Adaptive Incident Radiance Field Sampling and Reconstruction Using Deep Reinforcement Learning

YUCHI HUO, KAIST, State Key Lab of CAD&CG
RUI WANG, State Key Lab of CAD&CG
RUZAHNG ZHENG, State Key Lab of CAD&CG
HUALIN XU, State Key Lab of CAD&CG
HUJUN BAO, State Key Lab of CAD&CG
SUNG-EUI YOON, KAIST

Fig. 1. Different applications of the proposed radiance field reconstruction algorithm: irradiance caching (left), guiding path for unbiased rendering (middle) and filtering for fast preview (right). Details are further demonstrated in Fig. 12, 14 and 15.

Serious noise affects the rendering of global illumination using Monte Carlo (MC) path tracing when insufficient samples are used. The two common solutions to this problem are filtering noisy inputs to generate smooth but biased results and sampling the MC integrand with a carefully crafted probability distribution function (PDF) to produce unbiased results. Both solutions benefit from an efficient incident radiance field sampling and reconstruction algorithm. This study proposes a method for training quality and reconstruction networks (Q- and R-networks, respectively) with a massive offline dataset for the adaptive sampling and reconstruction of first-bounce incident radiance fields. The convolutional neural network (CNN)-based R-network reconstructs the incident radiance field in a 4D space, whereas the deep reinforcement learning (DRL)-based Q-network predicts and guides the adaptive sampling process. The approach is verified by comparing it with state-of-the-art unbiased path guiding methods and filtering methods. Results demonstrate improvements for unbiased path guiding and competitive performance in biased applications, including filtering and irradiance caching.

CCS Concepts: • **Computing methodologies** → **Neural networks**; *Ray tracing*; *Unsupervised learning*; • **Theory of computation** → **Sequential decision making**;

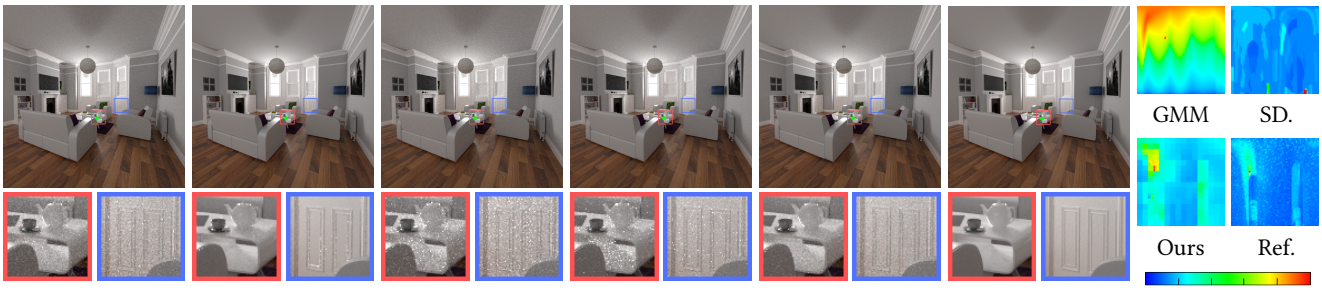
Additional Key Words and Phrases: Incident radiance field; Deep neural network; Adaptive sampling

ACM Reference format:
Yuchi Huo, Rui Wang, Ruzahng Zheng, Hualin Xu, Hujun Bao, and Sung-Eui Yoon. 2019. Adaptive Incident Radiance Field Sampling and Reconstruction Using Deep Reinforcement Learning . *ACM Trans. Graph.* 0, 0, Article 0 (2019), 17 pages.
https://doi.org/0

1 INTRODUCTION

The efficient rendering of realistic images is one of the main challenges in the field of graphics. Realistic rendering is time-consuming because solving the integration of the well-known rendering equation [Kajiya 1986] requires many samples to achieve convergence. Various approaches have been thoroughly studied to handle this problem. In the context of interactive rendering, biased image-space filtering (for a single image) and light-field filtering (for light-field displays) are common solutions. However, biased rendering is not suitable for design, physical simulations, training data generation for deep learning, and high-quality rendering given that any mathematical bias, inconsistency or temporal flicking is unacceptable. The knowledge of the incident radiance field is beneficial for the rendering of high-quality images. For example, unbiased path guiding performs importance sampling for the following rendering equation [Kajiya 1986]:

$$L_o(\boldsymbol{x}, \boldsymbol{\omega'}) = \int_H L_i(\boldsymbol{x}, \boldsymbol{\omega}) f_r(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{\omega'}) c(\boldsymbol{x}, \boldsymbol{\omega}) d\boldsymbol{\omega}, \tag{1}$$



(a) PT rMSE 0.0064 (b) MEMLT rMSE 0.019 (c) GMM rMSE 0.15 (d) SD-tree rMSE 0.15 (e) Ours rMSE 0.0035 (f) Reference (g) Distribution

Fig. 2. *Gloom* scene rendered in one hour for unbiased rendering guided by reconstructed incident radiance fields. There are complex occlusions that cause spiky noises in the rendered images. Our method produces a more visually pleasing and numerically better result than other tested methods by utilizing the information in the 4D radiance field. (g) Pseudo-colors are used to show the reconstructed radiance field at the green dot. Note that GMM and SD-tree **do not** target synthesizing an unbiased radiance field but rather a PDF distribution related to the incident radiance. To compare various methods with the reference radiance field fairly, we visualize their results as pseudo-color images by dividing the maximum value of each method. This allows us to compare the PDF distributions. A similar visualization approach is also used by SD-tree [Müller et al. 2017]. A broader discussion of the compared methods is given in Section 7.

where $\boldsymbol{\omega}'$ is the outgoing direction toward the camera from pixel $\boldsymbol{x}$, $L_i(\boldsymbol{x}, \boldsymbol{\omega})$ is the incoming radiance from direction $\boldsymbol{\omega}$ to $\boldsymbol{x}$, $L_o(\boldsymbol{x}, \boldsymbol{\omega}')$ is the outgoing radiance from direction $\boldsymbol{x}$ to $\boldsymbol{\omega}'$, $f_r(\cdot)$ is the bidirectional reflection distribution function (BRDF) of $\boldsymbol{x}$, $c(\boldsymbol{x}, \boldsymbol{\omega})$ is the clamped cosine term of the incoming direction $\boldsymbol{\omega}$ and the surface normal at $\boldsymbol{x}$, and H is the hemisphere of the shading point to be integrated. In this scenario, the incident radiance $L_i(\boldsymbol{x}, \boldsymbol{\omega})$ can be estimated by a reconstructed radiance field to improve the efficiency of the importance sampling.

In this study, we propose the adaptive sampling and reconstruction of the incident radiance field (radiance field) with deep reinforcement learning (DRL). In contrast to some online reinforcement learning (RL) methods [Müller et al. 2017; Vorba et al. 2014], the proposed approach is the first to facilitate the adaptive sampling by training a massive quantity of data at offline. DRL has realized many achievements, such as the well-known AlphaGo program [Silver et al. 2016] but is still relatively new with regard to rendering, except for a few recent studies [Dahm and Keller 2017a,b,c]. DRL follows the trial-and-error behavior of human beings to learn from an unsupervised environment, and it has been proven to be a powerful tool for solving difficult problems. Instead of fitting online data with artificial models in the complex radiance field space, DRL can learn from massive offline data to explore this high-dimensional space in a purely data-driven manner.

Our approach combines a DRL-based quality network (Q-network) and a new 4D convolutional neural network (CNN)-based reconstruction network (R-network). DRL is used to train the Q-network, which guides the adaptive partitioning and sampling of the radiance field. The R-network filters and predicts the incident radiance field while considering 4D features. We verify the proposed approach using three applications: unbiased path guiding, filtering irradiance caching, and filtering previews. The contributions of this study are as follows.

- Addresses the light-field sampling and reconstruction problem with deep learning techniques and offline datasets.
- Proposes a novel R-network that explores the image and direction spaces of the radiance field to effectively filter and reconstruct the incident radiance field.
- Presents a novel RL-based Q-network to guide the adaptive rendering process.

2 RELATED WORK

In this section, we mainly discuss image and light-field space techniques with adaptive sampling, after which we cover the recent progress in the area of neural network.

