mathbf{y}}_t), \mathcal{N}_t(\hat{\mathbf{y}}_s)), \quad (10)$$

where $\hat{\mathbf{y}}_s$ and $\hat{\mathbf{y}}_t$ are the student and teacher denoised outputs, $\mathcal{N}$ denotes the NPPD radiance normalization, and $\mathcal{N}_t$ applies the teacher’s normalization scale to the student.

Kernel distillation. $\mathcal{L}_{\text{kd-ker}}$ supervises the student’s log-linear expansion directly in kernel space. It compares the student’s expanded

normalized kernels with the teacher’s softmax-normalized kernels over the shared pyramid,

$$\mathcal{L}_{\text{kd-ker}} = \frac{1}{L} \sum_{\ell=1}^L \frac{1}{|\mathcal{F}_\ell|} \sum_{f \in \mathcal{F}_\ell} \|\mathbf{k}_{\ell,f}^s - \mathbf{k}_{\ell,f}^t\|_1, \quad (11)$$

where $\mathcal{F}_\ell$ is the set of kernel families at level ℓ : spatial kernels at all levels, the temporal reuse kernel at the finest level, and upsampling kernels at the coarser levels. This term aligns the student’s filtering behavior with the teacher’s at every pyramid level rather than only at the rendered output.

Implementation details. We use the Noisebase dataset released with NPPD [Balint et al. 2023], which contains 1024 animated training sequences of 64 frames each, rendered in Falcor [Kallweit et al. 2022] from production-style interior and exterior scenes. The data are provided as motion-compensated 256×256 crops with the auxiliary buffers required for neural Monte Carlo denoising (radiance, depth, normals, and motion). Training proceeds in two phases to stabilize the learning of the log-linear expansion. In the *freeze phase* (first $E_f=10$ epochs), the archetype and residual parameters $\mathbf{A}_f$ and δ_f are held fixed at the values produced by Kernel Cluster Generation (Section 4.1), and only the student backbone is trained with all three loss terms. This allows the backbone to learn to produce meaningful mixture weights over a stable archetype subspace before the subspace itself is allowed to update. In the *finetune phase* (remaining epochs), $\mathbf{A}_f$ and δ_f are unfrozen and trained jointly with the backbone at $0.1 \times$ the backbone learning rate. Keeping the components fixed initially lets the backbone learn stable mixture predictions before the representation itself changes. During finetuning, the reduced component learning rate limits collapse and excessive drift from the AA initialization. Once unfrozen, however,

