CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Hierarchical Architecture Optimization for Efficient Transformer-based Monte Carlo Denoising

Master's thesis in Computer science and engineering

Xinglu Gong

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2026

Master's thesis 2026

Hierarchical Architecture Optimization for Efficient Transformer-based Monte Carlo Denoising

Xinglu Gong

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2026

Hierarchical Architecture Optimization for Efficient Transformer-based Monte Carlo Denoising
Xinglu Gong

Supervisor: Erik Sintorn, Department of Computer Science and Engineering
Examiner: Ulf Assarsson, Department of Computer Science and Engineering

Master's Thesis 2026
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2026

Hierarchical Architecture Optimization for Efficient Transformer-based Monte Carlo Denoising
Xinglu Gong
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Physically-based Monte Carlo rendering heavily relies on deep learning denoisers to reconstruct photorealistic images from low Sample-Per-Pixel (SPP) inputs. Recently, the Joint Self-Attention (JSA) framework, a Transformer-base method utilizing auxiliary G-buffers, has achieved state-of-the-art visual fidelity. However, the quadratic computational complexity of standard multi-head self-attention disqualifies it from interactive or real-time rendering applications. To bridge this efficiency gap, this thesis proposes a highly efficient denoising architecture by systematically eliminating the computational redundancy of standard JSA frameworks. The proposed optimization strategy is executed on two architectural levels. At the micro-level, we adapt the Single-Head Joint Self-Attention (SH-JSA) module with a partial channel ratio to preserve high-frequency structural features while reducing computational cost. Furthermore, guided by hardware profiling, we replace early-stage attention blocks with optimized Convolutional Neural Networks (CNNs). At the macro-level, we progressively streamline the global U-Net structure by implementing a symmetric decoder, reducing the network depth to three stages, and expanding the input patch size. Extensive evaluations demonstrate a leap in efficiency. For high-resolution 1024×1024 inputs, the proposed framework reduces the network parameter count by 86.6% and slashes the inference latency by 91.6%. While this extreme acceleration introduces a minor 4.9% drop in Peak Signal-to-Noise Ratio (PSNR), it successfully transitions the JSA-based denoiser from offline computation to interactive frame rates.

Keywords: Computer graphics, Monte Carlo rendering, real-time rendering, Monte Carlo denoising, transformer-based denoising.

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor, Erik Sintorn, for his invaluable guidance and feedback during this master thesis. His continuous encouragement during technical difficulties made a huge difference in completing this work.
I am also very grateful to the Department of Computer Science and Engineering for providing the essential computational resources. Having access to these facilities made it possible to run the extensive experiments and achieve the final results of this research.
Finally, my deepest appreciation goes to my family and friends for their endless support. I am especially grateful to my parents for their unconditional love and for always believing in me, which gave me the strength to get through the hardest parts of this journey. Furthermore, I would like to thank my friends for their company and willingness to listen. Their presence made this research experience much more enjoyable and memorable.

Xinglu Gong, Gothenburg, May 2026

Contents

List of Figures

List of Tables

1
Introduction

This chapter establishes the foundation and motivation for the thesis. It begins by introducing the background of Monte Carlo rendering and the crucial role of deep learning in image-space denoising. Subsequently, it identifies the severe computational bottlenecks inherent in current Joint Self-Attention (JSA) framework. Based on these limitations, the chapter formulates the core research question that drives this study and summarizes the key contributions of the thesis.

1.1 Background

Physically-based Monte Carlo rendering is the industry standard for generating photorealistic images in computer graphics. However, to produce a noise-free image, this algorithm requires tracing thousands of light paths, which is an extremely time-consuming process. To bridge the gap between rendering speed and visual fidelity, modern graphics pipelines typically render images at a very low sampling rate and apply a post-processing denoiser to reconstruct the final image.
In recent years, deep learning has revolutionized Monte Carlo denoising. Specifically, the Joint Self-Attention (JSA) mechanism [1] has achieved state-of-the-art reconstruction quality. By leveraging auxiliary scene features, such as albedo and normal map, within the self-attention calculation, the JSA framework can accurately capture long-range global illumination dependencies and reconstruct complex optical phenomena.

1.2 Problem Statement

Despite its superior visual performance, standard JSA denoising framework suffer from severe computational bottlenecks. The core issue lies in the standard multi-head self-attention mechanism, which exhibits a quadratic computational complexity, $O(N^2)$, with respect to the spatial resolution. When processing high-resolution images, this quadratic scaling leads to an explosion in theoretical Floating-Point Operations (FLOPs). As a result, standard JSA denoisers typically require over a second to process a single frame. This inference latency confines them to offline rendering pipelines and completely disqualifies them from interactive or real-time rendering applications, where the denoising budget is strictly limited to fractions of a second.
To address the critical computational inefficiency of standard Transformer-based denoisers, this thesis is fundamentally guided by the following core research ques-

tion: Can we significantly reduce the latency and FLOPs of the JSA denoiser while maintaining comparable image reconstruction quality?

1.3 Contribution

To answer this research question, this thesis systematically reduces the computational redundancy of the standard JSA framework through multi-level optimizations and proposes a new lightweight denoising framework. Compared to the heavy JSA baseline, the proposed framework reduces the parameter count by 86.6% and slashes inference latency by 91.6%. This enables interactive rendering speeds ($\approx$ 10.4 FPS) with only a minor 4.9% drop in image quality, validating a highly efficient blueprint for future real-time rendering pipelines.

2

Theory

2.1 Monte Carlo Rendering and Denoising

This chapter introduces the theoretical foundation of physically-based rendering. It begins with the Rendering Equation and points out that it is impossible to solve it analytically in complex scenes. Next, Monte Carlo Integration is introduced as a numerical method to approximate this equation. Finally, the necessity of image-space denoising for real-time application is analyzed.

2.1.1 The Rendering Equation

The theoretical foundation of physically-based computer graphics relies on the Rendering Equation, introduced by James Kajiya in 1986 [2]. It describes the equilibrium distribution of light energy within a closed environment at a macroscopic level. The equation is formulated as an integral over a hemisphere and is mathematically expressed as follows:

$$L_o(p, \omega_o) = L_e(p, \omega_o) + \int_\Omega f_r(p, \omega_i, \omega_o) L_i(p, \omega_i)(\omega_i \cdot n) d\omega_i$$

where each term represents a distinct physical quantity:

- $L_o(p, \omega_o)$ denotes the total outgoing radiance from a surface point p along the direction ω_o.
- $L_e(p, \omega_o)$ represents the emitted radiance directly from point p along ω_o, which evaluates to zero for non-emissive surfaces.
- Ω signifies the unit hemisphere centered at point p and aligned with the surface normal vector n.
- $f_r(p, \omega_i, \omega_o)$ is the Bidirectional Reflection Distribution Function (BRDF), which defines how much radiance incoming from direction ω_i is reflected towards the outgoing direction ω_o based on material properties.
- $L_i(p, \omega_i)$ is the incoming radiance arriving at point p from direction ω_i.
- $\omega_i \cdot n$ represents the cosine factor, accounting for the geometric attenuation of irradiance due to the incident angle of light relative to the surface normal.

Due to the fact that the incoming radiance L_i at point p is recursively determined by the outgoing radiance L_o from other visible surfaces in the scene, finding an analytical solution to this equation is impossible in complex environments with intricate geometric topologies and light configurations. Consequently, numerical approximation methods must be employed.

2.1.2 Monte Carlo Integration

Monte Carlo integration is a powerful statistical technique used to numerically estimate the value of complex or high-dimensional integrals that are hard to be solved analytically. Suppose we intend to evaluate a definite integral I of a function $f(x)$ over a multi-dimensional domain V:

$$I = \int_V f(x)dx$$

The core principle of Monte Carlo integration is to draw N independent and identically distributed random samples $\{x_1, x_2, \ldots, x_N\}$ from the domain V according to a known probability density function (PDF), denoted as $p(x)$. An unbiased estimator $\hat{I}$ for the true integral value I is then constructed by computing the weighted average of these sample evaluations:

$$\hat{I} = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}$$

According to the Law of Large Numbers, as the number of samples N approaches infinity, the estimated value $\hat{I}$ converges to the true integral value I with a probability of 1.

2.1.3 Monte Carlo Rendering

This section details how the principles of Monte Carlo integration are applied to solve the Rendering Equation. The reflection term in the rendering equation is inherently a continuous integral defined over a hemispherical domain Ω, which aggregates light energy from all incident directions. To evaluate this integral numerically, a probability density function (PDF), denoted as $p(\omega_i)$, must be defined over the hemisphere Ω. This PDF must satisfy the normalization condition $\int_\Omega p(\omega_i)d\omega_i = 1$. Based on this probability distribution, N incident light directions $\{\omega_i^1, \omega_i^2, \ldots, \omega_i^N\}$ are randomly sampled over the hemispherical surface. By replacing the continuous integral with a discrete sum, we formulate the unbiased Monte Carlo estimator for the outgoing radiance at a surface point:

$$\hat{L}_o(p, \omega_o) = L_e(p, \omega_o) + \frac{1}{N}\sum_{k=1}^{N}\frac{f_r(p, \omega_i^k, \omega_o)L_i(p, \omega_i^k)(\omega_i^k \cdot n)}{p(\omega_i^k)}$$

In practical rendering algorithms such as Path Tracing, this mathematical estimator is evaluated on a per-pixel basis to synthesize the final 2D image. Specifically, to determine the accurate color and brightness of a single pixel, the virtual camera casts N independent viewing rays through that specific pixel on the image plane into the 3D scene. Each viewing ray initiates a random light transport path. It simulates the recursive process of ray intersection where the incoming radiance at the current point p equals the outgoing radiance from the intersected surface p', i.e., $L_i(p, \omega_i^k) = L_o(p', -\omega_i^k)$. Crucially, each of these independent paths serves as a single sample for the Monte Carlo integration. The total number of samples N in the estimator formula directly corresponds to the fundamental rendering metric

known as Samples Per Pixel (SPP). The final radiance value of a pixel is simply computed as the arithmetic mean of the radiance contributed by these N distinct paths. Through this pixel-wise stochastic sampling, Monte Carlo rendering translates the abstract hemispherical integral into a sequence of computable light transport paths, establishing itself as a well-developed standard for generating photorealistic images with complex phenomena like global illumination and soft shadows.