Image-Space Methods. Image-space methods have recently attracted considerable attentions. We recommend readers this survey [Zwicker et al. 2015] for more details. Some well-known techniques are based on filtering [Bitterli et al. 2016; Moon et al. 2016; Rousselle et al. 2012, 2013; Schied et al. 2017, 2018; Sen and Darabi 2012]. Inspired by the progress in the area of neural networks, recent studies have led to significant improvements by predicting the filtering bandwidths or kernels for alleviating noise [Bako et al. 2017; Chaitanya et al. 2017; Kalantari et al. 2015].

In general, image-space methods can be readily adapted to various applications with moderate modifications on existing rendering pipelines. Nonetheless, these methods fold the high-dimensional space and do not guarantee mathematical unbiasedness. Accordingly, they are not suitable for light-field rendering or high-precision applications.

Light field Reconstruction Methods. These methods directly work on the light-field space to explore image and direction space information. A classic application reconstructs the integrand by exploring the data coherence in the high-dimensional space [Egan et al. 2009; Hachisuka et al. 2008]. Extra applications include motion blur [Egan et al. 2009], soft shadows [Egan et al. 2011a], directional occlusion [Egan et al. 2011b], and indirect illumination [Lehtinen et al. 2012], among others. Lehtinen et al. [2011] considered visibility discontinuity in reconstructing high-quality diffusion effects. Certain interactive methods sacrifice rendering details for performance [Mehta et al. 2013, 2014; Yan et al. 2015]. Our method utilizes

neural networks to predict the incident radiance field and achieves satisfactory performance and quality for unbiased, high-quality rendering.

Light field Adaptive Sampling Methods. Instead of directly reconstructing the integrand, some prior methods adaptively sample and represent the incident radiance field to synthesize a probability density function (PDF) for unbiased MC estimations. Jensen et al. [1995] proposed guiding camera paths with traced photons and statistically counting the number of photons in a spherical histogram. Following their work, Budge et al. [2008] proposed a specific representation to render caustic effects efficiently. Vorba et al. [2014] proposed an online training method that uses a parametric Gaussian mixture model (GMM) to iteratively estimate the PDF. This method was then improved via adjoint-based Russian roulette and integration of bidirectional scattering distribution function (BSDF) into importance sampling [Herholz et al. 2016; Vorba and Křivánek 2016]. Hua et al. [2015] used virtual point lights instead of estimating the radiance from photons. Recently, Müller et al. [2017] proposed the representation of online data with a spatial-directional hybrid data structure for importance sampling. Online regression was also used for direct illumination sampling [Vévoda et al. 2018].

However, the aforementioned methods are based on the classic online learning of predefined models, whereas our method trains deep neural networks (DNNs) that facilitate the adaptive sampling by training a massive quantity of data at offline. We demonstrate the benefits of our work by comparing our method to the earlier methods of Vorba et al. [2014] and Müller et al. [2017].

Filtering using DNN. In recent years, DNNs have been widely used in a range of applications [Schmidhuber 2015]. Given a dataset, DNNs learn mapping in an end-to-end manner from the input to the outputs with multiple network layers. CNNs achieve impressive improvements in image classification [He et al. 2016], view interpolation [Flynn et al. 2016], and image denoising [Xie et al. 2012; Zhang et al. 2017].

DNNs are also applied to offline rendering. Kallweit et al. [2017] used a deep radiance-predicting neural network to synthesize multi-scattered illumination of clouds. Kalantari et al. [2015] trained a fully connected neural network to predict the optimal bandwidths for cross-bilateral and non-local mean filters. Bako et al. [2017] proposed two types of CNNs to predict illumination or filtering kernels, which were later improved by a modular convolutional architecture for the effective assembly of different rendering engines [Vogels et al. 2018]. A recurrent neural network is also used for the denoising of image sequences [Chaitanya et al. 2017]. Recently, neural-network-based methods have been used in importance sampling [Müller et al. 2018; Zheng and Zwicker 2018]. These techniques belong to the category of image-space filtering. Instead of directly filtering the final rendered image, we focus on radiance field reconstruction for unbiased and high-quality rendering.

DRL. One interesting problem is learning without the ground truth (GT). Müller et al. [2017] described a practical online RL approach for path guiding. However, the approach only fits limited online data with predefined models.

To learn from a massive amount of offline data, deep reinforcement learning (DRL) trains a model and the respective policy for performing a series of actions to achieve the maximum final reward. The agent must learn the outcome of an action through a trial-and-error process progressively to improve the model's ability.

	Description	Type
i	Index of pixel	scalar
j	Index of radiance field block	scalar
$\mathbf{\Gamma}/T$	A set of pixels/Pixel tile	set
θ	A set of radiance field blocks	set
B	Radiance field block	directional bin
$B^j_{\mathbf{x}}$	j-th radiance field block of pixel $\mathbf{x}$	directional bin
B^j_T	j-th radiance field block of pixel $\mathbf{x} \in T$	directional bin

Table 1. Summary of the common notations in the paper.

Apart from online RL, studies on learning ATARI TV and GO games by Google DeepMind have increased interest in end-to-end RL or DRL [Mnih et al. 2015; Silver et al. 2016]. Combining the newest progress in deep learning and the power of offline datasets, DRL achieves attractive results in many domains. Mnih et al. [2015] proposed deep Q-networks (i.e., networks that estimate the reward of actions in given states) with multiple hidden convolutional layers to play ATARI games. One result of this research was the well-known Alpha GO program, which defeated the world champion in the game Go [Silver et al. 2016]. Recently, double Q-learning was used to reduce the overestimation problem of classic Q-networks [Van Hasselt et al. 2016].

Overall, few studies have utilized DRL to solve graphics problems because most graphics problems can readily synthesize GT for supervised learning [Bako et al. 2017; Chaitanya et al. 2017; Kalantari et al. 2015; Kallweit et al. 2017]. With the goal of estimating Monte Carlo (MC) integration, Dahm et al. [2017a; 2017b; 2017c] discovered the similarity between RL and the MC integration process and initially discussed the use of DRL in solving various practical rendering problems. However, these works use online learning and have not been properly compared with other state-of-the-art approaches. In this study, we find that DRL can learn from offline datasets and guide the adaptive sampling and partitioning of the high-dimensional radiance field spaces. The adaptive progress can be regarded as the identification of a sequence of actions, whereas achieving GT action sequences for supervised learning is impractical. To the best of our knowledge, we are the first to use DRL and offline data to train a neural network for the adaptive sampling and reconstruction of (first-bounce) incident radiance fields for unbiased rendering. Another distinction of our method is that it only works on first-bounce shading points given that the multi-bounce shading points are spatially incoherent and difficult to embed into the image space as inputs for common CNNs.

3 OVERVIEW

In this section, we introduce the key ideas of our approach, followed by a pipeline overview.

Representation of Incident Radiance Field. We define the 4D incident radiance field as a combination of the image (i.e., the space of a pixel or a shading point) and direction spaces (i.e., the space of an incident hemisphere centered on the average normal of a group of shading points).

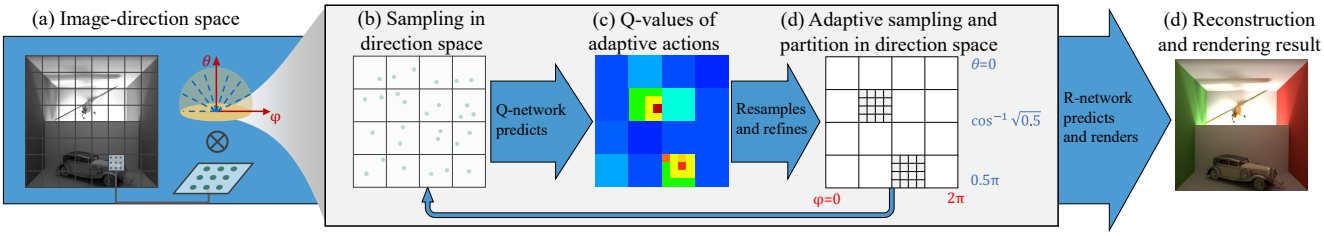


Fig. 3. Runtime overview of our approach. (a) The image space is partitioned into tiles. (b) The direction space that is shared by pixels within each tile is initialized with uniform blocks, and random samples are spread in each block. (c) We use the Q-network to evaluate the quality value of refining (partition the block) or resampling (double the samples in the block) the direction space. (d) The direction space is refined into a hierarchy of the radiance field by taking the best actions. The incident radiance field is reconstructed by using our R-network.