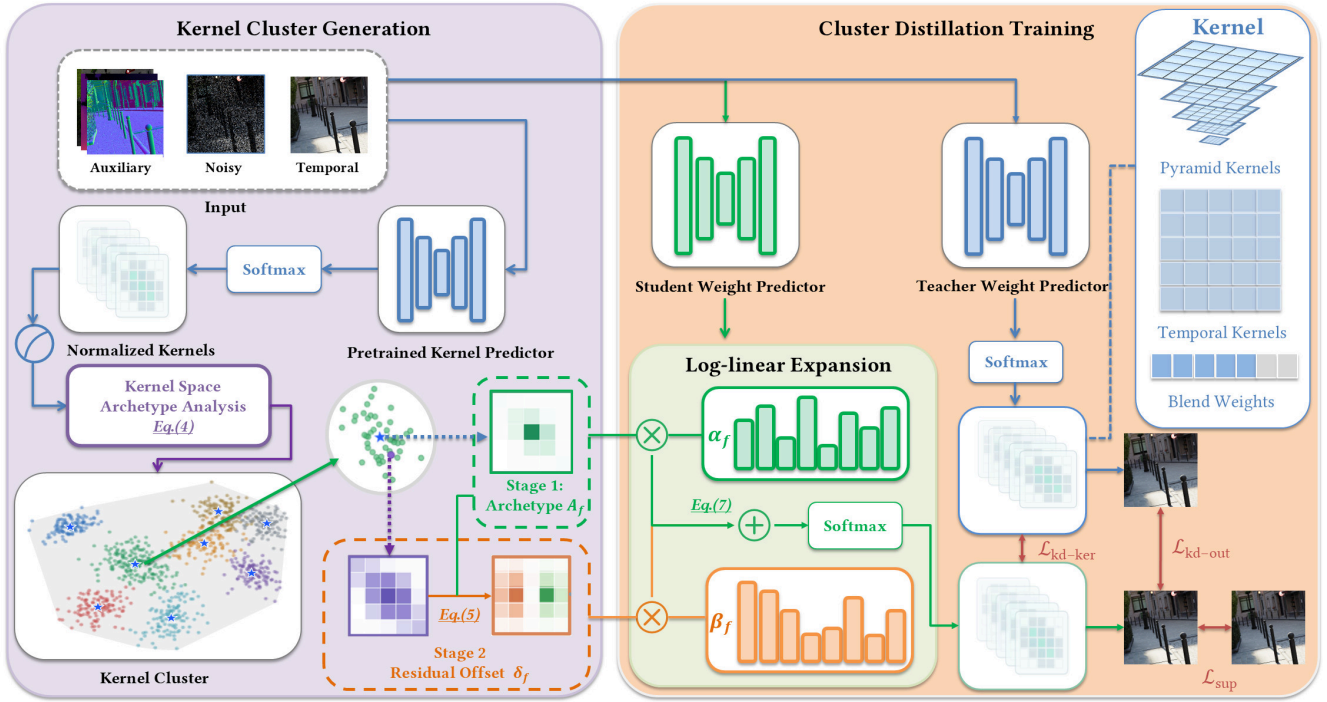


Fig. 4. Overview of our kernel cluster distillation framework. Before training, archetype and residual parameters A_f , δ_f are generated from the teacher’s kernel outputs by a two-stage archetypal analysis (Sec. 4.1). During training, the pretrained NPPD teacher and the lightweight student share the same input; the student predicts archetype weights α_f and residual offset weights β_f at each level, which the log-linear expansion module (Sec. 4.3) maps back to full pyramid kernels. Distillation losses compare the two models on both rendered outputs and expanded kernels.

the components are no longer constrained to their respective AA convex hulls and instead form an AA-initialized learned dictionary. This distinction between the offline AA construction and the trainable deployment representation is detailed in the supplementary material. The kernel distillation weight λ_k is halved after epoch 5, allowing the student to improve its rendered output beyond a strict match to the teacher kernels. The backbone learning rate is 10^{-4} with per-epoch exponential decay ($\gamma=0.94$), and the loss weights are $\lambda_s=1.0$, $\lambda_o=0.5$, $\lambda_k=0.1$. All models are trained in FP16 with gradient clipping at max norm 1.0.

5 Results

We evaluate the proposed *Log-linear Kernel Cluster Distillation* framework together with its large teacher model and compact student baselines (8M parameters) on a test set that combines sequences from the Noisebase benchmark used by NPPD [Balint et al. 2023] with additional scenes rendered using RTXPT [NVIDIA 2023]. The evaluated scenes include *Bistro* [Lumberyard 2017], *Zeroday* [Winkelmann 2019], *Dining Room*, *Bedroom*, and *Kitchen* scenes from the Computer Graphics Archive [Bitterli 2016].

For temporal methods, we allow a 16-frame warm-up phase at the start of each animated sequence to stabilize temporal accumulations. We tonemap all outputs following the ACES guidelines [Hill 2022]

to match professional animation pipelines. Reference frames are rendered at 4096 samples per pixel (spp).

5.1 Comparison of Different Denoising Methods

We compare our method against representative production and learning-based denoisers: NPPD [Balint et al. 2023], Intel Open Image Denoise (OIDN) [Áfra 2024] version 2.4.1, NVIDIA OptiX AI-accelerated denoiser (ONND) [Chaitanya et al. 2017] version 9.1, and Feature-Align Network with Knowledge Distillation for Efficient Denoising (FAN) [Young et al. 2021]. For a fair comparison with direct dense kernel prediction, we train an NPPD baseline with a model size comparable to our student (NPPD Small), while preserving the original dense pyramid-kernel output parameterization. For the FAN, we use the same compact NPPD student and NPPD teacher as ours. We adopt its feature-alignment distillation strategy by matching intermediate teacher features induced by the student output and the ground-truth image.

