Dynamic Neural Radiosity with Multi-grid Decomposition

RUI SU, School of Computer Science, Peking University, China
HONGHAO DONG, School of Computer Science, Peking University, China
JIERUI REN, College of Future Technology, Peking University, China
HAOJIE JIN, School of Computer Science, Peking University, China
YISONG CHEN, School of Computer Science, Peking University, China
GUOPING WANG*, School of Computer Science, Peking University, China
SHENG LI*, School of Computer Science, Peking University, China

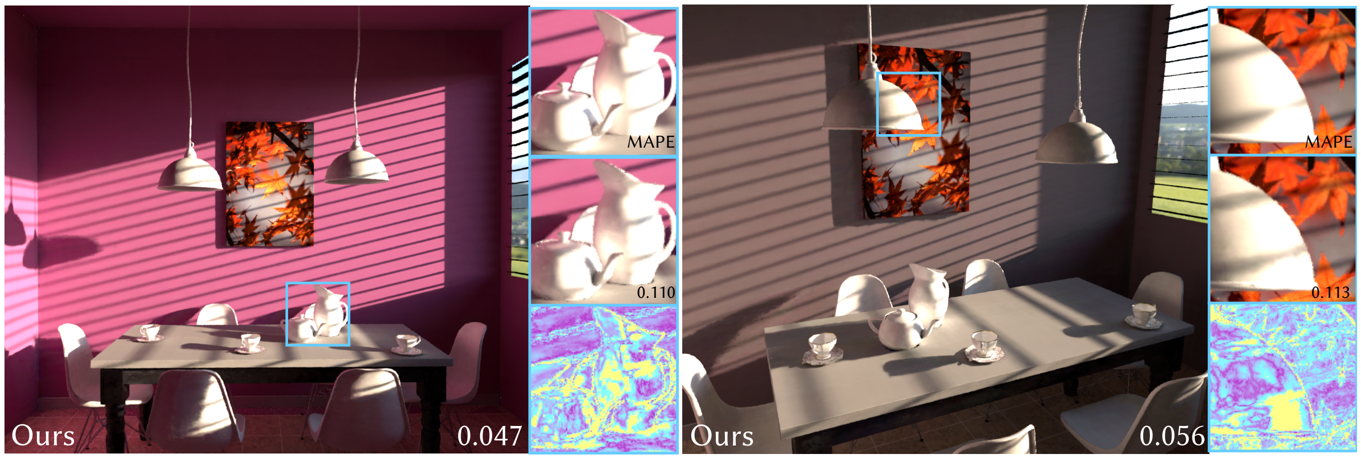

Fig. 1. Our method renders scenes during interactive editing of various types of animations, such as material property changes and geometry transformations, while supporting free viewpoint movement. Dining Room scene is rendered at about 10 fps (1080 × 920), with varying view points and environment lighting.

Prior approaches to the neural rendering of global illumination typically rely on complex network architectures and training strategies to model the global effects. This often leads to impractically high overheads for both training and inference. The neural radiosity technique marks a significant advancement by injecting the radiometric prior into the training process, allowing for efficient modeling of the global radiance fields using a lightweight network and grid-based representations. However, this method encounters difficulties in modeling dynamic scenes, as the high-dimensional feature space quickly becomes unmanageable as the number of varying scene parameters grows. In this work, we extend neural radiosity for variable scenes through a novel neural decomposition method. To achieve this, we first parameterize the animated scene with an explicit vector $\mathbf{v}$, which conditions a high-dimensional radiance field L_θ. We then develop a practical representation for L_θ by decomposing the high-dimensional feature grid into 3D grids, 2D feature planes, and lightweight MLPs. This strategy effectively models the correlation between 3D spatial features and dynamic scene variables, while maintaining a practical memory and computational cost. Experimental results show that our method facilitates efficient dynamic global illumination rendering with practical runtime performance, outperforming previous state-of-the-art techniques with both reduced training and inference costs.

CCS Concepts: • **Computing methodologies** → **Ray tracing**; **Neural networks**.

Additional Key Words and Phrases: Neural Rendering, Radiosity, Dynamic Scene, High-dimensional Feature, Global Illumination

ACM Reference Format:
Rui Su, Honghao Dong, Jierui Ren, Haojie Jin, Yisong Chen, Guoping Wang, and Sheng Li. 2024. Dynamic Neural Radiosity with Multi-grid Decomposition. In *SIGGRAPH Asia 2024 Conference Papers (SA Conference Papers '24), December 3–6, 2024, Tokyo, Japan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3680528.3687685

*corresponding author.
Project website: https://woaixuexisr.github.io/papers/dynamic-neural-radiosity.

1 INTRODUCTION

Global illumination has always been a challenging task in computer graphics. Recent developments in deep learning have also led to the use of neural networks for rendering high-quality images, particularly in dynamic scenes involving animated variables like meshes, materials, and light sources [Baatz et al. 2022; Kallweit et al. 2017; Zhu et al. 2021]. Some of these methods operate in screen space [Işık et al. 2021; Nalbach et al. 2017], exploiting the powerful capabilities of deep generative models. However, this also introduces typical limitations of the screen-space techniques, such as view-dependent artifacts and difficulties in modeling complex global effects. Another family of algorithms merges the advantages of screen-space techniques and the burgeoning field of neural scene representation techniques [Mildenhall et al. 2021]. These methods often involve intricate network architectures and sophisticated training processes, which result in significant computational overheads for network inference [Diolatzis et al. 2022; Granskog et al. 2021; Zheng et al. 2023]. Additionally, both approaches require training deep neural networks on a large set of synthetic images, which requires considerable time for both dataset generation and network training.

A promising solution to addressing these challenges is the neural radiosity technique [Hadadan et al. 2021]. This method efficiently models the radiance field by incorporating a radiometric prior into network optimization, utilizing lightweight neural networks and trainable spatial feature encoding. Apart from the theoretical simplicity and ease of implementation, its key advantage resides in its independence from pre-rendered images, allowing the expressiveness of a view-consistent radiance field. However, while effective for static scenes, the neural radiosity technique encounters difficulty with dynamic or animated scenes due to the need for frequent updates and retraining. This also limits its potential application in real-time rendering pipelines.

In this work, we tackle these challenges and extend the neural radiosity technique to accommodate dynamic scenes. We first encode the dynamic scene using an *explicit scene vector* $\mathbf{v}$, wherein each element parameterizes an animated scene variable. To model the animated radiance field, a straightforward approach is to use $\mathbf{v}$ to condition the network output, which often yields suboptimal and under-fitted results. To overcome this, we suggest treating each scene component as an additional dimension beyond the 3D Cartesian coordinates. This results in a possible solution to dynamic neural rendering, where we can utilize high-dimensional feature grids as learnable encoding to implicitly model the radiance fields within the multi-variable dynamic scene.

However, the curse of dimensionality inflicts significant challenges to this method, as a high-dimensional feature grid leads to an impractical level of memory and computational cost. Inspired by previous work on viewpoint synthesis for dynamic scenes [Cao and Johnson 2023; Fridovich-Keil et al. 2023; Shao et al. 2023], we decompose high-dimensional latent grids into low-dimensional representations, where 2D feature planes and lightweight MLPs are used to model the correlation between scene variables. This decomposition effectively models multi-variate dynamic scenes while maintaining reasonable runtime performance. Our implementation achieves $>$ 10 fps when rendering at 1280×720 on a single RTX3090, achieving $3.5\times$ speedup on frame rate with *better* reconstruction quality compared to the previous state-of-the-art.

Overall, our main contributions are as follows:

- Our approach extends the neural radiosity approach to handle high-dimensional dynamic scenes with neural rendering.
- We explicitly encode dynamic scenes using *scene vector*, which enables the use of high-dimensional grid-based features for scene representations.
- Our feature decomposition strategy uses low-dimensional implicit representations to model the correlations between scene variables, avoiding the impractical cost of high-dimensional representations caused by the dimensional curse.

2 RELATED WORK

Neural Rendering. Deep learning techniques have been widely applied to assist and enhance rendering applications (i.e. neural rendering). Earlier works tend to use image-space techniques and employ generative networks for post-process tasks such as denoising [Işık et al. 2021; Vogels et al. 2018] and up-scaling [Wu et al. 2023; Zhong et al. 2023], which often rely on high-quality ground truth images for training. Convolutional networks can also simulate screen-space effects, such as diffuse indirect lighting and motion blur [Nalbach et al. 2017]. Neural networks have also been leveraged to enhance the Monte Carlo importance sampling process [Dong et al. 2023; Müller et al. 2019; Zheng and Zwicker 2019], employing specialized network architectures to directly learn to sample complex manifolds.