2.1.4 Monte Carlo Denoising

Although the Monte Carlo rendering estimator is mathematically unbiased, it intrinsically suffers from severe high-frequency noise when operating at low sampling rates. The domain of all possible light paths in a complex scene is extremely large; consequently, sampling only a limited number of paths introduces substantial statistical variance. In the final image space, this variance manifests visually as noticeable noise and bright pixels commonly referred to as "fireflies." According to the Central Limit Theorem, the error in Monte Carlo integration diminishes at a rate of $O(1/\sqrt{N})$. This implies that to halve the perceived noise level in an image, the number of samples per pixel must be quadrupled. Generating a completely smooth, noise-free ground-truth image often requires thousands of samples per pixel.
For real-time applications or hardware with limited computational budgets, achieving such high sample counts within a single frame is completely unfeasible. To resolve this, image-space denoising has emerged as one of mainstream post-processing solutions to drastically reduce the required sample counts. By rendering the scene at an extremely low sample rate (e.g., 1 to 4 SPP) and passing the noisy output through an advanced denoising algorithm, it is possible to mathematically filter out the high-frequency variance while preserving geometric edges and texture details. This combination of sparse sampling and robust post-processing allows high-quality Monte Carlo rendering to become practical for time-constrained environments.

2.2 Related Work for Learning-based Monte Carlo Denoising

As introduced in previous sections, the visual quality of Monte Carlo rendering severely degrades at low sampling rates. Early deep learning approaches successfully addressed this by replacing traditional heuristic filters with Convolutional Neural Networks. A pivotal work in this era is the Kernel-Predicting Convolutional Network proposed by Bako et al. [3]. Instead of directly predicting pixel colors, it estimates local weighting kernels to compute denoised pixels from their neighbors, offering robust and stable training. Concurrently, Chaitanya et al. [4] introduced a recurrent denoising autoencoder to process image sequences interactively, marking an early success in temporally stable MC reconstruction through deep autoencoder structures and skip connections.
As the demand for real-time rendering grew, researchers focused on optimizing network architectures to reduce inference latency. Meng et al. [5] utilized a differentiable neural bilateral grid, enabling end-to-end training and achieving real-time

denoising for 1 sample-per-pixel inputs in just a few milliseconds per frame. Similarly, Fan et al. [6] proposed a weight-sharing kernel prediction network that generates compact importance maps to decode complete filtering weights. This approach significantly improved the capacity of fully convolutional networks without adding extra time costs during network inference.

To further preserve high-frequency details and handle complex lighting scenarios, researchers began exploring advanced filtering mechanisms beyond standard pixel-space operations. Işık et al. [7] introduced a method that computes pairwise affinity over neural features in a pixel neighborhood to assemble dilated spatial kernels. To tackle the persistent issue of low-frequency artifacts such as blotchiness and ringing, Balint et al. [8] proposed neural partitioning pyramids. Their lightweight U-Net architecture partitions the input radiance and predicts kernels across different scales, effectively balancing image detail preservation and noise reduction.

Despite the success of CNN-based architectures, their localized receptive fields inherently limit their ability to capture long-range dependencies and global illumination contexts. To overcome this, attention mechanisms were introduced to the MC denoising pipeline. Yu et al. [9] pioneered the use of self-attention in this domain with the Auxiliary Feature Guided Self-Attention network, which conducts non-local filtering in the embedding space.

2.3 Vision Transformer Architecture

The Vision Transformer (ViT), originally introduced by Dosovitskiy et al. [10], marked a major paradigm shift in computer vision by adapting the powerful Transformer architecture from natural language processing to image-domain tasks. Unlike Convolutional Neural Networks (CNNs) that rely on localized receptive fields and inductive biases, ViTs leverage self-attention mechanisms to model global dependencies and long-range interactions across an entire image from the very first layer. Although ViT was initially designed for primary image-related tasks such as image classification, it also has the potential to handle image restoration and denoising tasks, where understanding global lighting contexts and structural relationships across different image regions is vital. A standard structure of ViT for classification is shown in Fig. 2.1.

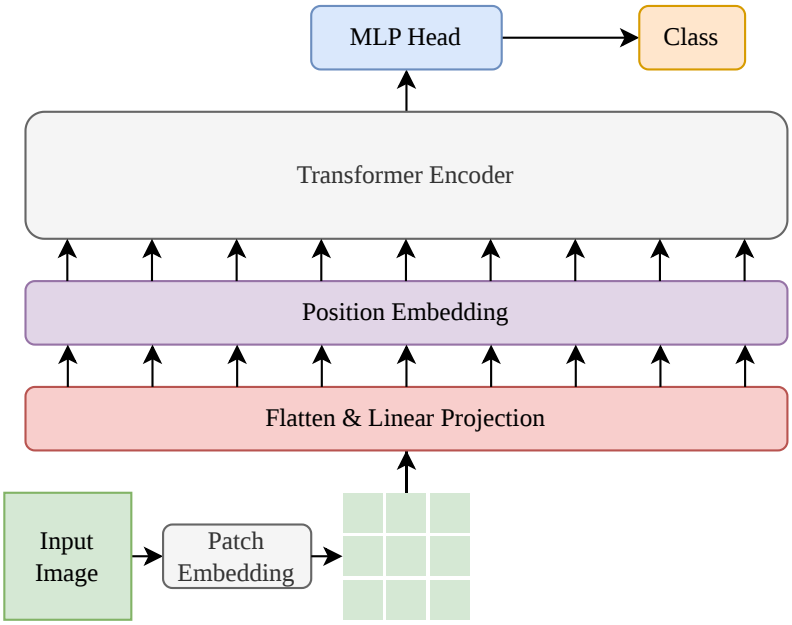


Figure 2.1: Vision Transformer Architecture for Classification Task

2.3.1 Patch Embedding

Standard Transformer architectures are fundamentally designed to process one-dimensional token sequences rather than two-dimensional image grids. To bridge this structural discrepancy, ViT utilizes a patch embedding process. In the context of computer vision tasks, the implication of a patch is equivalent to that of a token in the context of natural language processing tasks. Given an input image $I \in \mathbb{R}^{H \times W \times C_{in}}$, it is partitioned into a sequence of non-overlapping 2D patches $I_p \in \mathbb{R}^{N \times P \times P \times C_{in}}$, where (P, P) is the spatial resolution of each patch and $N = (HW)/P^2$ represents the total number of resulting patches. The Vision Transformer adopts a patch size of 16 (16×16 pixels) as its standard configuration. The patch size is usually determined empirically through experiments, as it represents an architectural trade-off between representation accuracy and computational efficiency. These patches are then flattened and projected into a linear embedding space of dimension C through a trainable linear projection layer. Learnable position embeddings are finally added to the patch embeddings to retain the spatial geometric layout of the original image before entering the Transformer layers.

2.3.2 Encoder-only Structure

The standard Vision Transformer follows an encoder-only design, omitting the auto-regressive decoder architecture prevalent in machine translation models. In this setup, the complete sequence of patches is processed symmetrically through multiple stacked Transformer blocks, allowing every patch to interact globally with every other patch via self-attention.

2.3.3 Transformer Block

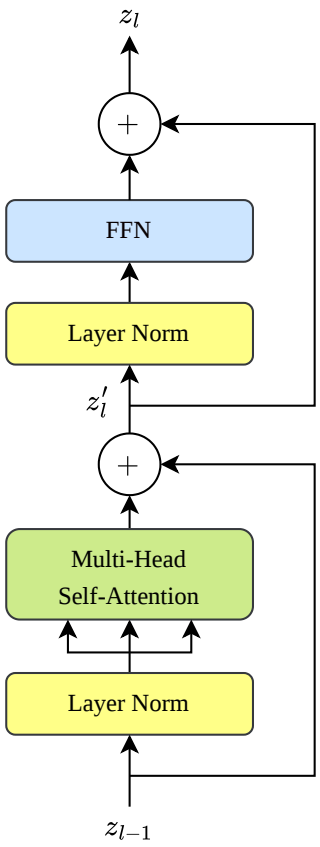


Figure 2.2: Transformer block

The core of the Vision Transformer is the Transformer block [11], which is sequentially stacked to form the network hierarchy. A standard Transformer block consists of two primary sub-layers: a Multi-Head Self-Attention (MHSA) layer and a Position-wise Feed-Forward Network (FFN) layer, as shown in Fig. 2.2. To facilitate stable optimization and prevent vanishing gradients in deep architectures, each sub-layer is preceded by Layer Normalization (LN) and augmented with a residual connection. The overall data flow within a single Transformer block can be mathematically expressed as:

$$z'_l = \text{MHSA}(\text{LN}(z_{l-1})) + z_{l-1}$$
$$z_l = \text{FFN}(\text{LN}(z'_l)) + z'_l$$

where z_{l-1} and z_l denote the input and output tokens of the l-th block, respectively.

2.3.3.1 Multi-Head Self-Attention

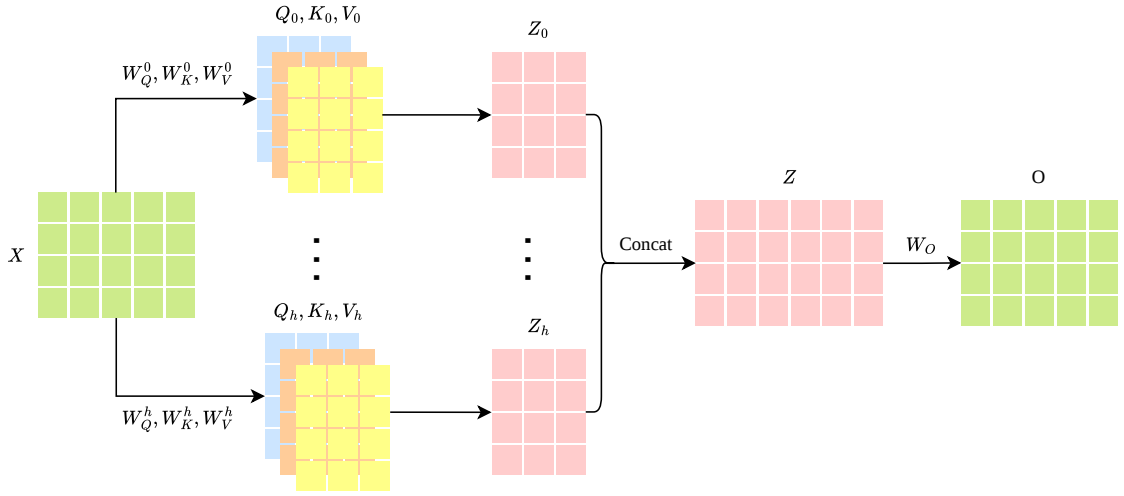


Figure 2.3: Multi-Head Self-Attention Mechanism

The Multi-Head Self-Attention (MHSA) mechanism is the key component of the Transformer block, as shown in Fig. 2.3. It allows the network to attend to information from different representation subspaces at different positions. For a single attention head, the input feature matrix $X \in \mathbb{R}^{L \times C}$ (where L is the token count, and C is the channel dimension) is linearly projected into three distinct spaces: Queries (Q), Keys (K), and Values (V) via learnable weight matrices $W_Q, W_K, W_V \in \mathbb{R}^{C \times d}$:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V$$

The core mathematical formula of the attention layer is expressed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$

This formula functions through three fundamental operations. First, the matrix multiplication QK^T computes the pairwise dot products between queries and keys, serving as a geometric measure to evaluate the semantic similarity between different tokens in the sequence. Second, the scaling factor $1/\sqrt{d}$, where d is the manually defined dimension of keys and queries, is applied to neutralize the variance inflation inherent in high-dimensional dot products. It helps to prevent vanishing gradients during training. Finally, the normalized attention map is multiplied by the value matrix V. Through this step, tokens with high similarity weights contribute more heavily to the final representation of the target token, while irrelevant tokens are effectively filtered out.
In a MHSA module, rather than performing this operation once with a single large attention head, the process is executed in parallel across h independent attention heads, each possessing its own unique learnable projection matrices. This allows the network to jointly attend to information from different representation subspaces

simultaneously:

$$Z_i = \text{Attention}(XW_Q^i, XW_K^i, XW_V^i)$$
$$O = \text{Concat}(Z_1, Z_2, \ldots, Z_h)W_O$$

where $W_O \in \mathbb{R}^{h \cdot d \times C}$ is a linear projection layer that fuses the concatenated outputs of all heads back into the original feature space.