We partition the image space (pixels) into tiles and the direction space into bins referred to as radiance field blocks; note that GPUs can efficiently perform network inference for batched and aligned inputs with single instruction multiple data (SIMD) operations. In the rest of the paper, a pixel tile can also contextually indicate a single pixel without loss of generality. A radiance block B^j is defined by a bounded domain of the direction space and can be expressed as:

$$B^j \triangleq \{\theta, \phi; 0 \le \theta_0^j \le \theta \le \theta_1^j < \frac{\pi}{2}, 0 \le \phi_0^j \le \phi \le \phi_1^j < 2\pi\}, \quad (2)$$

where $[\theta_0^j, \theta_1^j]$ and $[\phi_0^j, \phi_1^j]$ are the elevation and azimuth angle bounds of B^j, respectively. Note that while B^j only defines the bound of the direction space, B^j can sometimes be used to briefly represent $B_{\boldsymbol{x}}^j$, which is the j-th partition in the direction space at the particular pixel $\boldsymbol{x}$, or B_T^j, which is the j-th partition in the direction space shared by all pixels in the tile.

Guided by the Q-network, we adaptively partition the direction space into a radiance field hierarchy with nodes of various sizes by recursively partitioning the azimuth angle θ and the cosine weighted zenith angle θ into half. Fig. 3 presents an example.

We reconstruct an incident radiance field per tile, instead of per pixel, as the inputs of the networks. Therefore, the hierarchy is built in the hemisphere of the local frame of a tile, which is defined from the average normal of pixels in the tile. After the reconstruction of the radiance field on the average local frame, we project the result to the individual frames of the pixels. In this setting, a special case in which the hemisphere of an individual pixel has certain incident directions (i.e., uncovered domains) that are not covered by the hemisphere of the average local frame exists. Reconstructing these singular domains using the networks also leads to poor runtime performance. To address this, we assign a uniform PDF to each uncovered domain for unbiased sampling.

Radiance Field Reconstruction Using the R-network. For a CNN, $\mathcal{N}$, the radiance field reconstruction of an incident radiance block B^j is

$$\hat{L}_{in}^{B^j}(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{X}, \Xi; \boldsymbol{w}), \quad (3)$$

where $\hat{L}_{in}^{B^j}(\boldsymbol{x})$ is the output of the network (i.e., the average incident radiance in B^j at pixel $\boldsymbol{x}$), and $\boldsymbol{X}$ is the incident radiance sample in the domain of B^j; Ξ indicates auxiliary features (e.g., the position, normal and depth) and $\boldsymbol{w}$ is the trainable weight and bias term of $\mathcal{N}$. During training, the network is fed with offline training data to optimize $\boldsymbol{w}$, and thereby minimizes the error between the output and the GT.

Filtering 4D Radiance Space (Section 4). Filtering in 4D space introduces novel challenges. First, the number of samples per radiance block is smaller than the number of samples per pixel because one pixel has more than one block. Second, due to the curse of dimensionality, performing convolutions in a 4D space requires higher memory, training time, and data. To address these problems, we investigate and compare four different reconstruction networks. The results show that our image-direction R-network realizes the best quality given an equal-time comparison.

DRL-based Adaptive Sampling (Section 5). In addition to the reconstruction network, sample distribution and radiance field resolution greatly influence the results. Intuitively, a higher number of samples provides richer information for even using well-trained denoising CNNs [Kalantari et al. 2015]. Adaptively refining the radiance field is a commonly used strategy to preserve lighting details with a limited budget [Müller et al. 2017; Zwicker et al. 2015].

In the same vein, we propose the use of the DRL-based Q-network to guide the sampling and refinement of the radiance field hierarchy. DRL is used to train the network because attempting to cover all the possible radiance field hierarchies and sampling distributions to search for GT are impractical. Specifically, we treat our adaptive sampling as a dynamic process that iteratively takes action to refine radiance field blocks into smaller blocks or to increase the number of samples. The trained Q-network evaluates the value of each action at the runtime to guide the adaptive process.

Pipeline Overview. The entire pipeline includes two main steps: a training process (Figure 4), and a runtime rendering process (Figure 3). The GT data is the average values of the incident radiance field with a directional resolution of 256×256, which is used to train the R- and Q-networks.

During the training process, we initially train the R-network, after which we use the trained R-network to train the Q-network. Rewards are evaluated according to the difference between the GTs and the reconstruction outputs from the R-network to minimize the loss of the Q-network.

When all network training is finished, the trained networks are used for runtime rendering. We partition the image into tiles and use the same radiance field hierarchy for all pixels in a tile. We only

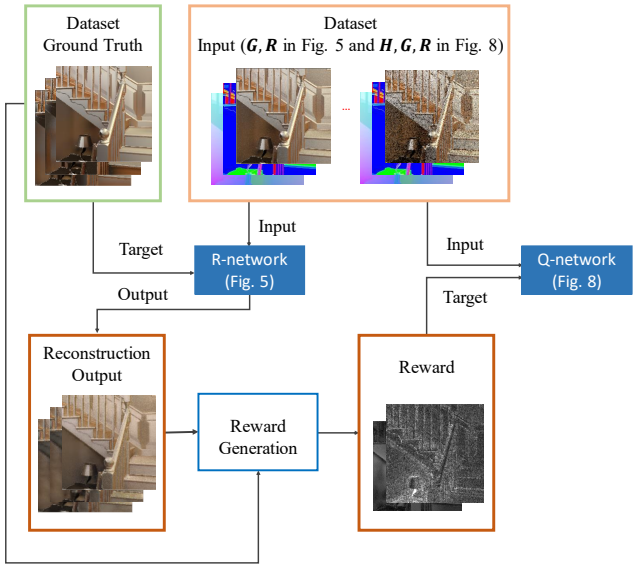


Fig. 4. Overview of the training process. The R-network is initially trained to converge. It then provides the reconstruction outputs, along with the GT and features, to train the Q-network.

use the Q-network iteratively to predict the value of the actions and guide the refinement or sampling of the radiance field hierarchy until a given stopping criterion is reached. Then we use the R-network to reconstruct the leaf nodes of the incident radiance hierarchy (Fig. 3). If the application is filtering, we directly evaluate the product of a BRDF and the reconstructed incident radiance field as a preview. If the desired result is an unbiased image, then we compose the incident radiance field with the BRDF to generate a PDF for the final rendering (guided path tracing).

4 RADIANCE FIELD RECONSTRUCTION NETWORK

In this section, we present the R-network, which is used in the reconstruction of radiance fields. Technically, it has several challenges and involves certain differences compared to image-space filtering. First, samples are dispersed in many directions, and thus inputs are sparse at individual radiance field blocks. Second, direct convolution in the 4D light-field space requires high memory and computation.

To meet these challenges, we explore and compare four different CNNs, namely, image, direction, image-direction, and direction-image networks. As suggested by their names, the image network uses image-space auxiliary features and performs image-space convolution, whereas the direction network works with features and convolution in the direction space. Meanwhile, the image-direction, and direction-image networks connect image- and direction-space convolutions in different orders to perform 4D light field convolutions. In conclusion, the image-direction network, which we use as the final R-network, achieves the lowest error given a fixed time budget (Section 4.3). We mainly discuss image-direction network while leaving the details of other R-networks in the supplementary document.

4.1 Image-Direction Network

Our chosen strategy for the reconstruction of radiance fields is the image-direction network (Fig. 5). The idea is that we initially convolve image-space auxiliary features with incident radiances to learn directional feature maps, after which we rearrange these feature maps into the direction space, finally convolving them to explore coherence in the direction space.

The image-direction network can be partitioned into image and direction parts. We use the notion $\boldsymbol{X}_\Gamma^i$ to represent a **feature map** associated with directional block i and pixel tile Γ. The image part takes some image-space auxiliary feature maps $\boldsymbol{G}_\Gamma$ and radiance feature maps $\boldsymbol{R}_\Gamma^j$ (mean, variance and gradient of the radiance) as inputs. The output is the direction-space feature map $\mathbf{F}_{\mathbf{d}\Gamma}^j$ (Fig. 5). Γ indicates the entire pixel set in the image space, and the superscript j refers to the j-th radiance field block B^j. For example, $\mathbf{F}_{\mathbf{d}\Gamma}^1$ indicates the feature map defined in the image space for the first radiance field block B^1. Instead of relying on hand-designed features, the direction part takes the learned directional feature map $\mathbf{F}_{\mathbf{d}\Gamma}^j$ from the image part as input to simultaneously convolve the radiance predictions of all radiance field blocks.