Quality comparison. Table 1 reports PSNR and SSIM after ACES tonemapping on 1 spp inputs. NPPD (Teacher) achieves the highest reconstruction quality among kernel-prediction baselines, whereas our student maintains comparable quality at substantially lower computational cost. Compared with FAN, the improvement measures the benefit of kernel-level distillation. OIDN and OptiX serve

Method	FLOPs	Param.	<i>Bistro</i>		<i>Zeroday</i>		<i>Dining Room</i>		<i>Bedroom</i>		<i>Kitchen</i>	
			PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
Ours	1.4 T	8.8 M	29.47	0.857	24.66	0.760	30.03	0.944	28.32	0.804	28.77	0.850
NPPD (Small)	1.5 T	8.9 M	28.95	0.828	23.92	0.731	29.24	0.939	28.06	0.801	28.54	0.848
FAN	1.5 T	8.9 M	28.85	0.833	24.28	0.750	29.09	0.941	28.12	0.803	28.46	0.843
ONND	—	—	28.94	0.838	23.20	0.724	28.91	0.946	27.94	0.798	28.77	0.849
OIDN	—	—	27.17	0.793	23.57	0.729	29.18	0.935	27.21	0.791	27.83	0.833
NPPD (Teacher)	2.8 T	15.0 M	29.43	0.857	24.68	0.760	30.04	0.941	28.28	0.804	28.85	0.852

Table 1. Quantitative comparison on 1 spp inputs across five test scenes (*Bistro*, *Zeroday*, *Dining Room*, *Bedroom*, and *Kitchen*). **Best** and **second-best** results per scene and metric are highlighted in color.

	Ours	w/o residual offsets	w/o $\mathcal{L}_{\text{kd-out}}$	w/o $\mathcal{L}_{\text{kd-ker}}$	k -means init.
PSNR ↑	28.25	27.78	28.02	21.23	27.31
SSIM ↑	0.843	0.841	0.842	0.653	0.835

Table 2. Ablation of the compact kernel parameterization, averaged over the five test scenes in Table 1. Higher is better.

as external quality references because they do not expose trainable kernel outputs. Across the five scenes, our student remains close to the teacher in both metrics and consistently improves over the matched NPPD (Small) baseline. These results indicate that the compact output representation preserves most of the teacher’s filtering quality without retaining its dense prediction head.

Figure 5 shows representative visual comparisons across all five test scenes. OIDN and ONND produce over-smoothed outputs that suppress high-frequency details, and occasionally introduce chromatic artifacts on specular surfaces. The equal-capacity NPPD (Small) and FAN recover more details but still fall short on fine textures and sharp edges. Our method preserves sharper boundaries and finer surface details, remaining comparable to the NPPD teacher and demonstrating the effectiveness of distillation. Temporal-stability metrics on animated test sequences are reported in the supplementary material.

Runtime performance. We measure runtime on a single NVIDIA RTX 4090 at 1920×1080 . Networks are deployed in FP16 with TensorRT 10.13 [NVIDIA Corporation 2025], and the pyramid filter uses fused CUDA kernels; timings exclude I/O. Fig. 7 compares the NPPD teacher, a lightweight NPPD of matched capacity using native pyramid-partition kernel prediction, and our log-linear kernel cluster student, at 41.5, 20.8, and 16.3 ms respectively. The 4.5 ms gain over the lightweight NPPD comes from the full-resolution prediction ($6.7 \rightarrow 3.0$ ms) and the CUDA pyramid stage ($3.7 \rightarrow 2.9$ ms), with the remaining network layers unchanged at 10.4 ms. As external references, ONND is GPU-optimized but does not expose component-level timings, and OIDN is not a kernel-prediction denoiser. Their runtime costs are 10.13 and 6.96 ms, respectively.

5.2 Ablation on Kernel Cluster Distillation

We report 1 spp ablations averaged over all five test scenes. All variants share the student backbone, output dimension, and training schedule. Table 2 reports the metrics and Fig. 8 shows visual comparisons; w/o $\mathcal{L}_{\text{kd-ker}}$ is omitted because it fails to converge.