2.3.3.2 Feed Forward Network

Following the attention layer, the token representations are processed by a Feed-Forward Network (FFN), which operates on each token independently. The FFN is responsible for non-linear feature transformation and channel-wise feature interaction. It typically comprises two linear layers with a non-linear activation function, such as the Rectified Linear Unit (ReLU):

$$\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2$$

Typically, the first linear layer expands the channel dimension by an expansion factor (usually 4), and the second layer projects it back to the original dimension. The FFN provides the necessary model capacity to encode intricate pattern configurations once spatial relationships have been aggregated by the attention mechanism.

2.4 Single-Head Vision Transformer Architecture

To bridge the gap between high-performance vision models and efficient deployment on resource-constrained platforms, the Single-Head Vision Transformer (SHViT) architecture [12] was introduced as a novel alternative to traditional ViTs. Standard ViTs, despite their exceptional accuracy, suffer from severe computational and memory redundancies across multiple design levels. The SHViT framework directly mitigates these bottlenecks through macro-level and micro-level structural optimizations, eliminating redundant attention operations without sacrificing the network's performance.

2.4.1 Macro-Level Design Optimization

Conventional efficient Vision Transformers [13, 14, 15] widely employ a 4×4 patch embedding combined with a 4-stage hierarchical network structure. At the macro-level, this configuration introduces spatial redundancy in the early stages of the network, as processing a large token sequence length generates a massive computational bottleneck. SHViT addresses this spatial redundancy by implementing a larger-stride 16×16 non-overlapping patch embedding layer. By utilizing a larger patch stride from the outset, the architecture drastically diminishes the spatial dimensions of the intermediate feature maps, thereby significantly reducing memory access costs and token processing workloads from the very first layer. SHViT also simplifies the global network architecture into a streamlined 3-stage design.

2.4.2 Micro-Level Design Optimization

At the micro-level, empirical evidence demonstrates that attention heads in the earliest network stages primarily capture highly localized, low-level features and operate in a manner equivalent to standard convolutions. Recognizing this operational redundancy, SHViT completely replaces the computationally heavy self-attention layers in the first stage with lightweight depthwise convolutions. Unlike a standard convolution that simultaneously filters and combines features across all channels, a depthwise convolution applies a single spatial filter to each input channel independently. This substitution effectively eliminates unnecessary global dot-product computations in the early stages while preserving the network's ability to process local details.

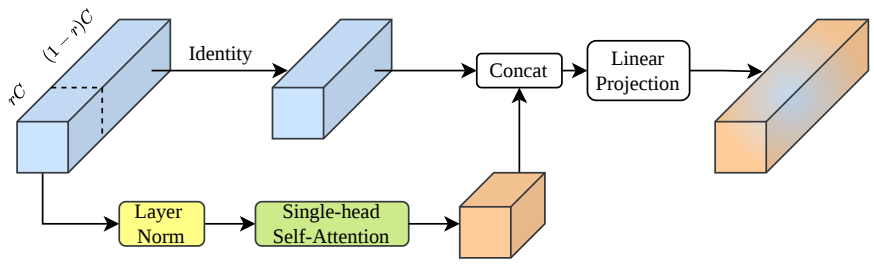


Figure 2.4: Single-Head Self-Attention Module

Furthermore, standard MHSA mechanism independently computes separate attention maps across parallel heads. However, experiments reveal noticeable head redundancy, particularly in the latter stages of the network, where many heads exhibit high similarity and can be removed with only a marginal drop in performance. To prevent the head redundancy, SHViT introduces the Single-Head Self-Attention (SHSA) module, as shown in Fig. 2.4. Instead of utilizing multiple heads, SHSA applies a single attention head to only a partial subset of the input channels ($C_p = rC$) to calculate spatial feature aggregation. The remaining channels ($C - C_p$) are left entirely untouched and passed through an identity connection. By processing only partial channels and minimizing memory-bound operations like reshaping and normalizations, the SHSA module massively reduces computational overhead and memory access costs while successfully capturing global dependencies.

2.5 Joint Self-Attention Denoising Framework

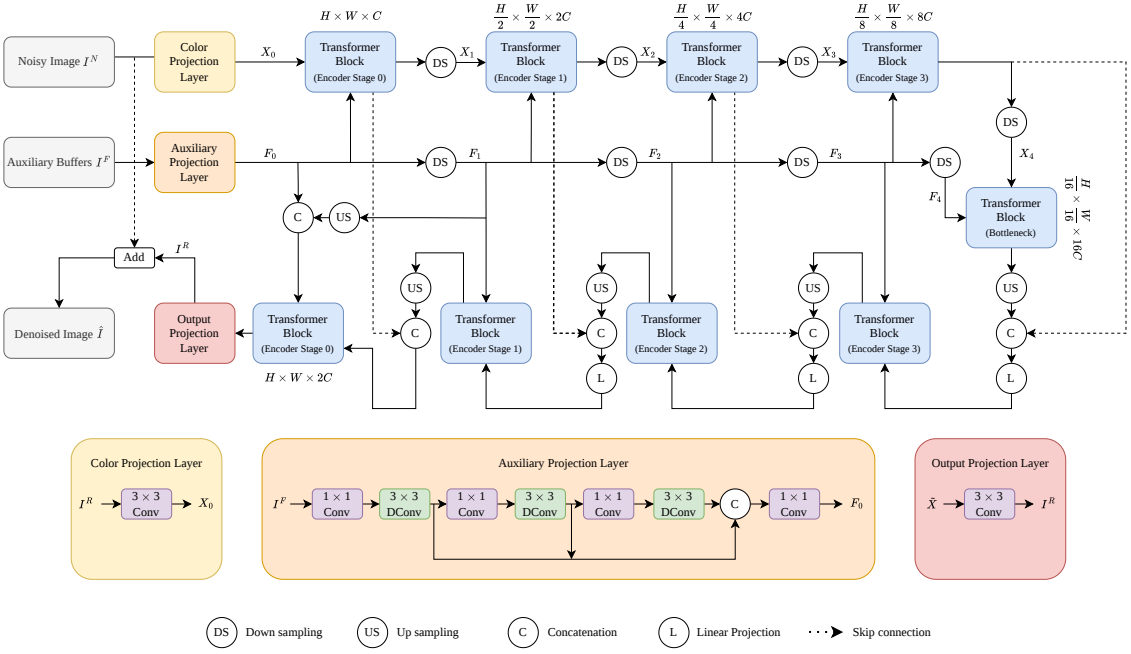


Figure 2.5: Joint Self-Attention Framework

Unlike standard single-input denoisers, rendering-specific image denoising benefits from exploiting auxiliary rendering buffers (G-buffers) alongside the noisy input colors. The Joint Self-Attention (JSA) framework [1] processes these two distinct data sources simultaneously. A standard JSA network architecture is shown in Fig. 2.5.

2.5.1 Multi-Resolution U-Net Architecture

At the macro-level, the model is configured as a U-shaped denoising neural network. This multi-resolution framework is designed to extract, down-sample, and then up-sample features to capture both local details and global context.

- Initial Projections: The network receives two separate inputs: a noisy color image $I^N \in \mathbb{R}^{H\times W\times 3}$ and auxiliary buffers $I^F \in \mathbb{R}^{H\times W\times 7}$ (comprising 3-channel albedos, 3-channel normals, and 1-channel depths). To begin, the framework uses color and auxiliary projection layers to extract an initial color feature map X_0 and an initial auxiliary feature map F_0, both with a channel size of $C = 32$. These projection layers do not perform any patch embedding. Instead, every single pixel directly represents an individual token, maintaining the maximum spatial resolution of $H \times W$ from the very beginning.
- Encoder-Decoder Stages: These initial feature maps are passed through a hierarchical structure consisting of four encoder stages, one bottleneck stage, and four decoder stages. Each stage is built using JSA Transformer blocks. The details of the blocks will be illustrated in next section.

- Resolution Scaling: To aggregate contextual information across different scales, the network utilizes a pixel unshuffle operation for down-sampling in the encoder path, and a pixel shuffle operation for up-sampling in the decoder path. Specifically, pixel unshuffle rearranges an input tensor of shape (H, W, C) into a new tensor of shape $(H/s, W/s, s^2C)$, where s is the scale factor. As the inverse operation, pixel shuffle takes an input tensor of shape (H, W, C) and reshapes it into a high-resolution tensor of shape $(H \cdot s, W \cdot s, C/s^2)$. Compared with other down-sampling operation such as max pooling and other up-sampling operation such as bilinear interpolation, the shuffle/unshuffle operations preserve all feature information, which helps to maintain details to the largest extent in the denoising process.
- Residual Output: After traversing the U-Net structure, the output from the final decoder stage is passed to an output projection layer, which produces a residual image $I^R \in \mathbb{R}^{H \times W \times 3}$. The final denoised image $\hat{I}$ is generated by adding this residual image back to the original noisy input image ($\hat{I} = I^N + I^R$).