The initial input of the network includes the image-space auxiliary feature map $\boldsymbol{G}_\Gamma$ and the radiance feature map $\boldsymbol{R}_\Gamma^j$. The image-space auxiliary features are mostly geometrical features, i.e., surface normals (three channels), positions (three channels), and depth (one channel). The radiance features are from the sparse samples, i.e., the average incident radiance $\bar{L}_{in}^{B^j}(\boldsymbol{x})$ of each radiance field block B^j. Similar to kernel-predicting convolutional networks (KPCN) [Bako et al. 2017], we also feed each feature's per pixel variance and image-space gradients to facilitate training and highlight important details. The geometrical features have a total of 26 channels as follows:

- three channels for the average normal, one channel for the average variance in the normals, and six channels for the gradients of the average normal;
- three channels for the average position, one channel for the average variance in the positions, and six channels for the gradients of the average position;
- one channel for the average depth, one channel for the variance in the depth, and two channels for the gradients of the average depth;
- two channels for the gradients of the average radiance of all blocks.

The radiance features have a total of four channels, which comprise:

- three channels for radiance;
- one channel for average variance in the radiance.

These features are all stacked as feature maps in the image space as a tensor to be convolved. The radiance features for a radiance field block are initialized with zeros if there are no samples in the block, which is similar to the features in Bako et al. [2017]'s work.

The first part of the image-direction network contains six standard convolutional hidden layers with 100 kernels of size 5×5. Compared to fully connected layers, using convolutional layers allows a decrease in the number of parameters to speed up the training and avoid overfitting. This structure has been proven to be efficient in image-space filtering problems [Bako et al. 2017]. In

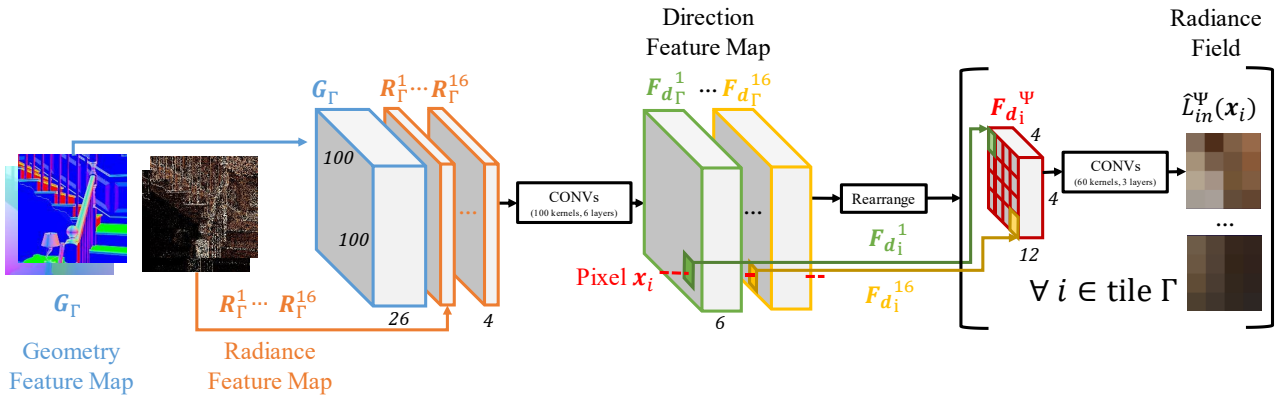


Fig. 5. Illustration of the image-direction network, where subscripts i and Γ respectively represent pixel $\boldsymbol{x}_i$ and the entire pixel set, superscripts j and Ψ respectively represent the radiance field block B^j and the entire set of the radiance field blocks, respectively. $\boldsymbol{G}_\Gamma$ is a feature map of the image-space auxiliary feature; $\boldsymbol{R}_\Gamma^1$, $\boldsymbol{R}_\Gamma^2$, $\cdots$, $\boldsymbol{R}_\Gamma^{16}$ are feature maps of the radiance features of B^1 to B^{16}, and $\mathbf{F_d}$ represents the direction feature map of the corresponding radiance field block, and the final output is the predicted incident radiance. The first part of the network works on the image space, whereas the second part works on the direction space for each separate pixel $\boldsymbol{x}_i$. The rearrangement operation reshapes the outputs of the first part as direction space inputs.

each layer, some trainable kernels are applied to the output of the previous layer to generate linear convolution in nearby pixels. The convolution output is added with a trainable bias and applied to an elementwise nonlinear activation function as the output of the current layer. In this study, a rectified linear unit (ReLU) activation function is used for all networks. The six layers convolve the input tensor to extract 4×4 direction-space feature maps $\mathbf{F}_{\mathbf{d}_\Gamma}^j$ for each radiance field block.

The second part of the network convolves these feature maps to produce the final output. To perform convolution in the direction space, we initially rearrange the output of the image part into the direction space as a $4{\times}4{\times}12$ input tensor representing 12 directional features for 4×4 radiance field blocks; we test various numbers of features and find that using 12 features allows us to balance accuracy and runtime performance.

Note that the rearrangement operation simply rearranges the features so that the entire network is fully differentiable to make end-to-end training possible. The network then separately convolves the input with three standard convolutional hidden layers using 50 kernels of $3{\times}3$ and the final output layer. The output of the network is the predicted or filtered incident radiance $\hat{L}_{in}^{\Psi}(\boldsymbol{x}_i)$ of 16 radiance field blocks at each pixel, where Ψ represents a set of radiance field blocks and $\boldsymbol{x}_i$ represents a pixel. Because each pixel has different radiance field blocks, these radiance field blocks across different pixels can be considered as a 4D radiance field. By treating this output as a PDF for MC estimation, we perform the final unbiased rendering (Section 6).

4.2 Alternative Networks

While the image-direction network works best, we initially test simpler networks: an image network and a direction network. We also test the direction-image network, which is the opposite approach to the image-direction network. In the paper, we briefly introduce these networks and present comparisons in Section 4.3. More details about these networks are available in the supplementary document.

Image network. The image network, which was inspired by a recent image-space filtering and denoising method [Bako et al. 2017], is similar to the first part of our image-space network. Given that this network works only in the image space, we must use the image network to process each radiance field block separately, which requires a high number of parameters and causes a long inference time. In addition, the individual processing of each radiance field using the image network also degrades the results, especially when there are not enough samples, which is mainly because the image network does not consider the information of nearby radiance field blocks. We also test the filtering kernel prediction as in the KPCN technique proposed by Bako et al. [2017] instead of directly predicting the radiance as the output but find that this approach is too expensive when handling tens of radiance field blocks.

Direction network. The direction network only explores data coherence in the direction space. Its network structure is similar to the second part of the image-direction network. Because it only considers directional information, the loss of geometric features leads to blurred geometric boundaries. In addition, the results are not smooth in the image space.

Direction-image network. The direction-image network bears the opposite idea of the image-direction network. In this approach, the direction part first explores directional features and then rearranges intermediate features into the image space, convolving with geometric features to predict the final outputs. Although the direction-image network shows improved quality over the image-direction network given the sample information, it requires much more inference time because it must convolve the image-space features for each individual radiance field block.

4.3 Experiments on Reconstruction Networks

To evaluate these reconstruction networks, we train them using the same training set and then compare them in an equal sample setting. We use 64 samples per pixel, a 4×4 radiance field resolution, and a 500×500 image resolution with the *Staircase* scene outside the training set (Fig. 6). Section 6 contains an additional discussion on the training and testing sets.