We remove the residual offset branch and replace Eq. (7) with a single mixture over $M_{a,f} + M_{\delta,f}$ shared archetypes. Fig. 6 illustrates the element-wise refinement by the residual offsets. Without them, the student leaves low-frequency noise and block-like patterns in flat regions. Removing $\mathcal{L}_{\text{kd-out}}$ instead keeps the kernel structure close to *Ours*, but residual low-frequency noise remains visible in the rendered images, since kernel-level distillation alone does not penalize image-space mismatches. Without $\mathcal{L}_{\text{kd-ker}}$, the log-linear expansion is never constrained in $\mathcal{K}_f$ and validation PSNR fails to improve, confirming that $\mathcal{L}_{\text{kd-ker}}$ is what anchors $\mathbf{A}_f$ and δ_f during training. We further replace the two-stage archetypal initialization with log-space k -means, using the same freeze-then-finetune schedule. The resulting model produces globally darker reconstructions, suggesting that k -means provides a less faithful initialization for $\mathbf{A}_f$ and δ_f than archetypal analysis. Additional comparisons with NMF and VQ-codebook initializations are provided in the supplementary material.

6 Conclusion and Future Work

In this work, we analyze the performance bottleneck of kernel prediction and show that the dominant cost comes from finest convolution stage with large output channels. Motivated by the observation that teacher kernels concentrate within a compact region of the kernel space $\mathcal{K}_f$, we propose *Log-linear Kernel Cluster Distillation*, which trains a student to predict low-dimensional archetype and residual offset weights and reconstructs the full kernels through a

log-linear expansion. Our results show that the student model reduces end-to-end inference time by 60.7% relative to the 15M NPPD teacher while maintaining comparable denoising quality.

Although the proposed formulation is designed to be broadly applicable, we have not conducted wider generalization studies across more diverse denoising pipelines, nor compared against more aggressive compression settings. Since the archetypes and residual offsets are precomputed and shared across all kernels, they are amenable to further compression via post-training quantization [Jacob et al. 2018] or low-precision numerical formats [Micikevicius et al. 2022]. We also expect that more compact representations of the kernel space beyond a log-linear archetype mixture can be discovered through dictionary learning [Mairal et al. 2009] or implicit neural parameterizations [Sitzmann et al. 2020].

Acknowledgments

The authors would like to thank the reviewers for their constructive comments and helpful suggestions. This work was supported by the National Natural Science Foundation of China (No. 62672229); the Nanjing Major Science and Technology Special Project (No. 202209003); the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (No. JYB2025XDXM118); and the 111 Center (No. B26023).