2.5.2 Joint Self-Attention Mechanism

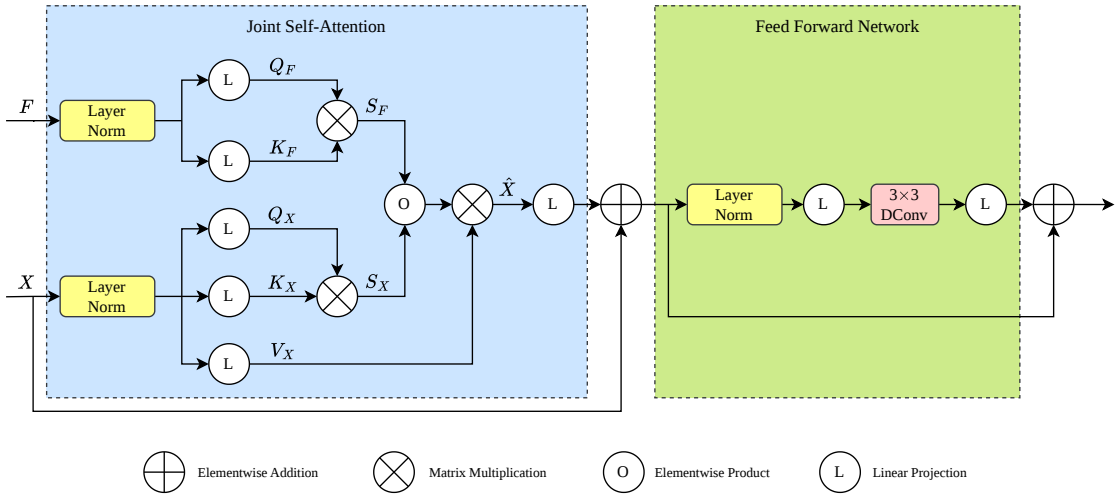


Figure 2.6: Joint Self-Attention (JSA) Transformer Block

Within each stage of the U-Net, the JSA Transformer blocks perform the critical task of identifying pixel-wise similarities across the image space. While standard self-attention infers similarities from a single source, the JSA module extracts dual self-attention scores—one from the noisy colors and the other from the auxiliary buffers—to infer a joint similarity. The mechanism is illustrated in Fig. 2.6.
For any given level l, the block receives a color feature map X_l and an auxiliary feature map F_l. The module computes separate queries (Q) and keys (K) for both feature maps using learnable projection matrices.
The independent attention scores for the color features (S_X) and auxiliary features (S_F) are calculated as follows, where d represents the channel size of the queries and keys, and B represents a learnable bias matrix:

$$S_X = \frac{Q_X K_X^T}{\sqrt{d}} + B \quad S_F = \frac{Q_F K_F^T}{\sqrt{d}} + B$$

To construct the joint self-attention score, $S_{X \cap F}$, the module fuses the dual scores using an element-wise product (denoted by $\circ$):

$$S_{X \cap F} = S_X \circ S_F$$

Finally, the output feature $\hat{X}$ is computed by applying the softmax function to the joint similarity score and multiplying it by the value matrix V_X (which is derived from the color features):

$$\hat{X} = \text{softmax}(S_{X \cap F}) V_X$$

By separately considering the properties of the noisy colors and the shading information in the G-buffers, this joint weighting ensures accurate and robust detail preservation.

2.5.3 Window-based Attention Mechanism

To make the JSA mechanism computationally practical for high-resolution rendering applications, a window-based attention mechanism is integrated into the framework. Computing standard global self-attention leads to a quadratic computational complexity of $\mathcal{O}(L^2 \cdot C)$ relative to the total pixel count, which quickly becomes infeasible for high-resolution denoising. To mitigate this issue, the window-based mechanism partitions the input color feature maps X_l and auxiliary feature maps F_l into a grid of non-overlapping local windows. Rather than calculating pairwise pixel relationships across the entire image sequence, the joint self-attention scores S_X, S_F, and their combined matrix $S_{X \cap F}$ are computed strictly within each localized window. This partitioning strategy successfully linearizes the overall computational complexity to $\mathcal{O}(N \cdot M^2 \cdot C)$, where N is the total number of windows and M is the window size. By confining the intensive dot-product attention operations to localized windows, the network drastically reduces operational latency and memory access costs while still enabling subsequent layers to aggregate broader contextual features.

3

Methods

This chapter outlines the proposed efficient denoising framework, which bridges the gap between high-performance JSA denoising and computationally efficient model architectures. By integrating the macro-level and micro-level design principles of SHViT [12], the proposed framework tries to eliminate potential structural redundancies inherent in the baseline JSA model [1].

3.1 Micro-Level Optimization I: Single-Head Joint Self-Attention Replacement

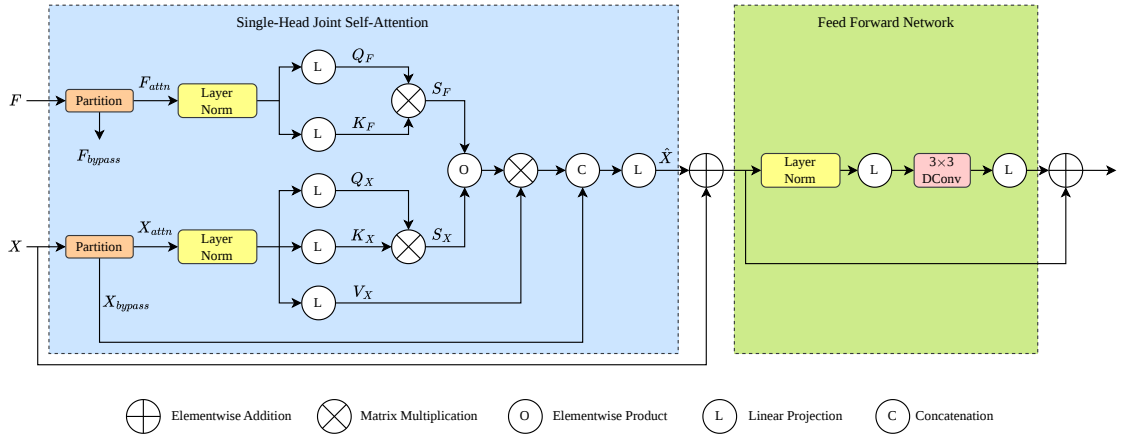


Figure 3.1: Single-Head Joint Self-Attention Block

To resolve the micro-level redundancies identified in the standard Multi-Head Self-Attention mechanism, we introduce the Single-Head Joint Self-Attention (SH-JSA) module into the denoising framework. Instead of projecting the entire feature dimension C into queries, keys, and values, the SH-JSA module utilizes a channel-splitting strategy. Upon receiving the color feature map and the auxiliary feature map, the module divides both inputs along the channel dimension. A specific subset of these channels is routed into a single joint attention head, where the color and auxiliary features are fused to extract the long-range pixel similarities and geometric dependencies. The remaining unselected channels bypass the attention mechanism entirely and are propagated forward through a highly efficient identity shortcut. After the selected channels are processed by the single attention head, they are concatenated

with the untouched shortcut channels to restore the original feature dimension before proceeding to the subsequent Feed-Forward Network. The architectural layout of the proposed SH-JSA module is illustrated in Fig. 3.1.
The proportion of channels processed by the attention head is governed by a partial ratio r (where $C_p = rC$). This ratio serves as a critical hyperparameter in our architecture, dictating the balance between global context modeling and computational efficiency. To achieve the optimal trade-off between floating-point operations and final image reconstruction quality, the most effective value for the partial ratio r will be evaluated and determined through comprehensive experiments detailed in the following chapter.

3.2 Micro-Level Optimization II: Early-Stage Convolutional Replacement

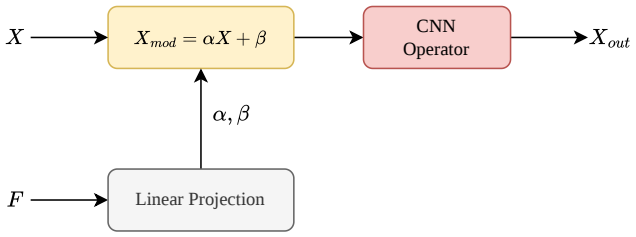


Figure 3.2: FiLM-based CNN Block

As illustrated in previous sections, experiment results indicate that attention heads in early stages primarily operate in a convolution-like manner, focusing on localized details and geometric boundaries rather than long-range global dependencies. To eliminate this spatial redundancy and accelerate inference, we completely replace the computationally heavy JSA Transformer blocks in the first two stages with a lightweight, purely convolutional architecture.
However, a standard convolutional block processes inputs independently, which would cause the network to lose the critical cross-domain fusion capability that defines the JSA framework. To resolve this and simulate the joint attention mechanism efficiently, we integrate a conditioning structure inspired by the Feature-wise Linear Modulation (FiLM) conditioning layer [16]. The proposed block utilizes the auxiliary G-buffers as a conditioning signal to dynamically modulate the color features, as shown in Fig. 3.2. Specifically, the auxiliary feature maps F are processed through convolutional layers to predict two set of modulation parameters: a scaling vector γ and a shifting vector β for each feature channel c. The color feature maps X are then subjected to an affine transformation modulated by the auxiliary parameters:

$$FiLM(X_c|F) = \gamma_c \cdot X_c + \beta_c$$

Furthermore, the specific choice of convolutional operators within this block significantly impacts the overall performance. To ensure maximum efficiency, we investigate two variants of convolutional layers (Fig. 3.3):

1. Standard Convolution: Utilizes standard spatial convolutions, which offer dense channel interactions but incur higher computational costs.
2. Depthwise Separable Convolution: Factorizes the standard convolution into a depthwise spatial convolution followed by a 1×1 pointwise convolution, which can massively reduce the parameter count.

To determine the optimal architectural configuration, a comprehensive study comparing these two convolutional variants will be presented in the subsequent Results chapter.