First, Fig. 6 shows the reconstruction results of one example of a radiance field block and the means of all blocks. The individual block

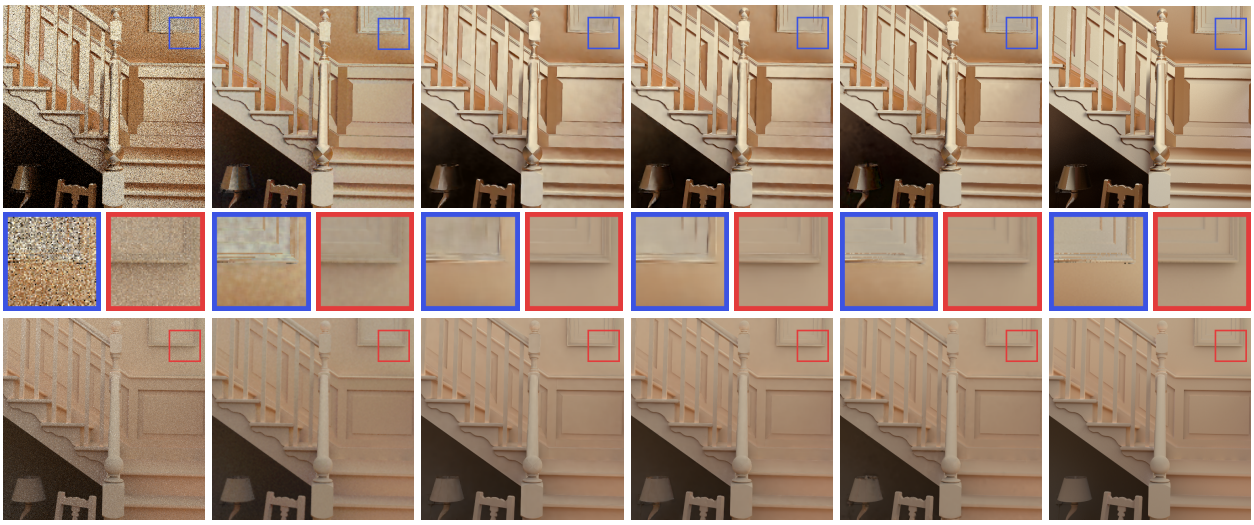


Fig. 6. *Staircase* scene rendered with 64 SPP at a radiance field resolution of 4×4 and image resolution of 500×500. Top row: reconstruction results of one radiance field block, which can be used for path guiding. Bottom row: integration of all radiance field blocks, i.e., the mean of individual blocks, including those shown in the first row, which can be used for ambient occlusion or preview. While the error of the image-direction network is slightly higher error relative to that of the direction-image network, it is more than three times faster. As a result, the image-direction network achieves the best quality given the same time budget (the graph on the right in Fig. 6)

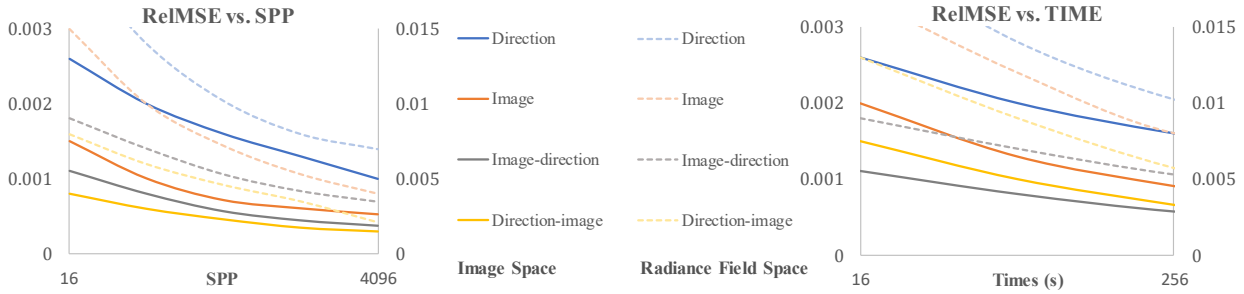


Fig. 7. Left: the rMSE of the reconstruction networks with various numbers of samples per pixel. Right: the rMSE of the reconstruction networks with various times (seconds). The solid lines represent the reconstruction errors when integrating all radiance field blocks into the image space (the third row of Fig. 6), and the dotted lines represent the average reconstruction errors of individual radiance field blocks (the first row of Fig. 6). The image-space errors are important when evaluating the networks when used for filtering previews. The radiance field space errors can be used to evaluate the reconstruction quality of the radiance field, which is critical for unbiased path guiding applications.

result is in the radiance field space and can be used to synthesize a PDF for path guiding. The mean result is the image-space result that can be used for radiance caching or previewing. As noted from the results, the direction network and the image network produce artifacts or overly blurred details because they utilize the information only in neither the image space nor the direction space. However, the image-direction and direction-image networks utilize both spaces and show enhanced results. Nonetheless, the direction-image results are smoother given the same samples per pixel (SPP) because the direction-image network directly outputs results in the image space, whereas the image-direction network outputs results of the radiance field blocks.

Fig. 7 shows the relative mean square error (rMSE) of the final rendering result as we generate more SPP. Here, the number of samples per radiance field block (SPB) is SPP divided by 16 because of the existence of 16 radiance field blocks per pixel. The advantage of combining the image and direction spaces is more pronounced when there are few samples. For the inference speed, the reconstruction times of the direction, image, image-direction, and direction-image networks are 0.75, 3.51, 0.62 and 2.14 s, respectively, at 64 SPP. These results indicate that the direction and image-direction networks are faster than the others. The corresponding rMSEs are 0.002, 0.001, 0.0008, and 0.0006, respectively. The image and direction-image networks require additional time because they separately convolve many image-space auxiliary features with each radiance field block.

However, the image-direction network shows the fastest inference time because it simultaneously convolves 16 radiance field blocks in the image space.

In summary, the direction-image network achieves the best quality given the same inference time relative to the other networks, as indicated in the figure on the right in Fig. 7. Despite the fact that the direction-image network has good reconstruction quality given the same SPP, it requires over three times the inference time. Therefore, we choose the image-direction network as the final reconstruction network. Below we explain the process of the proposed RL-based adaptive sampling method, which progressively improves the overall rendering quality given the chosen image-direction reconstruction network.

5 ADAPTIVE LIGHT-FIELD SAMPLING VIA REINFORCEMENT LEARNING

The R-network in the previous section takes the radiance field hierarchy with a fixed number of nodes, i.e., 4×4 node, as input. However, a strategy of adaptively refining the hierarchy can improve adaption to different scenarios. In this section, we propose the use of a DRL-based Q-network to adaptively guide the sampling and refinement of the radiance field hierarchy.

Two factors are critical when building the hierarchy. The first is the structure of the hierarchy, i.e., the method for discretizing the radiance field (Fig. 3). This factor was noted in a previous adaptive method [Müller et al. 2017]: a higher grid resolution can effectively capture high-frequency lighting features, but it comes with an overhead. The second factor is an adaptive sample distribution, in which more samples (i.e., a greater sample density) are placed in those noisy areas (blocks or nodes) to reduce reconstruction errors (Fig. 7).

In contrast to previous methods [Moon et al. 2016; Müller et al. 2017; Vorba et al. 2014], our approach uses a massive quantity of offline data to train a DNN to guide the process. However, we find that the adaptation of classic supervised learning is challenging because calculating a GT hierarchy and sampling distribution are NP-hard with regard to the solution space [Papadimitriou and Steiglitz 1998]. Thus, we choose to decompose the refining process into a sequence of atomic actions and adapt an unsupervised learning technique, DRL, to train a Q-network to guide the process and minimize reconstruction errors.

5.1 Problem Formulation

Departing from supervised learning, we treat our adaptive sampling and partition approach as a dynamic process and address our problem by DRL. Specifically, we adapt deep Q-learning [Mnih et al. 2015] to choose the next best sampling or partitioning action for a radiance field block with a given state.

Q-learning usually includes an agent situated in a certain environmental state s. Given s, the agent performs action a, which leads to new state s' and reward r. In this framework, the neural network learns from the reward and then predicts the long-term values of actions in different states. In our adaptive sampling scenario, we train a neural network that receives the global radiance field information (e.g., geometry information, radiance samples and radiance field hierarchy) as input states and predicts the quality value (Q-value) of possible actions as the output to determine the next action. First, we define the action, quality value, state and network structure of our approach, as shown below.

We use two different actions for a radiance field block at an epoch. First, we **resample** the block by doubling its sample density per block (the number of samples per block), and second, we **refine** the radiance field block to 4×4 new blocks by equally partitioning each axis as shown in Fig. 3, while keeping the average number of samples per block by adding some new samples. The goal of maintaining the sample density per block is to prevent the degeneration of the reconstruction quality due to the sparser samples. These two actions impact the reconstruction result in different ways. Increasing the number of samples can decrease the variance in the radiance feature, which suppresses noise. Refining can increase the resolution of the grid, which can capture high-frequency details.