Another emerging research trend focuses on encoding scene features with implicit representations and then decoding them into synthetic images with neural networks. Ren et al. [2013] utilized a fixed set of precomputed renderings to train neural networks, approximating indirect illumination with dynamic lights. Müller et al. [2021] used a lightweight network with online training as the radiance cache to approximate indirect illumination for real-time path tracing. Eslami et al. [2018] trained an encoder-decoder network on a variable scene by encoding multiple observations into a scene representation vector. Granskog et al. [2020] extended this idea by using an adaptively partitioned scene representation containing disentangled information of lighting, material, and geometry. However, their method still needs additional views for novel-view synthesis. Zheng et al. [2023] used networks to model the light transfer function between individual objects and the environment, subsequently composing them to produce the global illuminated images.

Recently, Hadadan et al. [2021] introduced neural radiosity, exploiting the rendering equation as a prior for network optimization. While their method effectively models radiance fields using lightweight networks, it is restricted to static scenes. More recently, Coomans et al. [2024] extended neural radiosity to dynamic scenes by learning a 1D manifold within the dynamic parameter space, restricting them to a 4D space. Our work aims to further extend neural radiosity to handle high-dimensional dynamic scenes without being confined to one additional dimension, while maintaining only a moderate increase in computational and storage overhead. Diolatzis et al. [2022] encoded scene animations with an explicitly parameterized vector to condition the network predicted radiance field, serving a similar purpose to our method. By incorporating MCMC-guided training, their method achieves state-of-the-art reconstruction quality for the neural rendering of dynamic scenes. However, their method still requires path-traced image patches for training. Moreover, their method requires extensive computational overhead due to their deep network. By contrast, our method achieves better visual quality with a $3.5\times$ frame rate boost by developing a trainable encoding with our feature decomposition technique.

Network Encoding. The encoding of inputs to neural networks is crucial for their performance. Mildenhall et al. [2021] applied Fourier feature mapping to project 3D coordinates to a high-dimensional space before feeding into the network, which is verified to be vital for modeling high-frequency details in complex scenes [Tancik et al. 2020]. Another set of approaches employ trainable encoding with multi-resolution spatial grids [Hadadan et al. 2021] or hash tables [Müller et al. 2022], therefore reducing the network's burden by using sparse embedding to model spatial features. This concept is later extended to applications on dynamic scenes. Li et al. [2022] incorporated temporal latent encoding into the network input. Fridovich-Keil et al. [2023], Cao and Johnson [2023], and Shao et al. [2023] used 4D voxel feature grids to represent dynamic scenes. By decomposing the straightforward 4D grids into multiple 2D planes, they achieve the goal of reducing spatial complexity while maintaining model quality. In our work, we parameterize dynamic scenes with an n-dim explicit vector, enabling interactive editing and free interpolating across multiple animated components, instead of just one time dimension. With the aforementioned techniques, this leads to quadratic memory and computational costs, which quickly becomes unmanageable as n grows. In this work, we model the high-dimensional scenes with linear computational and memory costs regarding the scene dimensionality. The key insight is that it is sufficient to model the $O(n^2)$ pairwise correlations between the additional scene variables using non-linear lightweight MLPs, which naturally fit into neural rendering applications.

Radiosity Algorithm. The radiosity algorithm is first suggested by Goral et al. [1984] to solve indirect illumination within diffuse scenes, with a detailed system introduction provided by Sillion and Puech [1994]. Similar to the finite element method, their approach involves dividing the scene into discrete patches and computing the light transfer across these patches, assuming uniform radiance distribution on each. Their method is extended to non-diffuse scenes by Immel et al. [1986]. To address the complexity of realistic lighting, Zatz [1993] applied the Galerkin integral equation to approximate surface radiance as higher-order polynomials. However, the complex visibility among scene objects creates discontinuities that are difficult to model, leading to further research on different basis functions to better fit the radiance distributions [Cohen et al. 1988; Gortler et al. 1993; Lehtinen et al. 2008; Lischinski et al. 1992]. More recently, Hadadan et al. [2021] introduced machine learning to improve the fitting of radiance distributions within the scenes. Despite these advancements, prior methods mainly focus on static scenes. Our work extends the radiosity technique to higher-dimensional dynamic environments while maintaining practical overhead.

3 BACKGROUND

3.1 Rendering Equation

Physical light transport simulation fundamentally relies on the rendering equation [Kajiya 1986], which formulates the outgoing radiance as the sum of integrated incident radiance and self-emission:

$$L(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_{\mathcal{H}^2} f_r(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i) \left|(\mathbf{n} \cdot \omega_i)\right| \mathrm{d}\omega_i, \quad (1)$$

where $L(\mathbf{x}, \omega_o)$ denotes the outgoing radiance at point $\mathbf{x}$ in direction ω_o, and $L_e(\mathbf{x}, \omega_o)$ denotes the radiance emitted from point $\mathbf{x}$ in direction ω_o. The integral accumulates the incoming radiance $L_i(\mathbf{x}, \omega_i)$ from all directions over the hemisphere $\mathcal{H}^2$ around the normal, modulated by the bidirectional reflectance distribution function (BRDF) $f_r(\mathbf{x}, \omega_i, \omega_o)$.

The rendering equation is recursive in nature, as the incoming radiance at a point is itself the outgoing radiance from other points in the scene, which makes solving the integral challenging.

3.2 Neural Radiosity

To solve the rendering equation, Hadadan et al. [2021] proposed Neural Radiosity by exploiting the rendering equation as a prior for network training. They train a neural network with parameter θ to approximate the outgoing radiance $L(\mathbf{x}, \omega_o)$. The neural network is optimized by minimizing the norm of the rendering equation residual, which is the left-hand side (LHS) minus the right-hand side (RHS):

$$\begin{aligned} r_\theta(\mathbf{x}, \omega_o) = L_\theta(\mathbf{x}, \omega_o) - L_e(\mathbf{x}, \omega_o) \\ - \int_{\mathcal{H}^2} f_r(\mathbf{x}, \omega_i, \omega_o) L_\theta(\mathbf{x}'(\mathbf{x}, \omega_i), -\omega_i) \left|(\mathbf{n} \cdot \omega_i)\right| \mathrm{d}\omega_i, \end{aligned} \quad (2)$$

where $\mathbf{x}'((\mathbf{x}, \omega_i), -\omega_i)$ means the closest surface intersection of the ray from $\mathbf{x}$ in direction ω_i. The reflected radiance (the last integral in Eq. 2) is estimated with additionally traced right-hand side rays (RHS rays) using Monte Carlo integration, while the radiance value is queried by the network prediction $L_\theta(\mathbf{x}, \omega_o)$.

This optimization is performed using a mean squared error (MSE) loss that minimizes the residual of the rendering equation (Eq. 2) across various points and directions in a scene:

$$\theta^* = \arg\min_\theta \int_{\mathcal{S}} \int_{\mathcal{H}^2} r_\theta(\mathbf{x}, \omega_o)^2 \mathrm{d}\omega_o \mathrm{d}\mathbf{x}, \quad (3)$$

where $\mathcal{S}$ is the scene surfaces and $\mathcal{H}^2$ is the upper-hemisphere over the shading point $\mathbf{x}$.

By leveraging the radiometric prior for network optimization, Neural Radiosity efficiently models the global radiance field with lightweight networks and trainable spatial feature grids, while being view consistent and without using any image-space techniques. Based on its principles, our work extends neural radiosity to dynamic scenes by designing a practical encoding for high-dimensional animated scenes.

4 METHOD

In this section, we introduce the design of dynamic neural radiosity using our proposed neural feature decomposition technique.

4.1 Scene Representation and Trainable Encoding

To describe a dynamic scene with predefined starting and ending states for n animated components, we can use a scalar value $v_i \in [0, 1]$ for encoding and interpolating the states for each. These values together constitute the *explicit scene vector* $\mathbf{v} \in \mathbb{R}^n$. This approach models a bijective mapping between an animation state and a scene vector $\mathbf{v}$. We illustrate the details of the scene vector in Fig. 2.