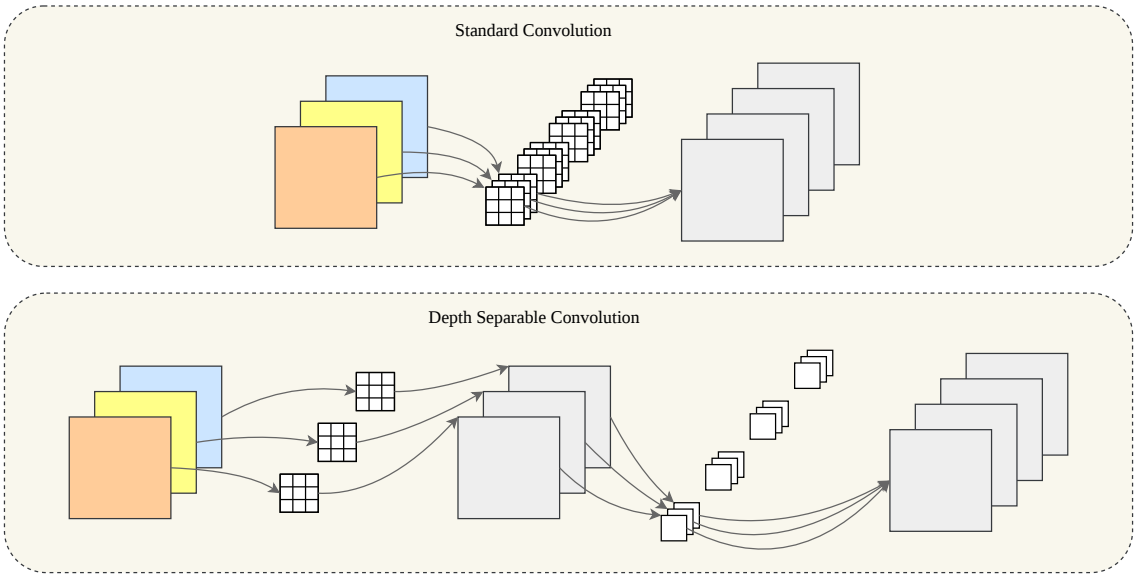


Figure 3.3: Standard Convolution vs. Depth Separable Convolution

3.3 Macro-Level Optimization: Progressive Network Scaling

Beyond convolutional replacements, the original JSA framework suffers from severe structural imbalances and overly conservative tokenization strategies that unnecessarily inflate computational costs. To tailor the architecture for higher efficiency without degrading the final rendering quality, we introduce a progressive network scaling strategy. This strategy systematically restructures the network across three specific dimensions:

- Symmetric Small Decoder Dimension: In the baseline JSA architecture, the highest-resolution level (Stage 0) is highly asymmetrical. While the Encoder Stage 0 processes an input dimension of $H \times W \times C$ using only 1 Transformer block, the corresponding Decoder Stage 0 processes a massively concatenated input of $H \times W \times 2C$ and routes it through 4 consecutive Transformer blocks. Because this stage operates at the peak spatial resolution, deploying a deep, wide decoder here consumes an excessive amount of the total inference time. To resolve this bottleneck, we aggressively scale down Decoder Stage 0, reducing both its channel dimension and block count to perfectly match the lightweight configuration of Encoder Stage 0.

- Reduced Encoder/Decoder Stages: The baseline architecture utilizes a 4-stage hierarchical configuration. We downscale this macro-design to a 3-stage architecture, omitting the deepest low-resolution layers which may offer diminishing returns for high-frequency noise removal.
- Expanded Patch Dimension: A major speed bottleneck in the baseline JSA model is its extremely conservative tokenization strategy. As mentioned in previous chapter, it essentially omits standard Vision Transformer patch embedding by treating every single pixel as an individual token. For high-resolution inputs, this results in an extremely large sequence length, making the computational complexity of any subsequent self-attention mechanism unacceptable for real-time use. To mitigate this, we adopt a more aggressive patch embedding strategy by grouping 2×2 adjacent pixels into a single patch token. This seemingly modest expansion reduces the total sequence length by a factor of four from the outset and quadratically accelerates the attention calculations in the deeper stages.

  The optimized framework based on these three dimensions is depicted in Fig. 3.4.

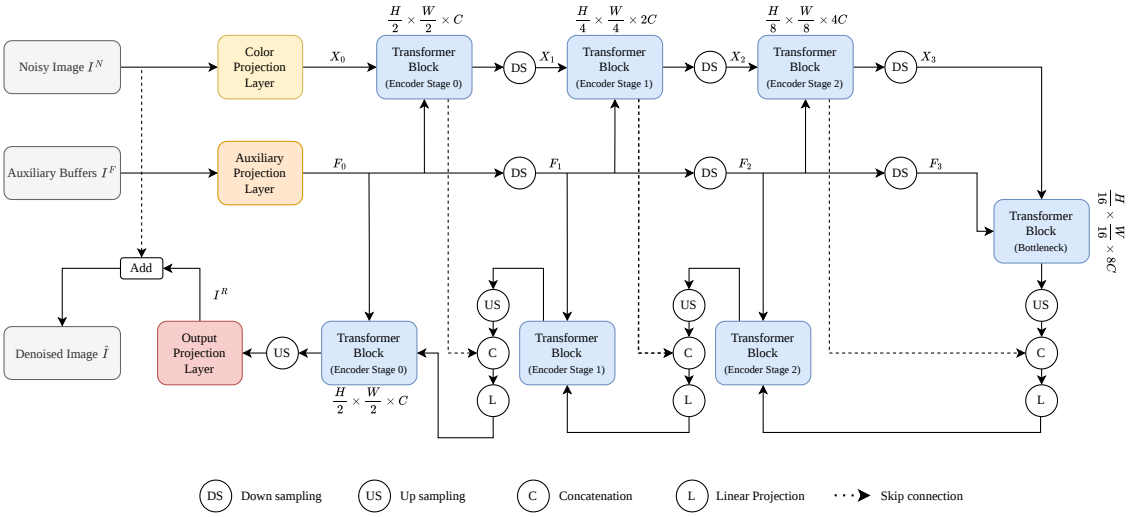


Figure 3.4: Scaled Joint Self-Attention Framework

4

Results

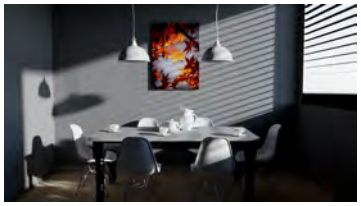
This chapter presents a comprehensive evaluation of the hierarchical optimizations for the denoising framework. Through extensive quantitative analysis, we aim to demonstrate that the proposed architecture successfully reduces computational cost while maintaining image reconstruction quality.

4.1 Experiment Setup

4.1.1 Dataset

(a) Bedroom

(b) Japanese Classroom



(c) Breakfast Room

(d) Pontiac GTO 67

(e) Spaceship

(f) Victorian Style House

(g) Bathroom

(h) Modern Hall

Figure 4.1: Scenes for Training Dataset

The original JSA baseline framework [1] did not publicly release its training and testing dataset. However, the authors noted that they utilized the Tungsten renderer [17] to render several benchmark scenes from Bitterli's rendering resources [18] to construct their data. To ensure a fair evaluation, we replicate this method and make our own dataset using the identical rendering engine.

We select eight 3D scenes from Bitterli's resources (Fig. 4.1). Using the Tungsten renderer, we generate a total of 1451 image pairs. To simulate a time-constrained environment, the noisy input frames are rendered at 32 SPP. The corresponding ground truth frames are rendered at a high sampling rate of 4096 SPP to ensure noise-free images. Each input pair consists of a 3-channel noisy color image and 7 channels of auxiliary G-buffers (3-channel albedo, 3-channel normal, and 1-channel depth maps).

For training the model, this dataset is split into two parts: 1200 image pairs are randomly assigned to the training set, and the remaining 251 pairs are used as the validation set. Furthermore, to evaluate the generalization capability of the trained models, we select ten more unseen 3D scenes to be our testing set.

4.1.2 Evaluation Metrics

Since the primary objective of the work is to balance image reconstruction quality with computational cost, we evaluate the models using two distinct categories of metrics:

1. Image Quality Metrics:
    - Peak Signal-to-Noise Ratio (PSNR): It measures the absolute pixel-level error between the denoised image and the ground truth. A higher PSNR indicates less distortion and better reconstruction accuracy.
    - Structural Similarity Index Measure (SSIM): It evaluates the perceived structural differences such as contrast and luminance. Its values range from 0 to 1, where a higher value indicates stronger structural similarity to the ground truth.
2. Computational Efficiency Metrics:
    - Floating-Point Operations (FLOPs): It quantifies the theoretical mathematical complexity of the network. A lower FLOPs value demonstrates a reduced computational burden.
    - Inference Latency: It measures the actual time required to process a single frame. This metric directly determines the real-time feasibility of the framework.
    - Parameter Count: It indicates the total number of learnable parameters (in millions) within the network.

It should be noted that all results presented in the following sections represent the average values calculated on the ten testing scenes. This averaging strategy ensures a generalized evaluation of each architectural configuration.

4.1.3 Training Details

To ensure a fair comparison, the training strategy of our proposed framework completely follow the protocol established by the JSA baseline. The model is trained in an end-to-end and supervised manner. The network is optimized using the L_2 Loss, which calculates the relative error between the denoised output and the ground truth image. For the optimization process, we utilize the AdamW optimizer. The hyperparameters are set to $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$, with a weight decay factor of 0.02. To guarantee optimization stability in the early iterations and smooth convergence later, we utilize a learning rate scheduler that combines a linear warmup phase with a cosine annealing decay strategy. The initial value of the learning rate is set to 10^{-4}. Then the learning rate increases linearly during the warmup stages follows a cosine curve, smoothly decreasing to a minimum learning rate of $\eta_{min} = 10^{-6}$. All experiments, including model training and testing, are conducted on a single NVIDIA L40S GPU.

4.2 Ablation on Single-Head Self-Attention Replacement

To evaluate the effectiveness of the proposed SH-JSA module, we first conduct an ablation study on the partial channel ratio r. This critical hyperparameter determines the proportion of feature channels processed by the single attention head, while the remaining channels bypass the attention mechanism through an identity shortcut. The quantitative results are presented in Table 4.1.

Table 4.1: SHSA Replacement Results with different partial ratio r

Model	PSNR	SSIM	Params(M)	FLOPs(G)	Inference Time(s)
JSA Baseline	34.845	0.914	33.29	29.20	1.140
SHSA($r = 1.00$)	34.214	0.908	28.06	24.45	1.019
SHSA($r = 0.75$)	34.881	0.916	25.77	22.34	1.009
SHSA($r = 0.50$)	34.353	0.913	24.13	20.83	0.971
SHSA($r = 0.25$)	34.467	0.913	23.14	19.92	0.948

As demonstrated in Table 4.1, replacing the multi-head JSA Baseline with our SH-JSA architecture significantly reduces the computational cost across all configurations. There is a strict positive correlation between the channel ratio r and the computational cost: as r decreases, the network becomes progressively lighter. The most aggressive configuration ($r = 0.25$) successfully drops the parameter count from 33.29M to 23.14M and reduces the FLOPs from 29.20G to just 19.92G. This represents a massive 31.7% reduction in theoretical computational complexity and achieves the fastest inference time of 0.948 s, which validates the design principles of our architecture.

When examining the image quality metrics, an interesting phenomenon occurs at $r = 0.75$, where the model actually outperforms the heavy JSA Baseline in both PSNR and SSIM. This indicates that forcing a portion of the features to bypass the

attention layer acts like a regularization mechanism. It may help to preserve the high-frequency details that might otherwise be overly smoothed by self-attention module. However, the fundamental objective of this thesis is to accelerate inference latency for real-time applications with minimal performance degradation. From this perspective, the $r = 0.25$ configuration emerges as the optimal choice. Although it exhibits a marginal PSNR drop of approximately 0.4 dB compared to the Baseline, it gives the best computational efficiency.