The quality value Q and reward of action r are defined in each radiance field block. For radiance field block B^j at a pixel (or a tile, as explained in Section 6.2), the quality value of taking action a in a state s^j is defined by the following equation, also known as the Bellman equation [Kirk 2012]:

$$Q^j(s^j, a) = r(s^j, a) + \gamma \max_{a'} Q^j(s'^j, a'), \tag{4}$$

where s^j and s'^j correspond to the states before and after action a is taken, a' is a possible next action, $r(s^j, a)$ denotes the reward of the action, and γ is a decay parameter between 0 and 1, which results in myopic and far-sighted thinking, respectively. Intuitively, given a state s^j represented by the given radiance field hierarchy and sample distribution, the equation attempts to find the maximum quality value of an action based **not only** on action a, but also with additional steps by recursively considering the next possible action a'. Evaluating this equation with many steps entails significant overhead and poses difficulty in terms of learning the long-term effects of actions. Therefore, in practice, we estimate the Q-value of an action by considering two additional steps, limiting the number of combinations of actions, and thus simplifying our estimation problem. Equation (4) is then approximated as follows:

$$Q^j(s^j, a) \approx r(s^j, a) + \gamma \max_{a'} r(s'^j, a'). \tag{5}$$

We define the reward $r(s^j, a)$ as follows:

$$r(s^j, a) = E^j(s^j) - E^j(s'^j), \tag{6}$$

where $E^j(s^j)$ is the reconstruction error of block B^j, i.e., the integration of the absolute difference between the reconstructed radiance and the ground-truth radiance in the domain of B^j.

Additionally, we found that the scalar range of $r(s^j, a)$ varies with the integration domain (the area of the image space) of B^j, called $dom(j)$. As making the network irrelevant to the scalar range improves its robustness [Bako et al. 2017], we train and infer a normalized substitute reward, $r'(a) = \frac{1}{dom(j)} r(s^j, a)$, and replace the Q-value with $Q'^j(s^j, a) = \frac{1}{dom(j)} Q^j(s^j, a)$. Using the action-value function, we simply use the squared error between $Q'^j(s^j, a)$ and its predicted value, $Q'_{pre}(s^j, a)$, as the loss during the learning process. At runtime, we recover $Q^j(s^j, a)$ for our adaptive sampling process (Section 6.2).

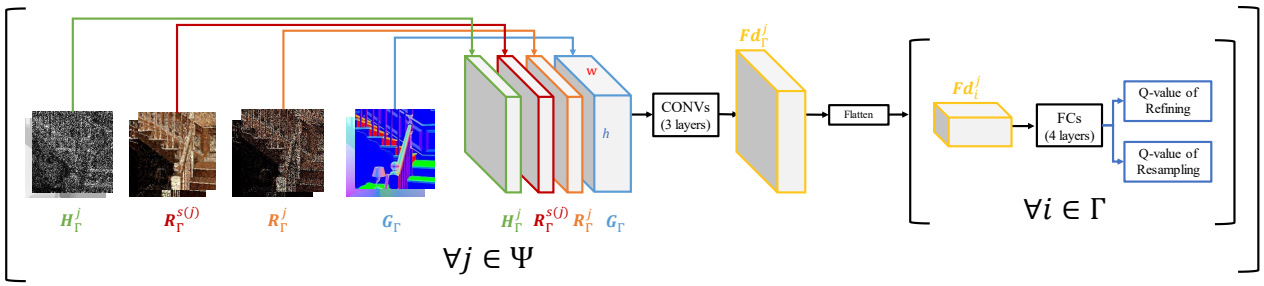


Fig. 8. Illustration of Q-network processing for each radiance field block, which receives various inputs containing image-space auxiliary features (G_Γ), the radiance information of a radiance field block (R_Γ^j) and its parent information ($R_\Gamma^{s(j)}$), and hierarchy features H_Γ^j represented in the image space. Q-network processing then provides the predicted Q-value of two actions.

Algorithm 1 Reinforcement Learning Algorithm

```
function REINFORCEMENT LEARNING(B^j)
    Initialize replay memory D and state s

    while Stopping criterion not reached do
        if RANDOM() < ε then
            a ← RANDOMACTION()
        else
            a ← max_{a'} Q(s', a')
        end if
        r, s' ← PERFORM(a)
        D.STORE(< s, a, r, s' >)
        < ŝ, â, r̂, ŝ' > ← D.DRAWSAMPLE()
        TRAINNETWORK(< ŝ, â, r̂, ŝ' >)
        s ← s'
    end while
end function
```

5.2 Reinforcement Learning Process

Given the definitions of the action, the quality value, and the state, the entire learning algorithm is listed in Algorithm 1. The agent performs an action (e.g., resampling or refining) and observes the state and the reward of a radiance field block B^j as the input to the learning process. First, we initialize replay memory D and state s. The replay memory is used to store a sequence of states, actions, and rewards. On the basis of the replay memory, we can delay the update of the Q-network to learn long-term rewards, instead of only observing short-term effects.

Upon each learning iteration, the algorithm initially calculates the Q-values of the two actions and performs the most valuable action. In increasing the robustness of the algorithm and searching for a wide range, the probability of $\epsilon = 0.01$ exists that the algorithm does not take the most valuable action but picks a random action from the two actions. Once the action is taken, its reward r and the new state s' are recorded in the replay memory D for the delayed updates of the network. At the end of the iteration, a random pair of $< \hat{s}, \hat{a}, \hat{r}, \hat{s}' >$ is selected from the memory to calculate the loss and train the Q-network. With the reply memory, the learning phase is separated from the process of gaining experience, thus helping to decrease the dependency of sequential learning samples and to increase convergence [Silver et al. 2016].

5.3 Q-network Structure

We adopt a widely used network structure in deep learning [Mnih et al. 2015], i.e., we first convolve information and then predict the reward with fully connected layers. As shown in Fig. 8, our Q-network takes information around a pixel i and radiance field block B^j as inputs to predict the rewards of two actions that we can perform on B^j.

Compared with the image-direction R-network, the Q-network only uses image-space convolutions and fewer hidden layers for performance. However, we can use hierarchy information during the adaptive partitioning process to access the information from nearby radiance field blocks. Specifically, two hidden convolutional layers with 50 kernels of 3×3 convolve the features of radiance field block B^j in the image space first, after which the fully connected layers are used to predict the final output. The input includes image-space auxiliary feature map $\boldsymbol{G}_\Gamma$, radiance feature map $\boldsymbol{R}_\Gamma^j$ of radiance field block B^j, newly introduced, hierarchy feature map $\boldsymbol{H}_\Gamma^j$, and parental radiance feature map $\boldsymbol{R}_\Gamma^{s(j)}$, which contains the radiance feature of the parent radiance field block of B^j to capture surrounding direction-space information. These features are all aligned in the image space. The meanings of $\boldsymbol{G}$ and $\boldsymbol{R}$ are identical to those of the R-network (Section 4).

The newly used hierarchy feature describes the current hierarchy state within the range of B^j, i.e., the number of samples and the **maximum** hierarchy level in radiance field block B^j. The hierarchy level represents the granularity of a node; e.g., the root node has level zero, and its corresponding children have level one. The **maximum** hierarchy level of B^j is then defined as the maximum hierarchy level of the current radiance field blocks within the domain of B^j. This feature indicates how nearby pixels refine B^j given the center pixel i. If the neighboring pixels have large **maximum** hierarchy levels within B^j, the central pixel should also look into a deep level of B^j.

5.4 Q-network Experiments

Whereas the R-network uses seven image-space convolutional layers to eliminate noise efficiently, the Q-network uses fewer image-space convolutional layers for two reasons. First, DRL requires more data and time for training, and thus the number of parameters need to be constrained to a reasonable size. Second, we need to consider the performance because the Q-network is frequently evaluated during the adaptive sampling process. We test the training losses of various layers and chose three layers. Readers can refer to the supplementary document for further information.

6 ADAPTIVE SAMPLING AND RENDERING

In this section, we discuss how to train and use our R- and Q-networks in the rendering pipeline. As shown in Fig. 3, the adaptive sampling and the rendering pipeline contain three steps given the trained networks. First, we use the trained Q-network to guide the process of adaptive sampling and refine the field blocks. This results in a hierarchy of radiance field blocks. Second, we use the trained R-network to reconstruct the incoming radiances from the hierarchy. Finally, we apply the reconstruction result for the final rendering.

6.1 Training and Preparation

We first train our R-network given that the Q-network depends on the R-network (Fig. 4). All variables of our networks are initialized using the Xavier method [Glorot and Bengio 2010] and are updated using the Adam optimizer [Kingma and Ba 2014] with a 0.0001 learning rate. The initializer and optimizer are available in TensorFlow [Abadi et al. 2016]. Our dataset includes 20 scenes. To enrich the training dataset, we randomly change the views, lighting, material, and textures of these scenes to generate approximately three hundred scenes. In addition, we randomly select five different values of samples per block (SPB) and radiance field resolutions per scene to adapt the R-network to various ranges. The SPB configurations are selected from $(1, 2, 4, 8, 16)$. The resolution of the radiance field blocks is selected from $(16^1, 16^2, 16^3, 16^4, 16^5)$.