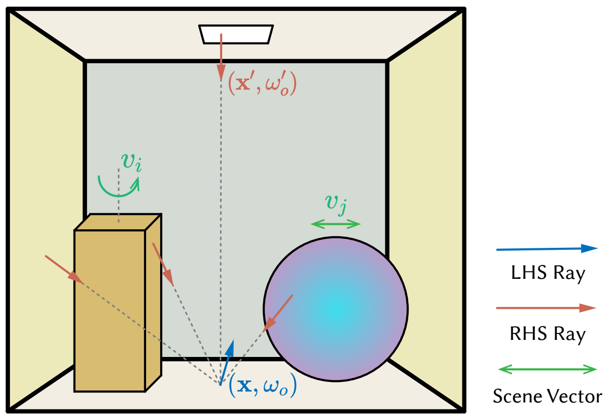

Fig. 2. Illustration of the scene vector and optimization procedures. In the modified Cornell Box scene, we have a rotating box and a translating sphere, where the animation state is parameterized with scalar floats v_i and v_j in $[0, 1]$, together constituting a 2D scene vector $\mathbf{v}$. To train our model, we first sample random vertex $\mathbf{x}$ and outgoing direction ω_o (the LHS ray). The RHS of the rendering equation (incident rays) is then estimated by sampling multiple RHS rays, with each LHS/RHS ray yielding a network query. Finally, an optimization step is performed with the residual (Eq. 10) estimated using the query results for network optimization (Eq. 12).

To model the spatially varying dynamic radiance field with animated components, previous works directly provide the animation state $\mathbf{v}$, concatenated with the 3D spatial coordinate $\mathbf{x}$ and the viewing direction ω_o as inputs [Diolatzis et al. 2022]. Conditioned on them, the network then predicts the radiance $L_\theta(\mathbf{x}, \omega_o, \mathbf{v})$ to approximate the ground truth radiance, where θ denotes the parameters of the network. However, relying only on the network to model the high-frequency details inflicts high demand on both the training data and network capacity, resulting in higher costs for both training and inference (e.g., ≈ 5 fps). To address this, previous NeRF-like applications generally use trainable parametric encoding for network input to model the high-frequency features (discussed in Sec. 2), among which a spatial 3D grid is most commonly used [Müller et al. 2022]. This results in less demand for the network capacity, effectively trading a larger memory cost for less computational overhead.

In this work, we also leverage trainable spatial encoding with a multi-resolution design to model dynamic radiance fields. Specifically, a set of 3D multi-resolution spatial grids consists of L uniform grids G_l, each covering the entire scene bounding volume with a resolution of D_l^3. For the grid G_l of the l-th resolution level, a trainable feature vector $y \in \mathbb{R}^F$ is assigned to each of its lattice points. To get the parametric encoding for 3D inputs $\mathbf{x}$, we trilinearly interpolate the 2^3 features of the voxel surrounding $\mathbf{x}$. Formally:

$$G(\mathbf{x}) = \bigoplus_{l=1}^{L} \text{trilinear}\left(\mathbf{x}, V_l\left[\mathbf{x}\right]\right), \tag{4}$$

where $V_l[\mathbf{x}]$ is the set of features nearby $\mathbf{x}$ within the l-th grid G_l, and $G : \mathbb{R}^3 \to \mathbb{R}^{L\times F}$ is the parametric encoding function.

Spatial feature grids could also be extended to higher dimensionality, which can be used to encode the additional inputs required for an animated radiance field. Specifically, to model a dynamic scene with n animated components, one can use an $(n+3)$−dim feature grid as the trainable encoding for $L_\theta(\mathbf{x}, \omega_o, \mathbf{v})$, where $\mathbf{x} \in \mathcal{X}$ is the 3D spatial coordinate and $\mathbf{v} \in \mathcal{V}$ is the n-dim explicit scene vector.

4.2 Neural Feature Decomposition

While the use of a high-dimensional grid to encode the animated scene is intuitive and straightforward, this results in impractical performance due to the curse of dimensionality. Specifically, to encode a dynamic scene with n animated components, each resolution level of the $(n+3)$-dim grid now has D_l^{n+3} feature vectors, while querying for input yielding interpolation between 2^{n+3} nearby features. This causes roughly $D^n\times$ storage cost and $2^n\times$ computational overhead, which quickly becomes unmanageable as n grows. Although the hash grid maintains physical memory overhead within a fixed limit, exponentially increasing the number of grids can lead to extensive hash collisions, diminishing its effectiveness in modeling dynamic radiance distribution.

To this end, we design a practical parametric encoding for dynamic scenes using multiple decomposed parametric encoding to model the additional animated variables and their spatial correlations, respectively. As illustrated in Fig. 3, we decompose the aforementioned $(n+3)$-dim grid into three distinct components:

- A space-only 3-dim feature grid $f_{\mathcal{X}}(\mathbf{x})$;
- $3n$ 2-dim feature planes combining the scene vector and spatial position: $f_{\mathcal{X}\mathcal{V}}(\mathbf{x}, \mathbf{v})$;
- A lightweight MLP $f_{\mathcal{V}}$ for modeling the correlations between scene variables.

These components can be expressed as follows:

$$f_{\mathcal{X}}(\mathbf{x}) = G(x, y, z) \tag{5}$$

$$f_{\mathcal{X}\mathcal{V}}(\mathbf{x}, \mathbf{v}) = \bigoplus_{i=1}^{V}(P_{i,1}(x, v_i) \oplus P_{i,2}(y, v_i) \oplus P_{i,3}(z, v_i)) \tag{6}$$

$$\begin{aligned} f_{\mathcal{V}}(\mathbf{v}) &= \text{MLP}(\texttt{Freq}(\mathbf{v})) \\ &= \text{MLP}(\ldots, \cos(2^0 v_i), \sin(2^0 v_i), \ldots, \cos(2^L v_i), \sin(2^L v_i), \ldots) \end{aligned} \tag{7}$$

where $\mathbf{x} = (x, y, z)$ represents the spatial coordinates, and $\mathbf{v} = (v_1, \ldots, v_n)$ represents the explicit scene vector. $G(\cdot, \cdot, \cdot)$ denotes the 3D feature grid for modeling spatial-only features, $P(\cdot, \cdot)$ denotes the feature plane for modeling the correlation between the variables and spatial features, while $\texttt{Freq}(\cdot)$ denotes the frequency encoding [Mildenhall et al. 2021] applied to the scene vector $\mathbf{v}$ before feeding into the MLP. The $\bigoplus$ and $\oplus$ denote feature aggregation operations between different grids and resolution levels, respectively. These components are then aggregated to get the final encoding $f(\mathbf{x}, \mathbf{v})$ for $\mathbf{v}$:

$$f(\mathbf{x}, \mathbf{v}) = f_{\mathcal{X}}(\mathbf{x}) \oplus f_{\mathcal{X}\mathcal{V}}(\mathbf{x}, \mathbf{v}) \oplus f_{\mathcal{V}}(\mathbf{v}). \tag{8}$$

Discussion. In this section, we introduce the design of a practical implicit representation $f(\mathbf{x}, \mathbf{v})$ for high-dimensional scenes parameterized by $\mathbf{v}$. Unlike the decomposed planes used in NeRF applications [Cao and Johnson 2023; Shao et al. 2023], we use an MLP to model the $O(n^2)$ pairwise correlations between n scene

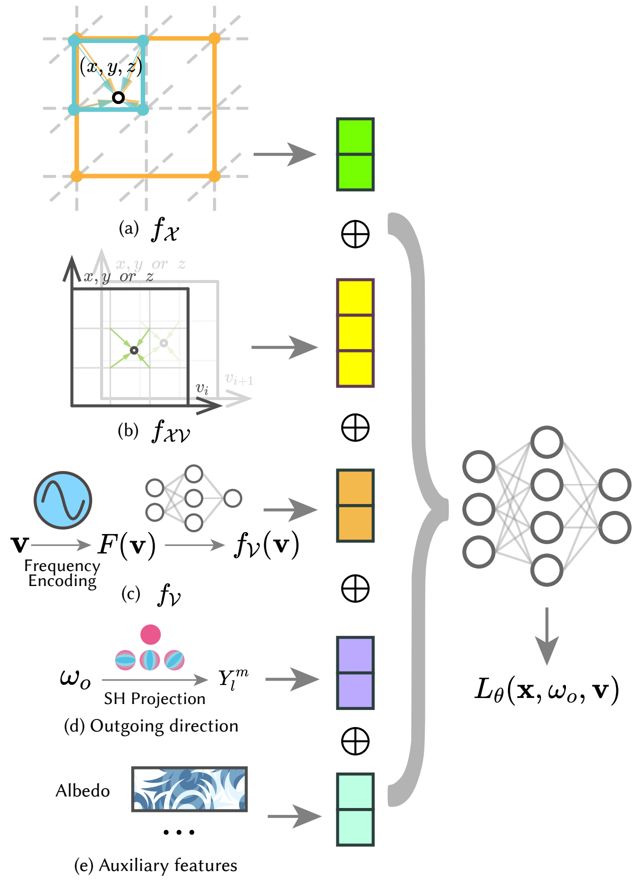

Fig. 3. Illustrating the network architecture of our method. (a) A 3D multi-resolution grid $f_\mathcal{X}$ for modeling animation-independent features. (b) 2D feature planes for modeling the correlation between the spatial features and the animated components ($f_{\mathcal{X}\mathcal{V}}$). (c) A tiny MLP with input frequency encoding, modeling the correlation in-between the animated variables ($f_\mathcal{V}$). (d) Projected coefficients onto the spherical harmonics basis functions of the outgoing direction ω_o. (e) Auxiliary geometry features (surface normal, albedo, etc.). The aforementioned encoded features are concatenated to obtain the final input, which is then fed into the lightweight MLP to predict the reflected radiance $L_\theta(\mathbf{x}, \omega_o, \mathbf{v})$.