4.3 Ablation on Early-Stage Convolutional Replacement

Inspired by the finding that attention heads in the earliest network stages primarily operate in a convolutional manner, we proposed to replace the early-stage Transformer blocks with FiLM-conditioned CNN blocks. In this section, we systematically evaluate this optimization. We first determine the most efficient convolutional operator through hardware profiling, and then evaluate the performance impact of replacing the initial network stages.

4.3.1 Standard vs. Depthwise Separable Convolution

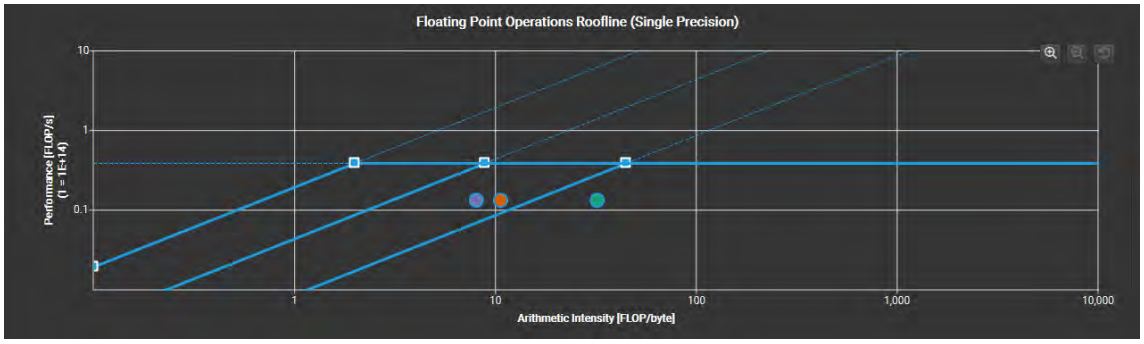
To design the optimal FiLM-CNN block, we evaluate two variants for feature extraction: standard convolution and depthwise separable convolution. Theoretical complexity suggests that depthwise separable convolutions should accelerate inference by factorizing a standard convolution into a depthwise convolution and a subsequent 1×1 pointwise convolution, which can reduce theoretical FLOPs.
However, our latency tests on a simple one-CNN-layer network with high-resolution 1024×1024 inputs reveal a counter-intuitive result: the standard convolution significantly outperforms the depthwise separable convolution in actual execution speed (1.261 ms vs. 1.590 ms).
To investigate this performance discrepancy, we conducted a low-level hardware profiling analysis using NVIDIA Nsight Compute. By analyzing the execution kernels, roofline charts, and memory charts, we identified several hardware-level bottlenecks of depthwise separable convolutions:

- Kernel Execution: Profiling reveals that the cuDNN library heavily optimizes the standard convolution by deploying the Winograd algorithm (`ampere_scudnn_winograd`) and completes the primary operation in just 529.23 μs. In contrast, the depthwise separable convolution must dispatch separate kernels for depthwise processing (`spatialDepthwiseConvolutionUpdateOutput`, 460.23 μs) and channel-wise projection (`implicit_convolve_sgemm`, 404.34 μs). The necessity to launch multiple kernels completely negates the theoretical FLOPs advantage.
- Arithmetic Intensity: As illustrated in Fig. 4.2b, the depthwise separable convolution exhibits a relatively low arithmetic intensity. Since depthwise convolutions lack cross-channel data reuse, the operation is severely memory-bound, which makes it almost touch memory bandwidth ceiling. Conversely,

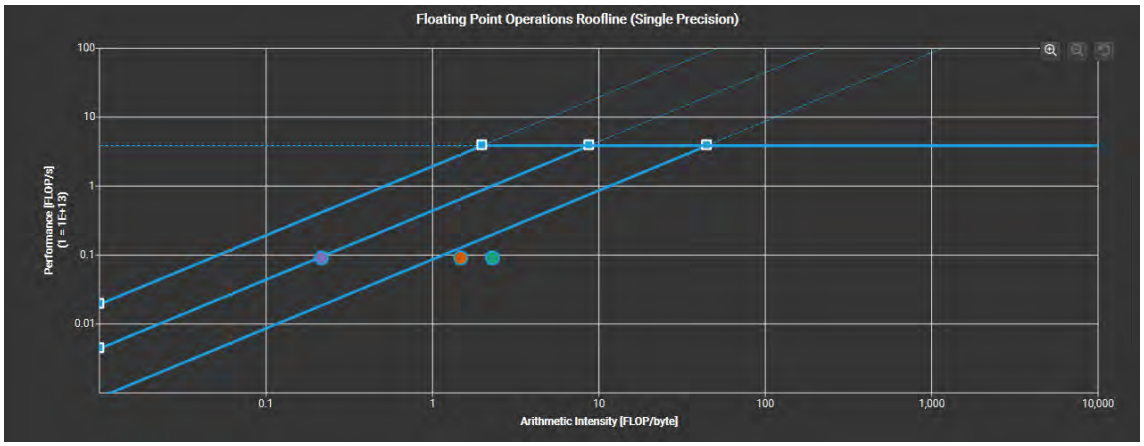
the standard convolution possesses a higher arithmetic intensity, allowing it to utilize the GPU's compute cores more efficiently.

- Memory Hierarchy and Shared Memory Utilization: The memory charts (Fig. 4.3) reveal the most striking difference. The standard convolution efficiently leverages the GPU's ultra-fast Shared Memory (issuing 32.24 Million instructions) to cache intermediate data tiles in the Winograd kernel and yields a high L2 Cache hit rate of 96.40%. In contrast, the depthwise separable convolution completely fails to utilize Shared Memory because its channel-independent nature prevents localized caching. Instead, it is forced to issue a massive 20.96 Million instructions directly to the slower Global Memory, which creates severe memory access latency.

Consequently, we adopt the standard convolution as the core operator for our early stages.

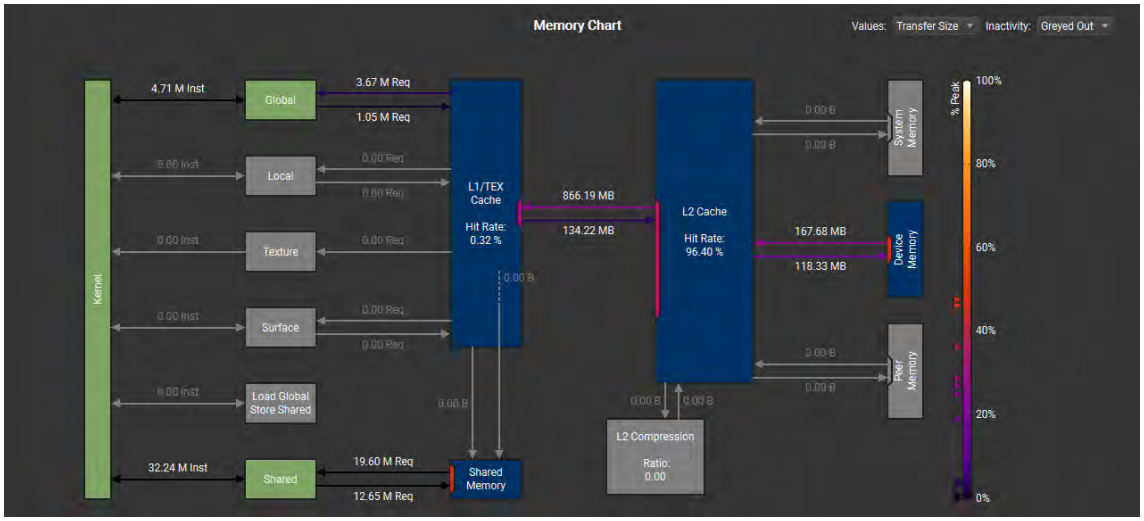


(a) `ampere_scudnn_winograd` Kernel in Standard Convolution

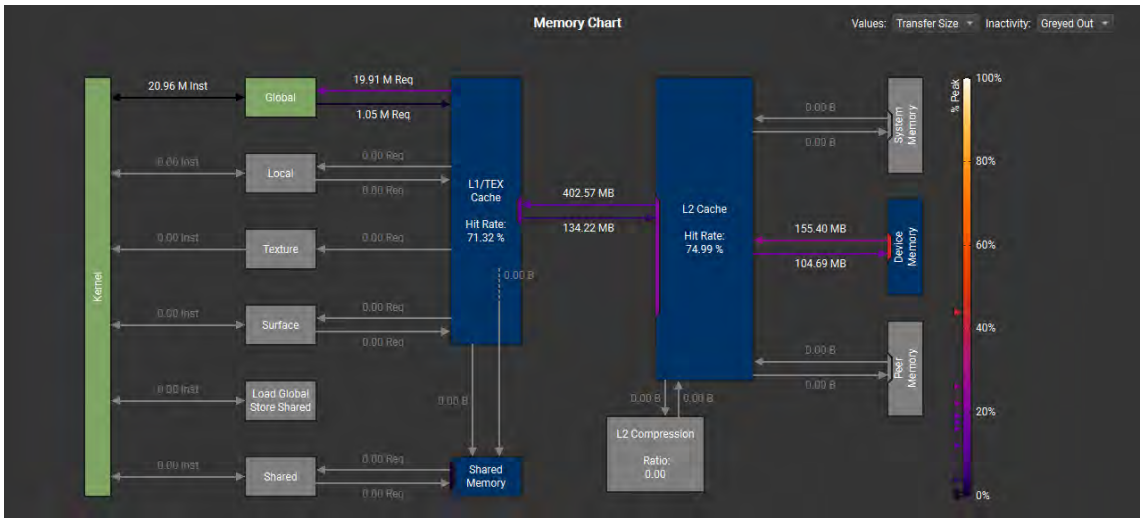


(b) `spatialDepthwiseConvolutionUpdateOutput` Kernel in Depth Separable Convolution

Figure 4.2: Roofline Charts for two CNN Variants



(a) `ampere_scudnn_winograd` Kernel in Standard Convolution



(b) `spatialDepthwiseConvolutionUpdateOutput` Kernel in Depth Separable Convolution

Figure 4.3: Memory Charts for two CNN Variants

4.3.2 Discrepancy Between Computational Complexity and Latency

Having established the standard convolution as the optimal operator, we now evaluate its impact by replacing the initial Transformer blocks in the JSA Baseline with our proposed FiLM-CNN blocks. The results are shown in Table 4.2.