References

- Attila T. Áfra. 2024. Intel® Open Image Denoise. <https://www.openimagedenoise.org>.
- Aleix Alcacer, Irene Epifanio, Sebastian Mair, and Morten Mørup. 2025. A Survey on Archetypal Analysis. arXiv:2504.12392 [stat.ME] <https://arxiv.org/abs/2504.12392>
- 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. In *ACM Transactions on Graphics (SIGGRAPH)*, Vol. 36. Association for Computing Machinery, New York, NY, USA, 97:1–97:14. doi:10.1145/3072959.3073708
- Martin Balint, Krzysztof Wolski, Karol Myszkowski, Hans-Peter Seidel, and Radosław Mantiuk. 2023. Neural Partitioning Pyramids for Denoising Monte Carlo Renderings. In *ACM SIGGRAPH 2023 Conference Proceedings*. ACM, New York, NY, USA, Article 60, 11 pages. doi:10.1145/3588432.3591562
- Benedikt Bitterli. 2016. Rendering resources. <https://benedikt-bitterli.me/resources/>.
- Zhilin Cai, Yang Zhang, Marco Manzi, Cengiz Oztireli, Markus Gross, and Tunç Ozan Aydin. 2021. Robust Image Denoising using Kernel Predicting Networks. In *Eurographics 2021 - Short Papers*, Holger Theisel and Michael Wimmer (Eds.). The Eurographics Association. doi:10.2312/egs.20211018
- Guillermo Carbajal, Patricia Vitoria, Mauricio Delbracio, and Lezama J Mus' eP. 2021. Single image non-uniform blur kernel estimation via adaptive basis decomposition. *arXiv preprint arXiv:2102.01026* (2021).
- 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. doi:10.1145/3072959.3073601
- Hajin Choi, Seokpyo Hong, Inwoo Ha, Nahyup Kang, and Bochang Moon. 2024. Online Neural Denoising with Cross-Regression for Interactive Rendering. *ACM Trans. Graph.* 43, 6, Article 221 (Nov. 2024), 12 pages. doi:10.1145/3687938
- Per H. Christensen and Wojciech Jarosz. 2016. The Path to Path-Traced Movies. *Foundations and Trends in Computer Graphics and Vision* 10, 2 (Oct. 2016), 103–175. doi:10/gfjwjc
- Adele Cutler and Leo Breiman. 1994. Archetypal Analysis. *Technometrics* 36, 4 (1994), 338–347. doi:10.1080/00401706.1994.10485840
- Hangming Fan, Rui Wang, Yuchi Huo, and Hujun Bao. 2021. Real-time Monte Carlo Denoising with Weight Sharing Kernel Prediction Network. *Computer Graphics Forum* 40, 4 (2021), 15–27. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14338 doi:10.1111/cgf.14338
- Michaël Gharbi, Tzu-Mao Li, Miika Aittala, Jaakko Lehtinen, and Frédo Durand. 2019. Sample-based Monte Carlo denoising using a kernel-splatting network. *ACM Trans. Graph.* 38, 4, Article 125 (July 2019), 12 pages. doi:10.1145/3306346.3322954
- Stephen Hill. 2022. ACES Tone Mapping Operator. <https://github.com/TheRealMJP/BakingLab/blob/master/BakingLab/ACES.hlsl>. Accessed: May 13, 2026.
- Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. arXiv:1503.02531 [stat.ML]
- Junjie Hu, Chenyou Fan, Hualie Jiang, Xiyue Guo, Yuan Gao, Xiangyong Lu, and Tin Lun Lam. 2023. Boosting LightWeight Depth Estimation via Knowledge Distillation. In *Knowledge Science, Engineering and Management*. Springer Nature Switzerland, Cham, 27–39. doi:10.1007/978-3-031-40283-8_3
- Mustafa Isik, Krishna Mullia, Matthew Fisher, Jonathan Eisenmann, and Michaël Gharbi. 2021. Interactive Monte Carlo denoising using affinity of neural features. *ACM Trans. Graph.* 40, 4, Article 37 (jul 2021), 13 pages. doi:10.1145/3450626.3459793
- Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2704–2713.
- Simon Kallweit, Petrik Clarberg, Craig Kolb, Tomáš 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>
- Paweł Kaźmierczyk, SungYe Kim, Wojciech Uss, Wojciech Kaliński, and Tomasz Galaj. 2025. Joint Denoising and Upscaling via Multi-branch and Multi-scale Feature Network. In *Proceedings of the 2025 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '25)*. Association for Computing Machinery, New York, NY, USA, Article 11, 12 pages. doi:10.1145/3728297
- Mengxun Kong, Jie Guo, Chen Wang, Ye Yuan, and Yanwen Guo. 2025. Real-Time Neural Denoising with Render-Aware Knowledge Distillation. *Proceedings of the AAAI Conference on Artificial Intelligence* 39, 4 (2025), 4392–4400. doi:10.1609/aaai.v39i4.32462
- Tzu-Mao Li, Yu-Ting Wu, and Yung-Yu Chuang. 2012. SURE-based optimization for adaptive sampling and reconstruction. *ACM Trans. Graph.* 31, 6, Article 194 (Nov. 2012), 9 pages. doi:10.1145/2366145.2366213
- Yifan Liu, Changyong Shu, Jingdong Wang, and Chunhua Shen. 2023. Structured Knowledge Distillation for Dense Prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 6 (2023), 7035–7049. doi:10.1109/TPAMI.2020.3001940
- Amazon Lumberyard. 2017. Amazon Lumberyard Bistro, Open Research Content Archive (ORCA). <http://developer.nvidia.com/orca/amazon-lumberyard-bistro>
- Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. 2009. Online Dictionary Learning for Sparse Coding. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*. 689–696.
- Xiaoxu Meng, Quan Zheng, Amitabh Varshney, Gurprit Singh, and Matthias Zwicker. 2020. Real-time Monte Carlo Denoising with the Neural Bilateral Grid. In *Eurographics Symposium on Rendering - DL-only Track*, Carsten Dachsbacher and Matt Pharr (Eds.). The Eurographics Association, London, UK, 45–54. doi:10.2312/sr.20201133
- Paulius Micikevicius, Dusan Stolic, Neil Burgess, Marius Cornea, Pradeep Dube, Richard Grisenthwaite, Sangwon Ha, Alexander Heinecke, Patrick Judd, John Kamalu, Naveen Mellempudi, Stuart Oberman, Mohammad Shoeybi, Michael Siu, and Hao Wu. 2022. FP8 Formats for Deep Learning. *arXiv preprint arXiv:2209.05433* (2022).
- Jacob Munkberg and Jon Hasselgren. 2020. Neural Denoising with Layer Embeddings. *Computer Graphics Forum* 39, 4 (2020), 1–12. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14049 doi:10.1111/cgf.14049
- Simon Niklaus, Long Mai, and Feng Liu. 2017. Video frame interpolation via adaptive separable convolution. In *Proceedings of the IEEE international conference on computer vision*. 261–270.
- NVIDIA. 2022. NVIDIA Real-Time Denoisers. <https://developer.nvidia.com/rtx/ray-tracing/rt-denoisers>. Accessed: 2026-05-08.
- NVIDIA. 2023. NVIDIA® RTX Path Tracing. <https://github.com/NVIDIA-RTX/RTXPT>
- NVIDIA Corporation. 2025. TensorRT. <https://developer.nvidia.com/tensorrt>. Version 10.13.
- Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2023. *Physically Based Rendering: From Theory to Implementation (4th ed.)* (4th ed.). The MIT Press. <https://pbr-book.org/>
- Christoph Schied, Anton S. Kaplanyan, Chris Wyman, Anjul Patney, Marco Salvi, Chakravarty R. Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten

- Dachsbacher, and Aaron Lefohn. 2017. Spatiotemporal Variance-Guided Filtering: Real-Time Reconstruction for Path-Traced Global Illumination. In *Proceedings of High Performance Graphics (HPG)*. Association for Computing Machinery, New York, NY, USA, 1–12. doi:10.1145/3105762.3105770
- Pradeep Sen and Soheil Darabi. 2012. On filtering the noise from the random parameters in Monte Carlo rendering. *ACM Trans. Graph.* 31, 3, Article 18 (May 2012), 15 pages. doi:10.1145/2167076.2167083
- Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. 2020. Implicit Neural Representations with Periodic Activation Functions. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 33. 7462–7473.
- Manu Mathew Thomas, Gabor Liktó, Christoph Peters, Sungye Kim, Karthik Vaidyanathan, and Angus G. Forbes. 2022. Temporally Stable Real-Time Joint Neural Denoising and Supersampling. *Proc. ACM Comput. Graph. Interact. Tech.* 5, 3 (July 2022), Article 21. doi:10.1145/3543870
- Thijs Vogels, Fabrice Rousselle, Brian Mcwilliams, Gerhard R  thlin, Alex Harvill, David Adler, Mark Meyer, and Jan Nov  k. 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
- Mike Winkelmann. 2019. Zero-Day, Open Research Content Archive (ORCA). <https://developer.nvidia.com/orca/beeple-zero-day>. <https://developer.nvidia.com/orca/beeple-zero-day>
- Zhihao Xia, Federico Perazzi, Micha  l Gharbi, Kalyan Sunkavalli, and Ayan Chakrabarti. 2020. Basis Prediction Networks for Effective Burst Denoising with Large Kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11844–11853.
- Run Yan, Hui Guo, Libo Huang, Nong Xiao, Shen Li, Yongwen Wang, Yashuai Lv, and Gang Chen. 2025. A Survey on Deep Learning for Monte Carlo Path Tracing. *ACM Comput. Surv.* 58, 4, Article 101 (Oct. 2025), 38 pages. doi:10.1145/3768618
- Lucas D. Young, Fitsum A. Reda, Rakesh Ranjan, Jon Morton, Jun Hu, Yazhu Ling, Xiaoyu Xiang, David Liu, and Vikas Chandra. 2021. Feature-Align Network with Knowledge Distillation for Efficient Denoising. arXiv:2103.01524 [eess.IV] <https://arxiv.org/abs/2103.01524>
- Beiji Zou, Yue Zhang, Min Wang, and Shu Liu. 2022. Toward Efficient Image Denoising: A Lightweight Network with Retargeting Supervision Driven Knowledge Distillation. In *Computer Graphics International Conference*. Springer, 15–27.
- Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, Bochang Moon, Ravi Ramamoorthi, Fabrice Rousselle, Pradeep Sen, Cyril Soler, and Sung-Eui Yoon. 2015. Recent Advances in Adaptive Sampling and Reconstruction for Monte Carlo Rendering. *Computer Graphics Forum* 34, 2 (2015), 667–681. doi:10.1111/cgf.12592