variables, thus avoiding quadratic overhead of their methods. The key observation resides in the fact that in our neural rendering setup, most scene variables have low correlations with each other, which is sufficient to model using a single MLP. The scene radiance fields, on the other hand, are still mostly dependent on spatial positions $\mathbf{x}$. In Sec. 6, we show our method more effectively models high-dimensional radiance distribution conditioned on $\mathbf{x}$, $\mathbf{v}$, and ω_o, outperforming previous methods using integrated MLPs, while with a practical computational overhead.

4.3 Optimization

To optimize our proposed method, we minimize the residual loss of the rendering equation over the high-dimensional space $\mathcal{V}$ expanded by the n animated components. For each training step, we uniformly sample the scene space $\mathcal{V}$ to obtain the current scene vector $\mathbf{v}$. For the specific scene animation state defined by $\mathbf{v}$, the residual of the rendering equation can be expressed as:

$$r_\theta(\mathbf{x}, \omega_o, \mathbf{v}) = L_\theta(\mathbf{x}, \omega_o, \mathbf{v}) - L_e(\mathbf{x}, \omega_o, \mathbf{v}) - \int_{\mathcal{H}^2} f(\mathbf{x}, \omega_o, \omega_i, \mathbf{v}) L_\theta(\mathbf{x}'(\mathbf{x}, \omega_i), -\omega_i, \mathbf{v}) \left|(\mathbf{n} \cdot \omega_i)\right| \mathrm{d}\omega_i. \tag{9}$$

For the left-hand side (LHS) of the rendering equation (Eq. 1), we first uniformly sample positions $\mathbf{x}^j$ on the scene surface, as well as the outgoing directions ω_o^j within the upper-hemisphere of that $\mathbf{x}^j$. We then query the network to get the radiance $L_\theta(\mathbf{x}^j, \omega_o^j, \mathbf{v})$. For the right-hand side (RHS) of the rendering equation, we employ Monte Carlo integration to estimate it. Specifically, we sample reflection direction $\omega_i^{j,k}$ and intersect with the scene at $\mathbf{x}'(\mathbf{x}^j, \omega_i^{j,k})$. We query the network to get the radiance $L_\theta(\mathbf{x}'(\mathbf{x}^j, \omega_i^{j,k}), -\omega_i^{j,k}, \mathbf{v})$. Since the distribution in the outgoing direction is uniform, the sampling weight is $p(\omega_i^{j,k}) = \frac{1}{2\pi}$. This process is repeated for M times, and the average value is taken as the result (Fig. 2):

$$r_{\theta,MC}(\mathbf{x}^j, \omega_o^j, \mathbf{v}) = L_\theta(\mathbf{x}^j, \omega_o^j, \mathbf{v}) - L_e(\mathbf{x}^j, \omega_o^j, \mathbf{v}) - \frac{1}{M} \sum_{k=1}^{M} \frac{f(\mathbf{x}^j, \omega_o^j, \omega_i^{j,k}, \mathbf{v}) L_\theta(\mathbf{x}'(\mathbf{x}^j, \omega_i^{j,k}), -\omega_i^{j,k}, \mathbf{v}) \left|(\mathbf{n} \cdot \omega_i^{j,k})\right|}{p(\omega_i^{j,k})}. \tag{10}$$

We define the loss of the network by integrating the square of the residual over the scene space, surface, and hemisphere:

$$\mathcal{L}(\theta) = \|r_\theta(\mathbf{x}, \omega_o, \mathbf{v})\|_2^2 = \int_\mathcal{V} \int_\mathcal{S} \int_{\mathcal{H}^2} r_\theta(\mathbf{x}, \omega_o, \mathbf{v})^2 \, \mathrm{d}\omega_o \, \mathrm{d}\mathbf{x} \, \mathrm{d}\mathbf{v}, \tag{11}$$

where $\mathcal{V}$ indicates the scene space, $\mathcal{S}$ indicates the surface space, $\mathcal{H}^2$ indicates the hemisphere of the point $\mathbf{x}$. This loss could be estimated by Monte Carlo integration (Eq. 10).

The optimal network parameter θ^* is thus

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}(\theta). \tag{12}$$

To help our model better learn the high frequency details in the dynamic radiance field $L(\mathbf{x}, \omega_o, \mathbf{v})$, we empirically propose the following practices:

Additional Inputs. To model $L_\theta(\mathbf{x}, \omega_o, \mathbf{v})$ we also provide the reflection direction of outgoing radiance with respect to the normal $\omega_r = 2\mathbf{n} - \omega_o$, as well as the auxiliary geometry features as network inputs in addition to the trainable encoding $f(\mathbf{x}, \mathbf{v})$. Specifically, we encode ω_r using the spherical harmonic basis [Verbin et al. 2022]. The geometry features consists of the albedo and surface normal of the primary hit point, which is encoded with OneBlob encoding [Müller et al. 2019] and spherical harmonic basis, respectively.

Adaptive RHS Samples. During our experiments, we observed that the number of RHS samples, denoted M, significantly impacts the quality of network convergence. However, a larger M can also lead to a slower training pace. To address this, we adopted an adaptive strategy for incrementing M. Initially set at $M = 32$, we double M in every one-fifth of the total training steps. This approach effectively balances performance and quality.

5 IMPLEMENTATION

System. We implement our algorithm using the Mitsuba3 renderer [Jakob et al. 2022]. The network is implemented using PyTorch [Paszke et al. 2019] and `tiny-cuda-nn` [Müller 2021]. We train our model on an Nvidia RTX 4090 GPU and render images on an Nvidia RTX 3090 GPU.

Renderer Integration. To render dynamic scenes with our proposed method, we first generate geometry and texture information and feed them into the network. Specifically, we first ray-intersect the camera rays to obtain the primary shading point, as well as the view directions ω_o and scene vector $\mathbf{v}$ for network queries. Once the network has computed the approximated radiance L_θ, we copy the shaded pixel data onto the OpenGL frame-buffer for presenting to the display. During this process, all the data remains on the GPU memory via data inter-operation between multiple APIs and without any CPU-GPU data transfer.

Antialiasing. In our method, we use 1-sample-per-pixel network queries to achieve interactive rendering of global illuminated scenes. However, this leads to aliasing issues, particularly noticeable around areas with high-frequency texture and geometry changes. We implement image-space anti-aliasing techniques (FXAA) [DesLauriers 2021] in the pixel shader to alleviate this issue and set the kernel radius to 4 pixels. This significantly reduces the aliasing issue while hardly increasing the performance overhead (< 1ms per frame).

Architecture & Training. In our experiments, for 3D spatial grid $f_{\mathcal{X}}$, we utilize a multi-scale feature grid consisting of 8 resolution levels, with the coarsest resolution set at 32. Each voxel point is assigned with 8 learnable parameters per resolution level. For 2D feature planes $f_{\mathcal{X}\mathcal{V}}$, we reduce the resolution level to 4 and the number of features per point to 2. The lightweight MLP $f_{\mathcal{V}}$ has 4 hidden layers, each containing 128 neurons. Its frequency encoding has $L = 8$ levels. The latent features of each level are then concatenated to obtain the final encoding $f(\mathbf{x}, \mathbf{v})$, then fed to a fully connected network. The architecture of the network includes 4 hidden layers, each containing 256 neurons. For the optimization of network parameters, we employ the Adam optimizer. The initial learning rate was set to 1×10^{-3}. Furthermore, to refine the training process, we implement a decay factor of 0.33 at every one-third interval of the total training duration.