Table 4.2: CNN Replacement Result

Model	PSNR	SSIM	Params(M)	FLOPs(G)	Inference Time(s)
JSA Baseline	34.845	0.914	33.29	29.20	1.140
FiLM-CNN	34.521	0.906	33.43	30.57	1.006

The quantitative results reveal a notable discrepancy. The modified FiLM-CNN configuration actually possesses a slightly larger parameter count and higher theoretical FLOPs compared to the JSA Baseline. However, the actual inference latency is reduced from 1.140 to 1.006s. This discrepancy perfectly aligns with our hardware profiling conclusions. In the early high-resolution stages, the standard MHSA mechanism requires frequent memory reads and writes for its Query-Key-Value projections and softmax operations, which makes it heavily memory-bound. In contrast, standard convolutions leverage highly optimized hardware algorithms and continuous memory access patterns to hide latency. Although this architectural substitution leads to a minor drop in PSNR and SSIM, it is a highly acceptable trade-off for a nearly 12% reduction in latency.

4.3.3 Scalability Analysis: FiLM-CNN vs. SH-JSA

Table 4.3: FiLM-CNN block vs. SH-JSA block at varying spatial resolutions and channel dimensions

Channel	**Resolution**	**FiLM-CNN(ms)**	**SH-JSA(ms)**
32	128	0.342	0.518
32	256	1.027	1.122
32	512	5.754	6.943
32	1024	36.719	40.644
32	2048	152.833	169.780
64	128	0.415	0.435
64	256	1.943	1.973
64	512	11.684	12.337
64	1024	84.065	83.625
64	2048	342.233	341.210

Although the evaluation proves that early-stage CNN replacement accelerates the overall network, a critical question remains: Under what conditions should a stage utilize FiLM-CNN instead of SH-JSA? To answer this, we conduct a benchmark to

compare the inference latency of a single FiLM-CNN block against a single SH-JSA block with varying spatial resolutions and channel dimensions. The results are detailed in Table 4.3. At low channel dimension, the FiLM-CNN block demonstrates absolute superiority across all spatial resolutions. However, when the channel dimension doubles, the computational complexity of standard convolutions explodes due to its quadratic scaling with respect to channels. This empirical evidence validates the structural design of our proposed hybrid U-Net. In the initial Stage 0, the network operates at the maximum spatial resolution but maintains a minimal initial channel capacity. Therefore, deploying FiLM-CNN blocks in this stage yields the maximum possible acceleration. For stages afterwards, deploying SH-JSA blocks is a better choice.

4.4 Ablation on Network Scaling

To evaluate our macro-level structural optimizations, we conduct a progressive study on the network's scaling configuration. In this experiment, each architectural modification builds upon the previous one. The results are detailed in Table 4.4.

Table 4.4: Network Scaling Result

Model	PSNR	SSIM	Params(M)	FLOPs(G)	Inference Time(s)
JSA Baseline	34.845	0.914	33.29	29.20	1.140
+Symmetric Decoder	34.850	0.916	33.05	21.32	0.545
+3 Stage	34.858	0.917	6.01	12.19	0.467
+Expanded Patch	34.466	0.906	6.07	3.70	0.115

Impact of the Symmetric Decoder The original JSA Baseline employs a heavy initial stage of the decoder that creates a massive bottleneck during the up-sampling phase. By aligning its channel dimension with the initial stage of the encoder, we achieve an immediate acceleration. The inference latency is cut by more than half, dropping from 1.140 to 0.545 s. Meanwhile, the FLOPs decrease from 29.20G to 21.32G. Remarkably, there is even a slight increase in PSNR and SSIM. This proves that the heavy capacity of the baseline's decoder is entirely redundant.
Impact of Network Stage Reduction Building upon the symmetric decoder, we further aggressively remove the deepest layer of the U-Net architecture, reducing the framework from 4 stages to 3 stages. The result reveals an astonishing breakthrough: instead of dropping, the PSNR reaches its peak at 34.858. This indicates that the deepest 4th stage in the original baseline was not only computationally wasteful but likely causing overfitting. By removing it, the network preserves spatial fidelity better while accelerating inference to 0.467 s.
Impact of Expanded Patch Size In the final step, we integrate the expanded patch size strategy to further alleviate the computational burden of the self-attention mechanisms. This optimization delivers a performance leap: the theoretical FLOPs are reduced to just 3.70G and the inference latency reaches 0.115 s. While this extreme spatial compression introduces a minor PSNR drop to 34.466, it represents an exceptional trade-off.

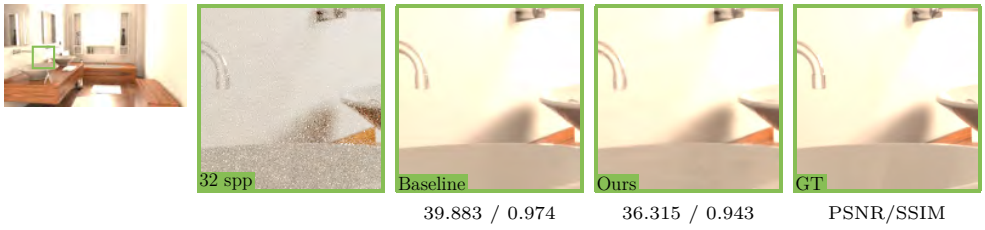
4.5 Overall Performance: The Unified Hybrid Architecture

In the preceding sections, we validated individual architectural improvements, including the partial-channel SH-JSA replacement, early-stage FiLM-CNN replacement, and progressive macro-level network scaling. In this final evaluation, we integrate all these optimal configurations into a single network. This unified architecture features a 3-stage symmetric U-Net, utilizes standard convolutions in the first stages of encoder and decoder, deploys the SH-JSA ($r = 0.25$) in the deeper stages, and incorporates expanded patch sizes. We compare this ultimate configuration against the original JSA Baseline to assess the cumulative impact of our work. The results are presented in Table 4.5. The unified framework compresses the parameter count from 33.29M to merely 4.44M. Simultaneously, the theoretical computational complexity (FLOPs) drops from 29.20G to just 3.03G. Most importantly, the actual inference latency is sharply reduced from 1.140 seconds to 0.096 seconds. This represents a near 12-fold acceleration. To achieve this performance, the network makes a necessary and strategic trade-off with image reconstruction quality. The PSNR drops from 34.845 to 33.151, and the SSIM shifts from 0.914 to 0.864.

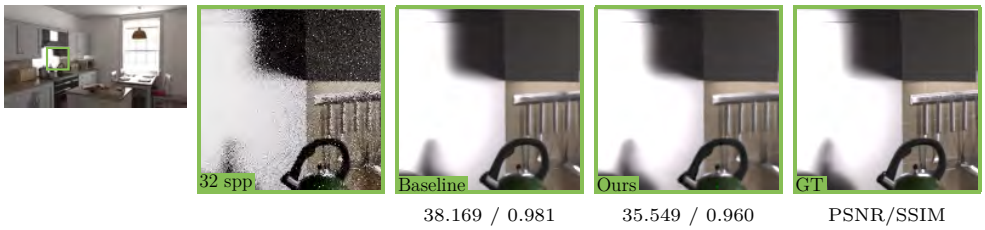
Table 4.5: Results for the Hybrid Architecture

Model	PSNR	SSIM	Params(M)	FLOPs(G)	Inference Time(s)
JSA Baseline	34.845	0.914	33.29	29.20	1.140
Hybrid	33.151	0.864	4.44	3.03	0.096

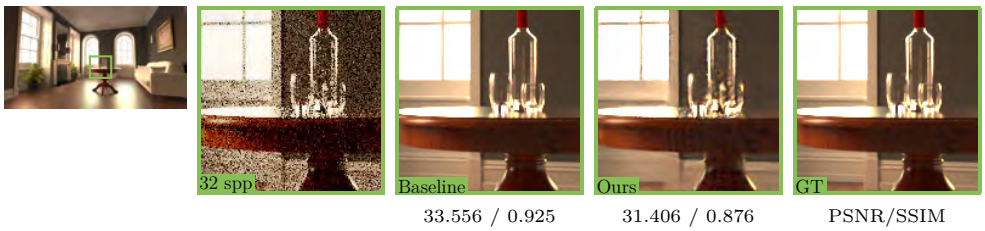
To further evaluate the reconstruction quality of the proposed unified framework, we provide a detailed visual comparison across representative testing scenes in Fig. 4.4 and Fig. 4.5. For indoor environments such as Fig. 4.4a and Fig. 4.4b as well as structurally simple scenes like Fig. 4.5c and Fig. 4.5d, the framework demonstrates outstanding denoising capabilities. These scenes are primarily dominated by low-frequency global illumination, large flat textures, and distinct geometric boundaries. Conversely, the qualitative results reveal noticeable limitations when processing scenes with extreme high-frequency details, such as the thin, overlapping geometry in Fig. 4.5e or the complex internal refractions in Fig. 4.5b. In these specific cases, the denoised outputs appear over-smoothed, resulting in a loss of fine-grained details compared to the ground truth. This phenomenon is a direct consequence of our aggressive architectural trade-offs. To achieve the latency under $0.1s$, the framework relies heavily on expanded patch sizes and reduced attention channels. While this effectively compresses the computational cost, it inevitably restricts the network's capacity to aggregate and preserve the micro features, causing the network to lean towards a slightly blurred, conservative reconstruction in highly complex regions.