Fig. 5. Our method is compared against OIDN, ONND, FAN, and NPPD across five rendered scenes (*Bistro*, *Zeroday*, *Dining Room*, *Bedroom*, and *Kitchen*). Our method consistently outperforms the competing methods in reconstructing fine shadow details, metallic surface highlights, delicate texture patterns, and sharp object boundaries, achieving a better overall balance between noise removal and detail preservation. *Zeroday* © Mike Winkelmann, *Bistro* © Amazon Lumberyard, *Dining Room* © Wig42, *Bedroom* © SlykDrako, *Kitchen* © Jay-Artist.

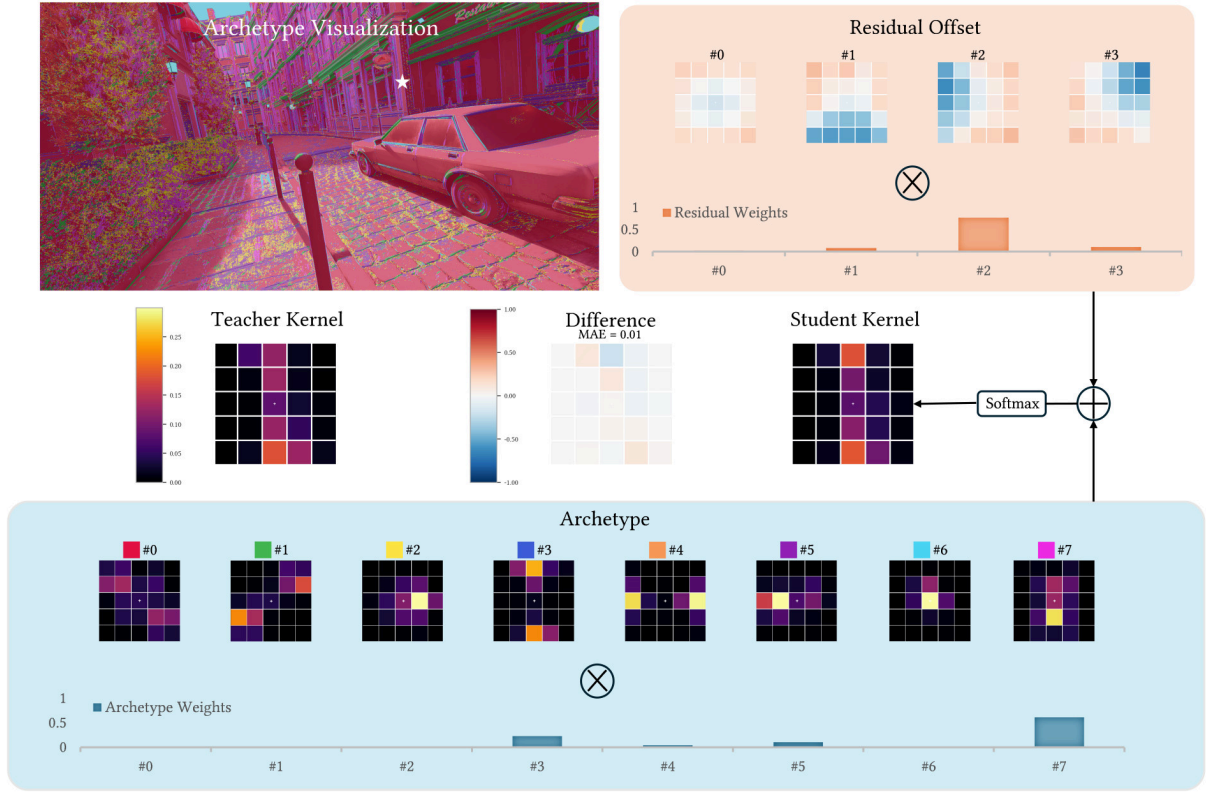


Fig. 6. Log-linear expansion for a single pixel (marked $\star$) on the *Bistro* with $M_{a,f}=8$, $M_{s,f}=4$. The archetype visualization shows each pixel colored by its dominant archetype cluster, revealing spatially coherent regions that share similar filtering behavior. The student predicts archetype weights