Specular Surfaces. Experimentally, we find that the network struggles to accurately capture the reflected radiance of specular surfaces (e.g. mirrors) with queries from the LHS network. This is because specular BSDFs are Dirac delta lobes with respect to the outgoing direction ω_o, resulting in view-dependent high frequency variations. Following the practices of previous work [Hadadan et al. 2021], we trace secondary rays on specular intersections until a non-specular surface is hit. This effectively models the reflected radiance while avoiding relying on the network for modeling specular distributions. For BSDFs with multiple Dirac delta lobes, tracing multiple rays for each lobe might be a possible solution, which is beyond the scope of this work.

6 RESULTS

6.1 Comparisons

- 7D Dining Room scene has rotational environment lighting, two dynamic ceiling lamps, a wall with dynamic reflectance, as well as three animated objects.
- 5D Living Room scene has blinds for the windows to control the incident light from the environment, as well as a ceiling lamp with varying intensity and height. We also make the table animated along the xy plane.
- 6D Veach Door scene has three animated meshes, an animated door with a changing amplitude of opening, as well as the floor with varying reflectance.
- 6D Bedroom scene has a rotational environment lighting, two walls with varying reflectance, as well as three animated meshes.

We refer readers to our supplemental video for the detailed animation setup in our test scene.

We train our model on a single Nvidia RTX 4090 with 5 $\sim$ 10 hours per scene. Our model generally takes 1 hour to achieve acceptable results. However, a longer training process is needed if higher visual quality is desired, resulting in a trade-off between performance and training overhead. We show our results on scenes with edited viewpoints and animated scene states in Fig. 13.

We compare our method with equal-time path tracing (PT), PT with Monte Carlo denoiser (Oidn) [Áfra 2023], and the state-of-the-art neural rendering method, Active Exploration (AE) [Diolatzis et al. 2022]. For AE, we use a variant that traces additional rays on specular surfaces to enhance quality beyond the original approach. For all the comparisons, we use a single RTX 3090 for rendering and MAPE as the error metric to measure the quality of the reconstruction. As shown in Fig. 8, denoised Monte Carlo (Oidn) tends to suffer from loss of high-frequency details. Moreover, this method fails to account

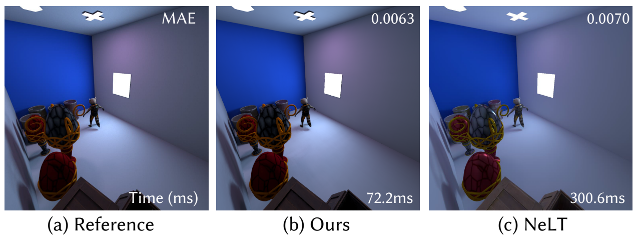

Fig. 4. We compare our method with NeLT [Zheng et al. 2023], and report the inference time cost per frame and mean average error (MAE) to ensure consistency with the metrics used in NeLT. Note that the Nelt-Box scene is reproduced with our setup, which may lead to visual differences in lighting and material. Our method shows superior performance over NeLT, with a close MAE value.

for the temporal coherence, resulting in flickering artifacts under sequential frames that can be inspected (see our supplementary video for details). For AE, the results indicate that while their method achieves moderate quality utilizing image-space techniques, both viewpoint flexibility and synthesis quality are limited by the training data, placing significant demands on the dataset and training process. Our method achieves better visual quality than AE, while reducing training time and obtaining an approximately 3.5× boost of frame rate. This also makes our method practical for integrating interactive and even real-time rendering pipelines.

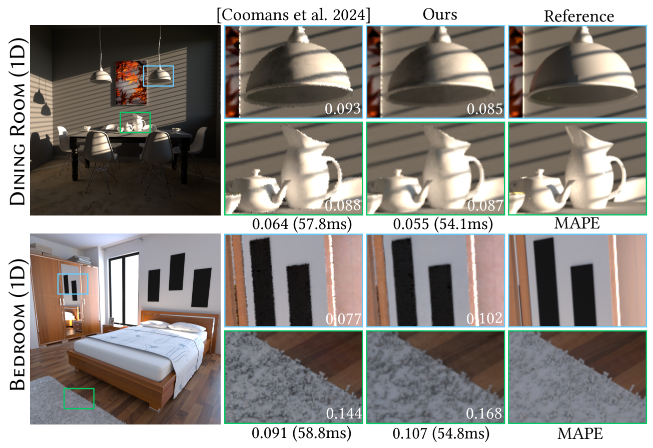

Fig. 5. We evaluate the performance of our method with [Coomans et al. 2024], measuring the inference time cost per frame and MAPE error. This test was conducted on an Nvidia RTX 4090 at a resolution of 1024 × 1024.

We also compare our method with NeLT [Zheng et al. 2023] with the pre-trained model and the rendering framework provided by the authors. We reproduce the NeLT-Box scene in their experiments using Mitsuba. As shown in Fig. 4, both methods model realistic global illumination with dynamic lighting and moving objects, while our method has a 4× higher frame rate than NeLT. However, we should note that NeLT focuses on object editing generalization across scenes, with linear inference cost of the number of editable objects. Moreover, their method requires extensive training budgets (several days on multiple GPUs). Our method, on the other hand, targets precomputed dynamic global illumination with more efficient training and inference performance.

Coomans et al. [2024] extended neural radiosity method to dynamic scenes by using a 4D grid to model only 1D animation. In contrast, our method can model higher-dimensional dynamics. We conduct a comparison with Coomans et al. [2024] on two 1D dynamic scenes (Dining Room and Bedroom with rotational environment lighting). To ensure a fair comparison, we employ a similar network capacity: an MLP with 4 hidden layers and 128 neurons per layer. As shown in Fig. 5, our method achieves quality and efficiency comparable to theirs in 4D scenes. The primary difference lies in the modeling approach: our method uses multiple 2D planes for decomposition, whereas theirs utilizes a 4D hash grid to model the radiance field. This decomposition enables us to reduce memory usage to 118.41 MB, compared to 157.71 MB required by their approach.

6.2 Evaluation

Performance Analysis on Scene Dimensionality. We evaluate the scalability of our method regarding scene dimensionality by conducting performance analysis on a high-dimensional scene with 23 variables. As shown in Fig. 6, our method accurately models the spatio-directional scenes even conditioned on high-dimensional inputs. We also analyze the detailed performance of different components of the network in Tab. 1. Unlike the k-plane method [Fridovich-Keil et al. 2023] that requires quadratic computational overhead regarding dimensionality, our designed representation exhibits linear cost on the number of scene variables. Moreover, we hardly notice any degradation of quality as the dimensionality of the scene increases.

Comparison with Fine-tuned Neural Radiosity. Neural representation is capable of adapting to new data, potentially by fine-tuning with a few training samples of the new domain. To this end, we compare the reconstruction quality of our method with fine-tuned

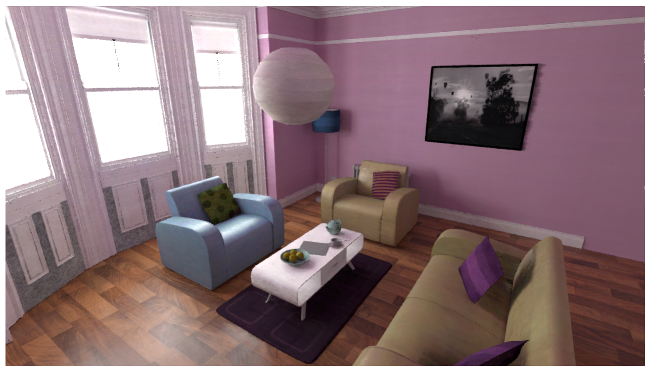

Fig. 6. Rendering of a high-dimensional scene (23D) with 8 animated meshes and 15 dynamic materials. Our method accurately captures the glossy reflections from the incident environment lighting.