Once the R-network converges, we use it to train the Q-network, i.e., we calculate the quality value (Equation (4)) in the same training data set. Nine examples of the training data set are available in the supplementary report. We train the Q-network through a classic stochastic RL algorithm (Algorithm 1), which can simulate practical data distributions and prevent data coherence through replay memory. However, we find that the stochastic training process has overhead. To accelerate deep reinforcement training, we additionally use a deterministic step to pre-train the Q-network by increasing either the resolution or the sample density with different configurations and then using the stochastic algorithm (Algorithm 1) to fine-tune the pre-trained Q-network. Although the deterministic step can encounter a data coherence problem, we do not observe problems in practice, mainly because of our use of the stochastic process.

6.2 Adaptive Sampling Algorithm

We apply the trained Q-network to guide our adaptive sampling process; the pseudocode is shown in Algorithm 2.

Algorithm 2 Adaptive Sampling Algorithm

```
1:  function AdaptiveSamplingTile(tile T)
2:      Initialize an incident radiance field hierarchy H_T for T
3:      for each B_T^j in H_T do
4:          Q_T^j ← EvaluateQValue(B_T^j)
5:      end for
6:
7:      while stopping criterion not reached do
8:          B_T^j, a_T^j ← SearchMostValueableAction
9:          if a_T^j is refinement then
10:             B_T^{sub(j)} ← Partition(B_T^j)
11:             Q_T^{sub(j)} ← EvaluateQValue(B_T^{sub(j)})
12:             Update(H_T)
13:         else
14:             Resample(B_T^j)
15:             Q_T^j ← EvaluateQValue(B_T^j)
16:         end if
17:     end while
18: end function
```

We partition the image space into 25×25 tiles, each of which is processed in a parallel manner. Note that the Q-network convolves the features of nearby pixels, and thus it is relatively efficient if the input is batched. We combine all pixels of tile T together using the average Q-values of pixels to determine the next action. Therefore, pixels in the same tile share the same hierarchy H_T and can be convolved simultaneously. Note that although pixels in the same tile share the same hierarchy H_T, i.e., the same partitioning on the radiance field, each pixel has different reconstructed incident radiance field values over other pixels in providing a customized reconstruction of each pixel in the tile. In the pseudocode, we use the subscript T to indicate that the blocks or actions are shared by the pixels of tile T.

The quality value for tile Q_T^j is simply the sum of all pixels' quality values: $Q_T^j = \sum_{i \in T} Q_i^j$. , we initialize the hierarchy by uniformly partitioning the radiance field to depth 1 ($4 \times 4 = 16$ radiance field blocks) and uniformly sample in each block with one sample. We then use the Q-network to predict the quality values of the refining or resampling blocks in H_T.

We undertake iterations of our adaptive sampling process as follows. We use a heap associated with actions and their Q-values to select the top (i.e., the most valuable) action a_T^j and perform the action on tile T. If the refining action is chosen, new blocks $B_T^{sub(j)}$ are used to replace B^j with new quality values. If resampling is chosen, only the quality value of B^j is updated after taking the action.

To constrain the training data in a practical range, which is beneficial for bounding the memory and the computation cost, we set a few rules. First, the maximum number of iterations per tile is set to 100 and the maximum depth of the hierarchy is 5, i.e., the smallest block is $\frac{1}{(4\times4)^5} = \frac{1}{2^{20}}$ of the radiance field. Second, the maximum sample density (the samples per block) is 16. Third, when the quality

value of a new action for $\boldsymbol{x}_i$ is less than $\epsilon = 0.02\%$ of the estimated irradiance, we do not distribute any new samples for $\boldsymbol{x}_i$ because the expected effect would be small.

6.3 Reconstruction and Final Rendering

We use the image-direction R-network to reconstruct the radiance field blocks of the hierarchy H_T. To generate a fast preview, we simply use the reconstructed incident radiance field to evaluate and integrate the product of the incident radiance and the BRDF. To render the unbiased image, we treat the reconstructed radiance field and the BRDF as two PDFs to generate the sampling directions. We then combine those two samplers via multiple importance sampling (MIS) [Veach 1998].

Specifically, MIS draws samples from both the BRDF and the reconstructed radiance field as two independent estimators, after which the results of these two estimators are weighted and summed as the final rendering result, which considers BRDF and incident radiance. The BRDF samples can be analytically drawn from a cumulative distribution function. The reconstructed radiance field samples are generated by initially selecting a radiance field block from a discrete PDF and then proportionally sampling a point from the block according to the cosine weighting term. Currently, we focus on reconstructing high-quality incident radiance fields at first-bounce shading points.

As multi-bounce vertices are sparse, they are not compatible with the input format of our networks. For multi-bounce vertices, we can either switch to standard multiple importance sampling, or adapt to other photon guided methods if the lighting is complex. For example, we simply spread one million photons to guide the sample directions [Jensen 1995]. Specifically, the unit square of the hemisphere is partitioned into distinct regions and the photon contributions in each region are accumulated to guide path.

7 RESULTS

We implement our algorithm based on Mitsuba [Jakob 2010], TensorFlow [Abadi et al. 2016], Intel Math Kernel Library and NVidia CUDA. We also compare our method with state-of-the-art path guiding methods, including an online learning method using a Gaussian mixture model (GMM) [Vorba et al. 2014] and an adaptive spatial-directional radiance field hierarchy (SD-tree) [Müller et al. 2017]. The source codes of these methods were implemented and made available by their authors. In addition, we compare path tracing (PT) [Kajiya 1986] and manifold exploration metropolis light transport (MEMLT) [Jakob and Marschner 2012].

GMM uses 30 pre-training passes by default and SD-tree automatically balances the learning and rendering budgets. Our method adaptively refines the incident radiance field, as described in Section 6. First, we compare path tracing guided by our method, the GMM and SD-tree without next event estimation (NEE) techniques, after which we compare our method against all other methods in complex scenes with NEE. The testing hardware includes an Intel Core i7-6700 CPU with 4 cores, 56 GB of memory, and a TITAN V GPU.

7.1 Incident Radiance Field Distribution Reconstruction

In this section, we evaluate different methods in terms of reconstructing the incident radiance field.

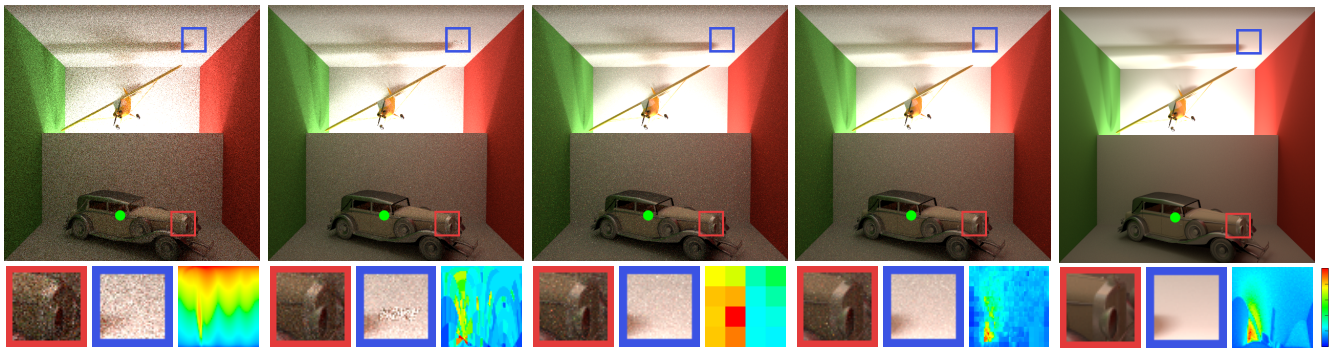
Comparison in terms of guiding distributions. We compare the radiance field reconstruction quality in the *Box* scene (Fig. 9) first. The overall equal-time performance comparison with NEE is presented in the next section. As the sampling strategies of these path guiding methods differ, we exclude the online training factor in this test. Only the final radiance field reconstruction quality is compared, given an equal number of rendering samples per pixel (the rays guided by the reconstructed radiance field to render the final image). Thus, we allow all methods to thoroughly learn the light field and then guide the unbiased path tracing (PT) method with 256 samples per pixel to generate the final images. Specifically, the numbers of reconstruction samples per pixel (the rays used to reconstruct the radiance field) are 508 and 300 for SD-tree and our method, respectively. GMM uses a different sampling strategy that only spreads photons from light sources. The total number of photons is nine million (36 SPP) by default, but it requires more than twice the time for training and fitting relative to the other methods. Another interference factor, NEE, is also turned off for all methods.