α and residual weights β for each pixel; their log-space components are added and normalized by softmax to yield the student kernel (MAE = 0.01 vs. teacher kernel). Archetypes alone provide only a coarse approximation of the teacher kernel but the residual offsets supply the remaining fine-grained corrections necessary to close the gap. *Bistro* © Amazon Lumberyard.

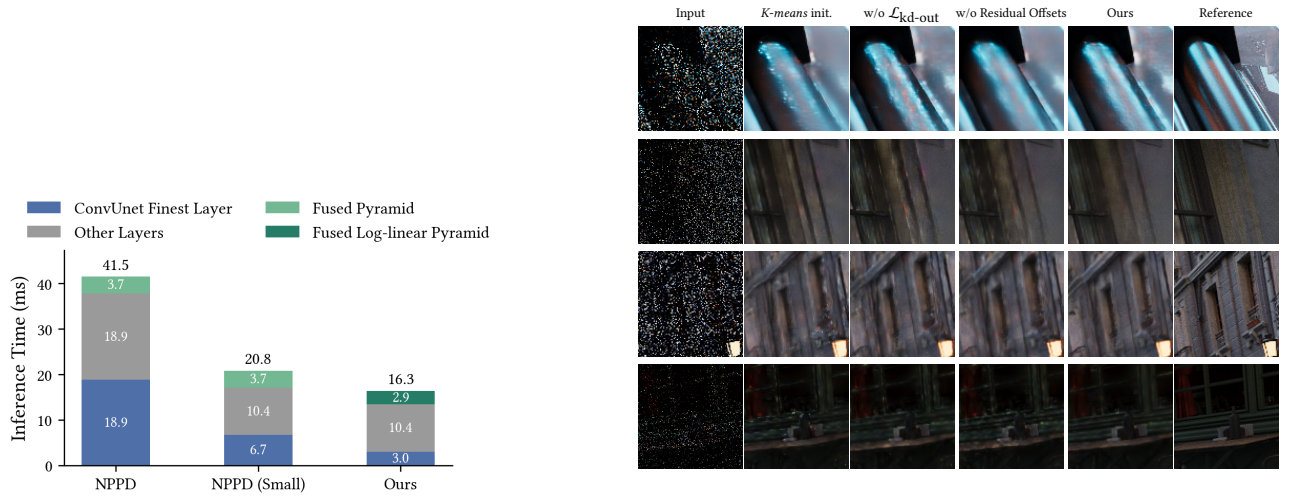


Fig. 7. Runtime decomposition of the teacher baseline, a lightweight NPPD, and our kernel cluster student at 1920×1080 on an NVIDIA RTX 4090. Segment labels and totals are in milliseconds.

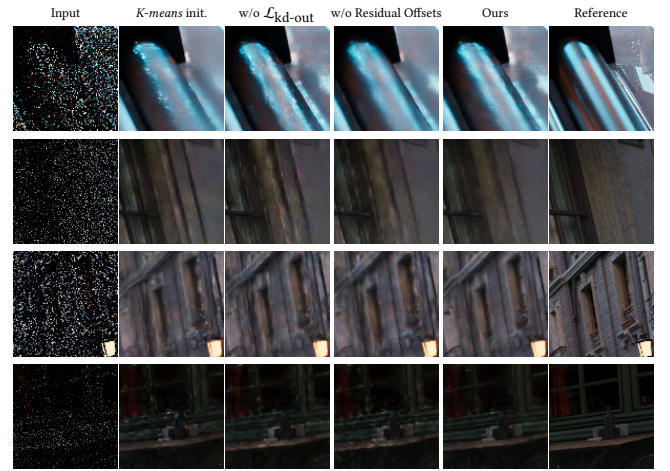


Fig. 8. Ablation studies on kernel distillation components. *K-means* initialization instead of archetypal analysis, removal of the output distillation loss $\mathcal{L}_{kd-out}$, and removal of residual offsets. All other training settings remain identical.