Table 1. We conduct performance analysis on a high dimensional scene (23D) to evaluate the effect of different scene dimensionality on computational overhead, as well as the performance breakdown of different components of our network architecture.

dimension	7	12	16	23
$f_{\mathcal{X}} + f_{\mathcal{X}\mathcal{V}}$	12.1ms	23.4ms	32.2ms	45.1ms
	118.9MB	128.9MB	136.9MB	150.8MB
$f_{\mathcal{V}}$	3.6ms	4.9ms	5.8ms	9.1ms
	0.3MB	0.3MB	0.4MB	0.4MB
MLP	19.0ms	21.8ms	24.2ms	28.5ms
	1.2MB	1.3MB	1.4MB	1.5MB
total	34.6ms	50.1ms	62.2ms	82.7ms
	120.4MB	130.5MB	138.6MB	152.8MB
MAPE	0.069	0.086	0.107	0.085

Neural Radiosity (NR) [Hadadan et al. 2021]. Specifically, we slightly perturb the scene state **v**, then visualize the reconstruction error of NR with respect to the number of fine-tuning steps. We train the NR on the static scene for about 2 hours. The fine-tuning process requires approximately 60 optimization steps, taking around 10 seconds to converge to comparable quality to our method (see Fig. 10). This limitation prevents the application of their method to real-time rendering of dynamic scenes.

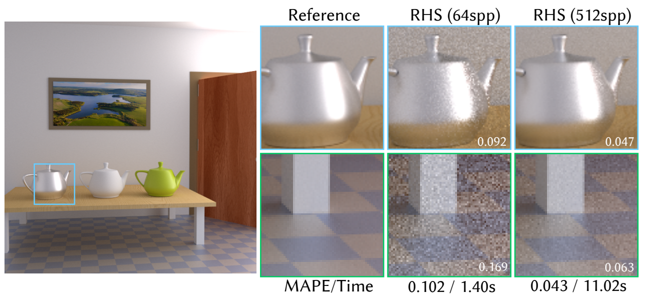

Fig. 7. We visualize the image results by sampling additional RHS rays on the first intersection, with 64spp and 512spp, respectively. Given sufficient time, our method effectively models high frequency details (e.g., glossy reflections) that are indistinguishable from the ground truth images.

RHS Rendering. Given a sufficient time budget, we can also render the image by estimating the right-hand side (RHS) of the rendering equation (Eq. 1) using Monte Carlo integration. As shown in Fig. 7, the image quality increases as the sample count grows, especially for areas with high-frequency details (e.g., glossy reflections).

Network Design Choices. We conduct ablation studies to evaluate the effectiveness of different design choices of our method on the Dining Room scene, as shown in Fig. 12. We first validate the effectiveness of the feature decomposition method by excluding the two types of network components $f_{\mathcal{X}\mathcal{V}}$ and $f_{\mathcal{V}}$, respectively. Both types of the feature plane contribute to the final visual quality, while $f_{\mathcal{X}\mathcal{V}}$ has more influence under a specific animation state. We also note that the adaptive RHS sample count significantly reduces the MAPE error while not increasing training time. Last but not least, we show the FXAA implementation significantly alleviates the aliasing issue with minimal computational overhead.

7 CONCLUSION, LIMITATIONS, AND FUTURE WORK

We present dynamic neural radiosity for rendering high-dimensional dynamic scenes. To achieve this, we first use an explicitly parameterized scene representation to condition the network on the scene animation state. We then develop a practical trainable encoding for the high-dimensional scene representation to model the dynamic radiance distribution. Our method efficiently models the dynamic radiance changes of multi-variate animated scenes, using only moderate additional computation and storage compared to the common feature 3D grids used for static scenes. Moreover, this leaves the network focused on decoding the implicit features into explicit radiance values only, allowing the network to be lightweight for both faster training and inference. Compared to previous work, our method enables better visual quality that can be rendered from arbitrary views, while with an approximately 3.5× higher frame rate.

Despite the effectiveness of our method, there are still some limitations on visual quality. Specifically, sampling and reconstruction issues reside in our current setup due to the fact that we use LHS network queries with 1 sample-per-pixel for rendering. This results in Monte Carlo noise on surfaces with multiple specular lobes (e.g., dielectric materials), as only one lobe is sampled in our current setup (discussed in Sec. 5). To enhance quality, multiple samples per pixel can be used, though this comes at the cost of reduced performance. We demonstrate the improved rendering of dielectric materials by increasing the samples per pixel (spp) in Fig. 11. In addition, it can be straightforward to alleviate this noise issue with auxiliary temporal denoising techniques (e.g., temporal anti-aliasing), which is orthogonal to our work.

In future work, we aim to explore more efficient techniques to accelerate and enhance both the training and inference processes. Specifically, the rendering quality is closely tied to estimating the RHS integral of the rendering equation. Therefore, a more efficient algorithm for estimating the RHS could yield higher quality results. For example, integrating an efficient BDPT renderer capable of handling difficult paths, such as proxy tracing [Su et al. 2024], or a progressive photon mapping algorithm that can synthesize caustic effects with good convergence, such as CPPM [Lin et al. 2020], could greatly benefit the training process and produce more realistic results in detailed illumination. In our pilot study, we observed that the quality of RHS samples significantly impacts the convergence process. However, identifying an optimal RHS estimation setting that improves the quality of neural radiosity-type algorithms still requires further investigation. Another promising avenue could be adaptively allocating the feature space, where high-frequency features that are hard to model occupy a larger portion of the trainable encoding space.

ACKNOWLEDGMENTS

This project was supported by the National Key R&D Program of China (No.2022ZD0160805), NSFC of China (No. 62172013), and the Foundation of Shuanghu laboratory (Grant No.SH-2024JK03). We also thank the following creators of the test scenes: SlykDrako (Bedroom), Wig42 (Dining Room, Living Room), Jaakko Lehtinen (Veach Door), Jay-Artist (White Room), as well as the efforts for converting scene formats by Benedikt Bitterli [2016].

REFERENCES

Attila T. Áfra. 2023. Intel® Open Image Denoise. https://www.openimagedenoise.org.
Hendrik Baatz, Jonathan Granskog, Marios Papas, Fabrice Rousselle, and Jan Novák. 2022. NeRF-Tex: Neural Reflectance Field Textures. In *Computer Graphics Forum*, Vol. 41. Wiley Online Library, 287–301.
Benedikt Bitterli. 2016. Rendering resources. https://benedikt-bitterli.me/resources/.
Ang Cao and Justin Johnson. 2023. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 130–141.
Michael F Cohen, Shenchang Eric Chen, John R Wallace, and Donald P Greenberg. 1988. A progressive refinement approach to fast radiosity image generation. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*. 75–84.
Arno Coomans, Edoardo A Dominci, Christian Döring, Joerg H Mueller, Jozef Hladky, and Markus Steinberger. 2024. Real-time Neural Rendering of Dynamic Light Fields. In *Computer Graphics Forum*, Vol. 43. Wiley Online Library, e15014.
Matt DesLauriers. 2021. *glsl-fxaa* . https://github.com/mattdesl/glsl-fxaa

Stavros Diolatzis, Julien Philip, and George Drettakis. 2022. Active exploration for neural global illumination of variable scenes. *ACM Transactions on Graphics (TOG)* 41, 5 (2022), 1–18.

Honghao Dong, Guoping Wang, and Sheng Li. 2023. Neural Parametric Mixtures for Path Guiding. In *ACM SIGGRAPH 2023 Conference Proceedings*. 1–10.

SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. 2018. Neural scene representation and rendering. *Science* 360, 6394 (2018), 1204–1210.

Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. 2023. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12479–12488.

Cindy M Goral, Kenneth E Torrance, Donald P Greenberg, and Bennett Battaile. 1984. Modeling the interaction of light between diffuse surfaces. *ACM SIGGRAPH computer graphics* 18, 3 (1984), 213–222.

Steven J Gortler, Peter Schröder, Michael F Cohen, and Pat Hanrahan. 1993. Wavelet radiosity. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. 221–230.

Jonathan Granskog, Fabrice Rousselle, Marios Papas, and Jan Novák. 2020. Compositional neural scene representations for shading inference. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 135–1.

Jonathan Granskog, Till N Schnabel, Fabrice Rousselle, and Jan Novák. 2021. Neural scene graph rendering. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–11.

Saeed Hadadan, Shuhong Chen, and Matthias Zwicker. 2021. Neural radiosity. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–11.

David S Immel, Michael F Cohen, and Donald P Greenberg. 1986. A radiosity method for non-diffuse environments. *Acm Siggraph Computer Graphics* 20, 4 (1986), 133–142.

Mustafa Işık, Krishna Mullia, Matthew Fisher, Jonathan Eisenmann, and Michaël Gharbi. 2021. Interactive Monte Carlo denoising using affinity of neural features. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–13.

Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, Merlin Nimier-David, Delio Vicini, Tizian Zeltner, Baptiste Nicolet, Miguel Crespo, Vincent Leroy, and Ziyi Zhang. 2022. *Mitsuba 3 renderer*. https://mitsuba-renderer.org.