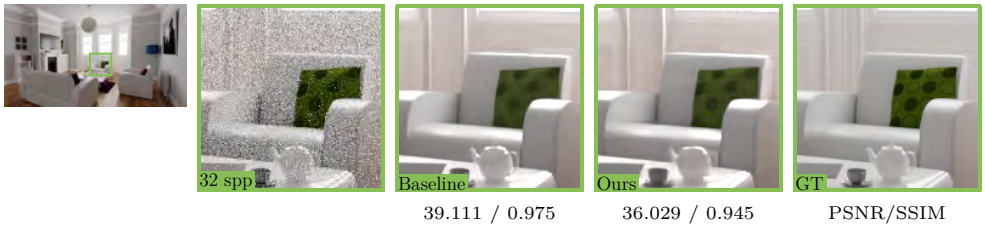
(a) Contemporary Bathroom



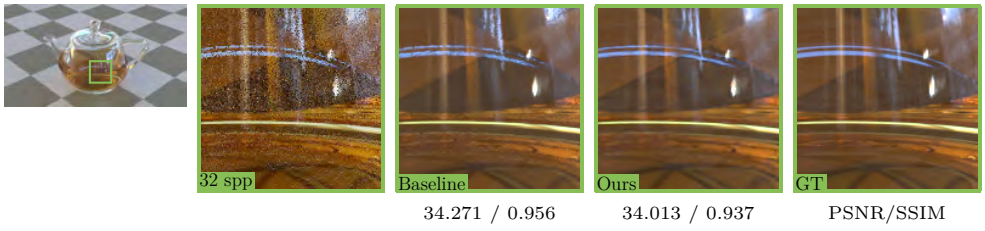
(b) Country Kitchen



(c) The Grey & White Room



(d) The White Room

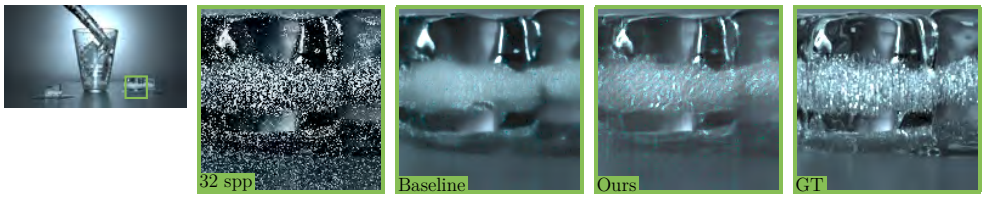


(e) Utah Teapot

Figure 4.4: Visual comparison of denoising performance across multiple scenes. For each scene, we show the full reference image, the noisy input, the baseline, our work, and the ground truth. Quantitative metrics (PSNR / SSIM) are reported below the patches. (Part 1)

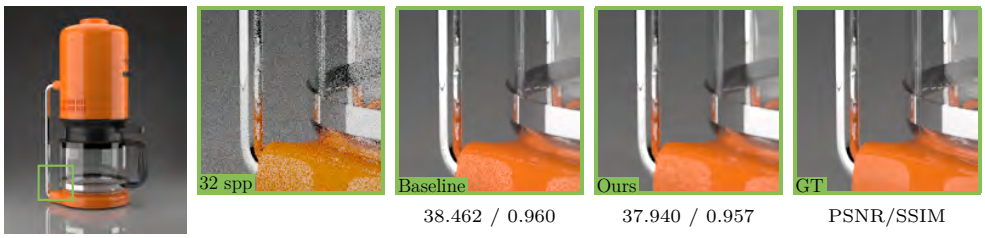
26.451 / 0.714 24.986 / 0.624 PSNR/SSIM

(a) Veach Ajar



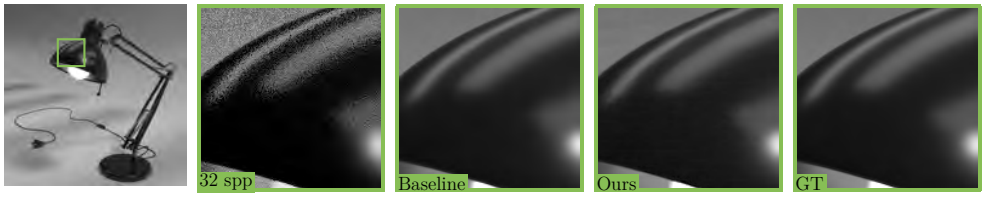
26.348 / 0.915 27.634 / 0.904 PSNR/SSIM

(b) Glass of Water



38.462 / 0.960 37.940 / 0.957 PSNR/SSIM

(c) Coffee Maker



42.301 / 0.994 39.124 / 0.985 PSNR/SSIM

(d) Little Lamp

29.902 / 0.751 28.517 / 0.509 PSNR/SSIM

(e) Curly Hair

Figure 4.5: Visual comparison of denoising performance across multiple scenes. For each scene, we show the full reference image, the noisy input, the baseline, our work, and the ground truth. Quantitative metrics (PSNR / SSIM) are reported below the patches. (Part 2)

5
Conclusion

This chapter first discusses the practical limitations of the current framework along with directions for future research. It then evaluates the broader ethical implications of this study. Finally, it concludes the thesis by summarizing its key contributions and results.

5.1 Limitation and Future Work

Although the proposed unified framework significantly accelerates Monte Carlo denoising, it still exhibits several practical limitations:

- High Input Sampling Rate (32 SPP): Currently, the network relies on 32 SPP as its noisy input. In strict real-time rendering environments, generating 32 SPP within a single frame is still too computationally expensive. Practical real-time engines typically operate under extreme budgets of 1 SPP to 4 SPP.
- Inference Speed for Real-Time Rendering: Although the framework drastically reduces inference latency to 0.096 seconds, which is equivalent to approximately 10.4 Frames Per Second (FPS), this speed still does not meet the strict 30 FPS standard required for real-time rendering applications, such as modern video games or virtual reality.
- Loss of High-Frequency Details: As observed in the qualitative analysis, the aggressive optimizations limit the network's capacity. Consequently, the model struggles to accurately reconstruct high-frequency details, often causing over-smoothing in complex regions.

To address these limitations, the most critical direction for future work is the transition from a purely spatial architecture to a spatiotemporal denoising framework. By integrating temporal information, the network can accumulate rendering samples across consecutive frames using motion vectors. This temporal accumulation effectively increases the sample count without requiring the rendering engine to generate more rays per frame, thus solving the 32 SPP bottleneck. Furthermore, historical frame data can serve as an extra input to the denoising framework and provide the rich structural information necessary to reconstruct high-frequency details that a single spatial frame might over-smooth.

5.2 Ethical Considerations

As computer graphics and deep learning technologies become increasingly integrated into daily life, it is crucial to evaluate the ethical implications of this research. The

ethical considerations for this thesis are discussed in the following three aspects:

- Environmental sustainability: The contribution of this thesis aligns with the principles of sustainable computing. The proposed lightweight framework successfully eliminates the computational redundancy in the standard JSA framework by reducing the parameter count and the required FLOPs. Consequently, this significantly decreases the energy consumption required for rendering and inference and promotes a more environmentally sustainable approach to utilizing AI in computer graphics.
- Data privacy: The dataset used to train and evaluate the models in this thesis mitigates the risk of data privacy by design. The dataset entirely consists of synthetic 3D scenes generated by the Tungsten renderer. Since all generated data does not contain any real personal information, privacy violation is completely avoided.
- Potential misuse: Advanced technologies for generating highly realistic images can theoretically be misused to create deceptive media. However, since this research only focuses on post-processing denoising within rendering pipelines rather than AI generation from scratch, its risk of being used for deepfake is extremely low.

5.3 Conclusion

Despite the aforementioned limitations, the primary objective of this thesis, which is to accelerate Monte Carlo denoising without severely losing image quality, was successfully achieved. By systematically addressing the computational redundancy inherent in the standard JSA framework, this thesis developed a highly efficient denoising architecture.

We achieved this through two levels of architectural innovation:

- Micro-Level Optimizations: We introduced the SH-JSA module with a partial channel ratio of $r = 0.25$. This design preserved critical geometric features from auxiliary G-buffers while drastically reducing computational complexity. Furthermore, guided by low-level hardware profiling, we replaced the memory-bound early-stage attention blocks with optimized FiLM-CNN blocks.
- Macro-Level Optimizations: We progressively optimized the global structure by implementing a symmetric decoder, reducing the network depth to three stages, and expanding the input patch size.

The integration of these optimizations yielded groundbreaking results. For a high-resolution 1024×1024 input, the final unified framework reduced the parameter count by 86.6% (from 33.29M to 4.44M) and slashed the inference latency by 91.6% (from 1.140s to 0.096s) with only a drop by 4.9% in image quality (from 34.845dB to 33.151dB in PSNR). Ultimately, this thesis provides a robust architecture for bridging the gap between high-fidelity denoising and efficiency demands of interactive rendering.

Bibliography

[1] G. Oh and B. Moon, "Joint self-attention for denoising monte carlo rendering," *The Visual Computer*, vol. 40, no. 7, pp. 4623–4634, 2024.

[2] J. T. Kajiya, "The rendering equation," in *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, 1986, pp. 143–150.

[3] S. Bako, T. Vogels, B. McWilliams, M. Meyer, J. Novák, A. Harvill, P. Sen, T. Derose, and F. Rousselle, "Kernel-predicting convolutional networks for denoising monte carlo renderings." *ACM Trans. Graph.*, vol. 36, no. 4, pp. 97–1, 2017.

[4] C. R. A. Chaitanya, A. S. Kaplanyan, C. Schied, M. Salvi, A. Lefohn, D. Nowrouzezahrai, and T. Aila, "Interactive reconstruction of monte carlo image sequences using a recurrent denoising autoencoder," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–12, 2017.

[5] X. Meng, Q. Zheng, A. Varshney, G. Singh, and M. Zwicker, "Real-time monte carlo denoising with the neural bilateral grid." in *EGSR (DL)*, 2020, pp. 13–24.

[6] H. Fan, R. Wang, Y. Huo, and H. Bao, "Real-time monte carlo denoising with weight sharing kernel prediction network," in *Computer Graphics Forum*, vol. 40, no. 4. Wiley Online Library, 2021, pp. 15–27.

[7] M. Işık, K. Mullia, M. Fisher, J. Eisenmann, and M. Gharbi, "Interactive monte carlo denoising using affinity of neural features," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, pp. 1–13, 2021.

[8] M. Balint, K. Wolski, K. Myszkowski, H.-P. Seidel, and R. Mantiuk, "Neural partitioning pyramids for denoising monte carlo renderings," in *ACM SIGGRAPH 2023 conference proceedings*, 2023, pp. 1–11.

[9] H. Fan, R. Wang, Y. Huo, and H. Bao, "Real-time monte carlo denoising with weight sharing kernel prediction network," in *Computer Graphics Forum*, vol. 40, no. 4. Wiley Online Library, 2021, pp. 15–27.

[10] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[12] S. Yun and Y. Ro, "Shvit: Single-head vision transformer with memory efficient macro design," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024, pp. 5756–5767.

[13] P. K. A. Vasu, J. Gabriel, J. Zhu, O. Tuzel, and A. Ranjan, "Fastvit: A fast hybrid vision transformer using structural reparameterization," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 5785–5795.

[14] Y. Li, G. Yuan, Y. Wen, J. Hu, G. Evangelidis, S. Tulyakov, Y. Wang, and J. Ren, "Efficientformer: Vision transformers at mobilenet speed," *Advances in neural information processing systems*, vol. 35, pp. 12 934–12 949, 2022.

[15] Y. Li, J. Hu, Y. Wen, G. Evangelidis, K. Salahi, Y. Wang, S. Tulyakov, and J. Ren, "Rethinking vision transformers for mobilenet size and speed," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 16 889–16 900.

[16] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[17] B. Bitterli, "Tungsten renderer," 2014, https://benedikt-bitterli.me/tungsten.html.

[18] ——, "Rendering resources," 2016, https://benedikt-bitterli.me/resources/.

A

Appendix 1