James T. Kajiya. 1986. The Rendering Equation. *SIGGRAPH Comput. Graph.* 20, 4 (aug 1986), 143–150. https://doi.org/10.1145/15886.15902

Simon Kallweit, Thomas Müller, Brian Mcwilliams, Markus Gross, and Jan Novák. 2017. Deep scattering: Rendering atmospheric clouds with radiance-predicting neural networks. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 1–11.

Jaakko Lehtinen, Matthias Zwicker, Emmanuel Turquin, Janne Kontkanen, Frédo Durand, François X Sillion, and Timo Aila. 2008. A meshless hierarchical representation for light transport. In *ACM SIGGRAPH 2008 papers*. 1–9.

Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. 2022. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5521–5531.

Zehui Lin, Sheng Li, Xinlu Zeng, Congyi Zhang, Jinzhu Jia, Guoping Wang, and Dinesh Manocha. 2020. CPPM: chi-squared progressive photon mapping. *ACM Trans. Graph.* 39, 6, Article 240 (nov 2020), 12 pages. https://doi.org/10.1145/3414685.3417822

Dani Lischinski, Filippo Tampieri, and Donald P Greenberg. 1992. A discontinuity meshing algorithm for accurate radiosity. *IEEE CG&A* 12, 4 (1992), 10–1109.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)* 41, 4 (2022), 1–15.

Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. 2019. Neural importance sampling. *ACM Transactions on Graphics (ToG)* 38, 5 (2019), 1–19.

Thomas Müller, Fabrice Rousselle, Jan Novák, and Alexander Keller. 2021. Real-time neural radiance caching for path tracing. *arXiv preprint arXiv:2106.12372* (2021).

Thomas Müller. 2021. *tiny-cuda-nn*. https://github.com/NVlabs/tiny-cuda-nn

Oliver Nalbach, Elena Arabadzhiyska, Dushyant Mehta, H-P Seidel, and Tobias Ritschel. 2017. Deep shading: convolutional neural networks for screen space shading. In *Computer graphics forum*, Vol. 36. Wiley Online Library, 65–78.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035.

Peiran Ren, Jiaping Wang, Minmin Gong, Stephen Lin, Xin Tong, and Baining Guo. 2013. Global illumination with radiance regression functions. *ACM Trans. Graph.* 32, 4 (2013), 130–1.

Ruizhi Shao, Zerong Zheng, Hanzhang Tu, Boning Liu, Hongwen Zhang, and Yebin Liu. 2023. Tensor4d: Efficient neural 4d decomposition for high-fidelity dynamic reconstruction and rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16632–16642.

François X Sillion and Claude Puech. 1994. *Radiosity and global illumination*. Vol. 11. Morgan Kaufmann Publishers.

Fujia Su, Bingxuan Li, Qingyang Yin, Yanchen Zhang, and Sheng Li. 2024. Proxy Tracing: Unbiased Reciprocal Estimation for Optimized Sampling in BDPT. *ACM Trans. Graph.* 43, 4, Article 97 (jul 2024), 21 pages. https://doi.org/10.1145/3658216

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. 2020. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems* 33 (2020), 7537–7547.

Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. 2022. Ref-NeRF: Structured View-Dependent Appearance for Neural Radiance Fields. *CVPR* (2022).

Thijs Vogels, Fabrice Rousselle, Brian McWilliams, Gerhard Röthlin, Alex Harvill, David Adler, Mark Meyer, and Jan Novák. 2018. Denoising with kernel prediction and asymmetric loss functions. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–15.

Songyin Wu, Sungye Kim, Zheng Zeng, Deepak Vembar, Sangeeta Jha, Anton Kaplanyan, and Ling-Qi Yan. 2023. ExtraSS: A Framework for Joint Spatial Super Sampling and Frame Extrapolation. In *SIGGRAPH Asia 2023 Conference Papers*. 1–11.

Harold R Zatz. 1993. Galerkin radiosity: A higher order solution method for global illumination. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. 213–220.

Chuankun Zheng, Yuchi Huo, Shaohua Mo, Zhihua Zhong, Zhizhen Wu, Wei Hua, Rui Wang, and Hujun Bao. 2023. NeLT: Object-Oriented Neural Light Transfer. *ACM Trans. Graph.* 42, 5, Article 163 (aug 2023), 16 pages. https://doi.org/10.1145/3596491

Quan Zheng and Matthias Zwicker. 2019. Learning to importance sample in primary sample space. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 169–179.

Zhihua Zhong, Jingsen Zhu, Yuxin Dai, Chuankun Zheng, Guanlin Chen, Yuchi Huo, Hujun Bao, and Rui Wang. 2023. FuseSR: Super Resolution for Real-time Rendering through Efficient Multi-resolution Fusion. In *SIGGRAPH Asia 2023 Conference Papers*. 1–10.

Junqiu Zhu, Yaoyi Bai, Zilin Xu, Steve Bako, Edgar Velázquez-Armendáriz, Lu Wang, Pradeep Sen, Milos Hasan, and Ling-Qi Yan. 2021. Neural complex luminaires: representation and rendering. *ACM Trans. Graph.* 40, 4 (2021), 57–1.

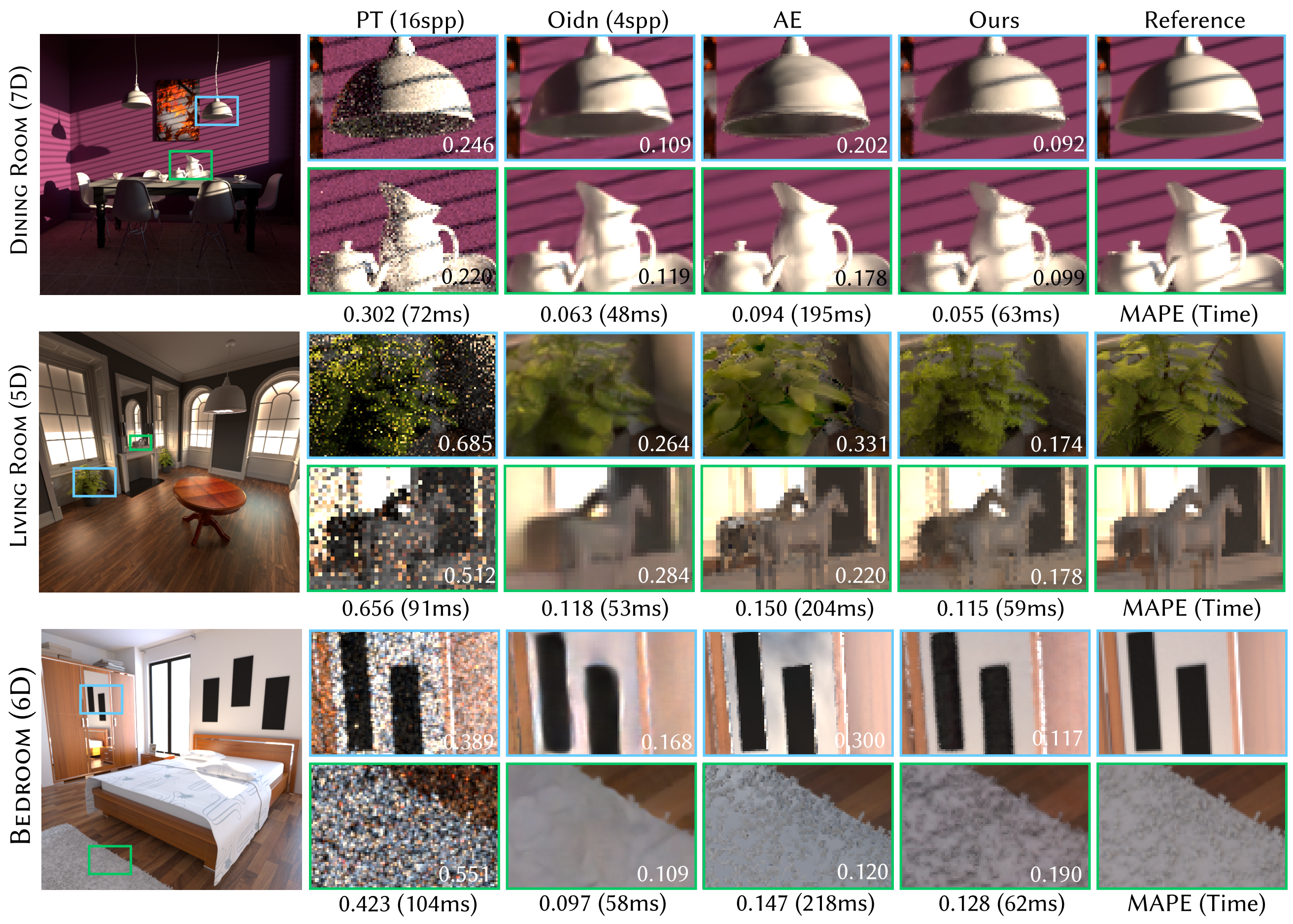

Fig. 8. Visual comparisons on multiple test scenes. We compare the performance of our method to path tracing (PT), denoised PT (Oidn [Áfra 2023]), and Active Exploration (AE) [Diolatzis et al. 2022]. For fair comparisons, we increase the per-pixel sample count (spp) for the path-tracing-based methods to align their frame time costs with ours. For each scene, we report the MAPE error and the average time per frame for each method (the value present in the bottom-right corner indicates the MAPE of the cropped image). Additional visualizations of sequential frames in dynamic scenes and interactive editing of animated components are included in our supplementary video.

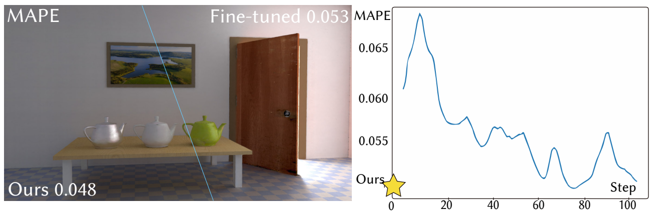

Fig. 10. We compare the reconstruction quality between our method and fine-tuned Neural Radiosity (NR) [Hadadan et al. 2021]. After the scene state changes, NR generally needs about 60 training steps (10s) to converge to our method's quality, making it impractical to adapt NR directly to dynamic scenes.

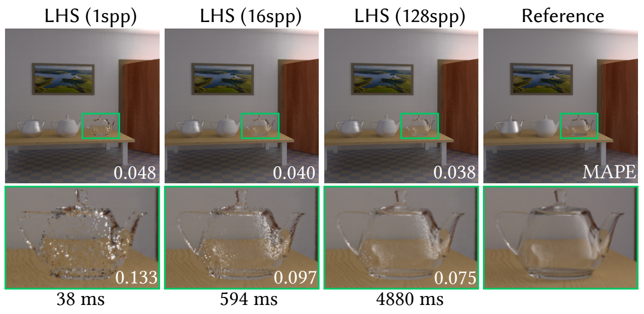

Fig. 11. The image quality can be improved by increasing samples per pixel for LHS, using 1 spp, 16 spp, and 128 spp, respectively. With sufficient rays, our method can accurately produce the correct result for the dielectric material.

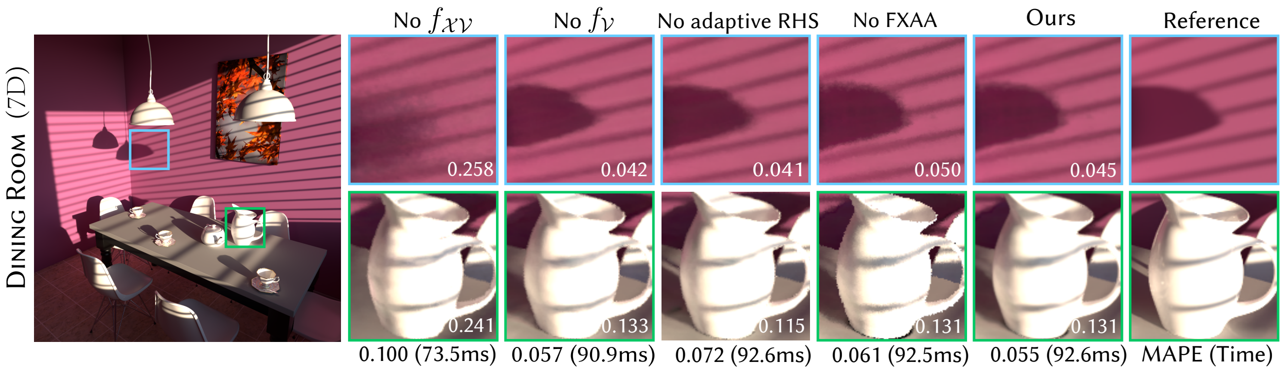

Fig. 12. We validate the effectiveness of different design choices by modifying and comparing the model components to our full method. We show the results of different methods for the DINING ROOM scene. Our full method has better visual quality, especially in the areas with high-frequency details (e.g., the silhouette of the shadows).

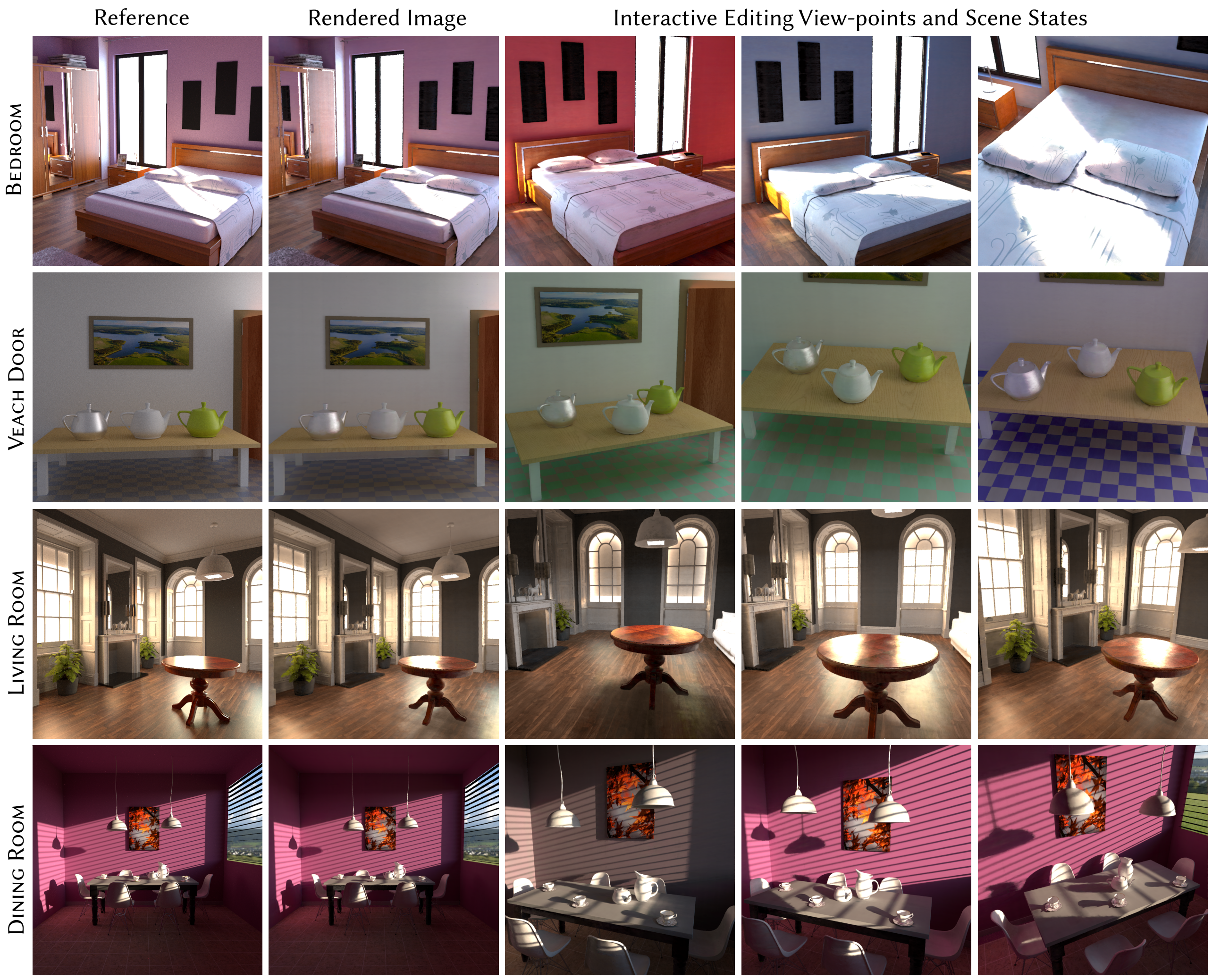

Fig. 13. Interactively edited camera poses and animated scene components on multiple scenes. We demonstrate the manipulation of animated lighting, geometry, and materials. Details of the animated components under sequential frames are included in our supplemental video.