To evaluate the improvement of the Q-network, we also render the result only with the R-network, i.e., using a fixed incident radiance field resolution and uniform samples, henceforth referred to as Ours (using only R-net).

Although rendering without NEE presents a disadvantage to our first-bounce reconstruction, our adaptive strategy still shows at least a 30% numerical improvement and has fewer visual artifacts compared with the prior methods (Fig. 9). As demonstrated in Section 7.2, our method realizes a more significant improvement with NEE turned on. To visualize the incident radiance fields of GMM and SD-tree, we uniformly spread 16M samples per pixel to query incident radiances from these methods. Similar to Fig. 2, we visualize the PDF distributions of the green dots for all of these methods.

Box has complex light spots, shadows, and occlusions. Without the help of NEE, all methods show noise. GMM has serious noise but no significant artifacts. SD-tree produces artifacts in the shadow of the airplane wing. In terms of the incident radiance field, SD-tree is good at capturing high-frequency features but is too sensitive to spikes because it directly stores the radiance without filtering.

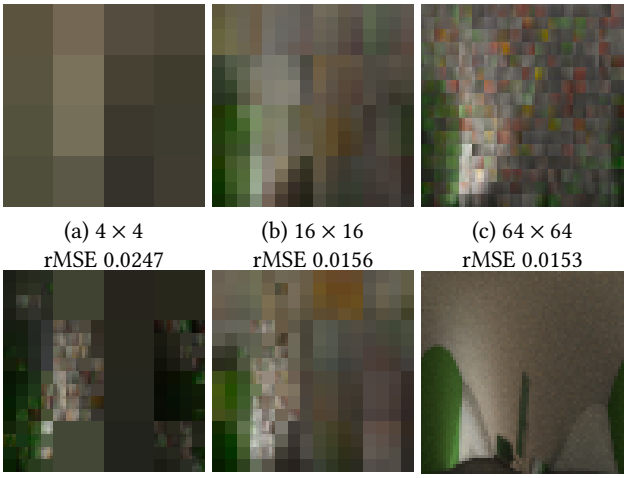
GMM does not trace samples from the camera, but it filters the radiance with the Gaussian model to smoothen out noises. In this scene, sparse eye paths cannot capture the true radiance field. Comparatively, the result of our adaptive method has little noise and no noticeable artifacts. Its reconstructed incident radiance field is also smooth. Overall, our adaptive method using the R- and Q-networks shows the lowest numerical error. As expected, our fixed method using only the R-network is worse than the adaptive method. Surprisingly, although the reconstructed incident radiance field is coarse, it is comparable to those of GMM and SD-tree visually and numerically because: a) a coarse resolution eliminates variance; and b) our method can directly sample the cosine weighted incident radiance from the local frame of an individual pixel to effectively approximate the rendering equation, whereas SD-tree and GMM methods need to share models among nearby pixels to reduce the learning cost.



(a) GMM rMSE 0.0101 | (b) SD-tree rMSE 0.0056 | (c) Ours (only R-net) rMSE 0.0062 | (d) Our Adaptive rMSE 0.0041 | (e) Reference (32k SPP)

Fig. 9. *Box* scene rendered with 256 guided rays per pixel. For this test, NEE is turned off for all methods to emphasize the differences in the guiding distributions, while the other tests use NEE, considerably improving our first-bounce reconstruction method. The resolution is 500×500. Whereas (d) is the result when our adaptive approach using R- and Q-networks, (c) is the result of using only the R-network to reconstruct the incident radiance field with uniform samples. The bottom right figure of each method shows the reconstructed PDF distribution of the reconstructed radiance field at the green point in pseudocolors.

Evaluation of adaptive partitioning. To evaluate the influence of the radiance field resolution and our Q-network based adaptive partitioning strategy further, we use different radiance field partitioning strategies to quantify the radiance field and spread samples, after which we use the R-network to reconstruct the radiance field. The results and mathematical errors are illustrated in Fig. 10. In the figure, (a), (b) and (c) are computed under the same uniform partitioning and sampling strategy with different resolutions. Note that the number of samples is less than the resolution of 64×64 blocks in the case of (c), which results in considerable variance. Fig. 10 (d) uses a greedy strategy similar to that of SD-tree, i.e., partitioning the blocks with the highest intensity. As expected, the greedy strategy may be driven by noisy samples and generate suboptimal partitions. In summary, our adaptive strategy achieves at least a 50% numerical improvement compared with the other strategies. Although (c) has a much higher resolution and more overhead, the result is still not compatible with that of Ours, which uses 631 partitions.



(a) 4×4 rMSE 0.0247 | (b) 16×16 rMSE 0.0156 | (c) 64×64 rMSE 0.0153

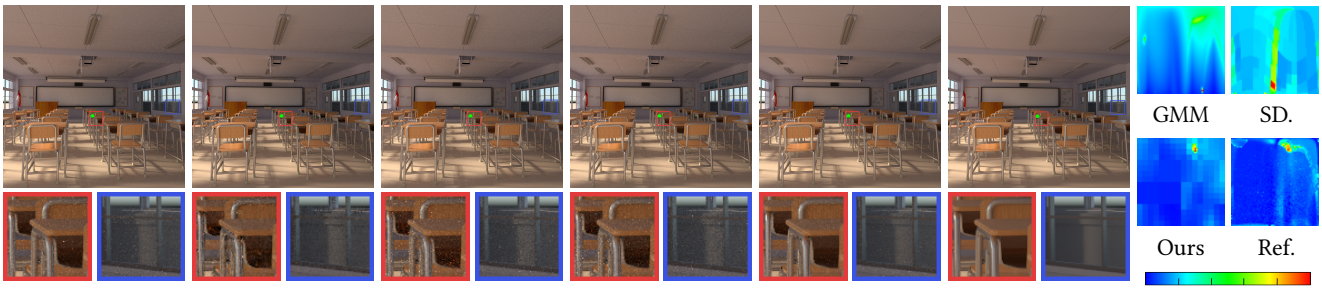
(d) Greedy rMSE 0.0305 | (e) Our Adaptive rMSE 0.00986 | (f) Reference

Fig. 10. Comparison of the **radiance field reconstruction errors** of the green point of the *Box* scene. All figures are reconstructed by the R-network with 768 SPP, but they are computed with different radiance field partitioning strategies. (a), (b) and (c) are uniformly partitioned into different resolutions. (d) uses a simple partitioning strategy that greedily partitions the blocks with the highest intensity. (e) uses our Q-network based adaptive partition strategy with 631 blocks.

7.2 Unbiased Rendering Application

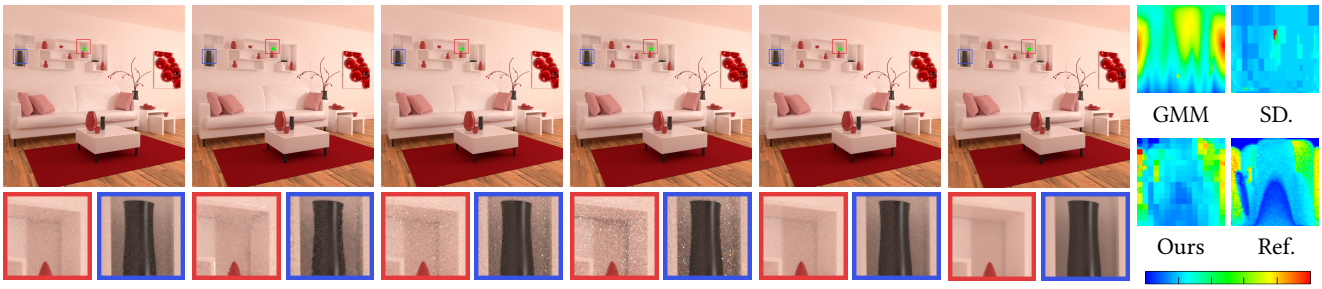
The primary application of the proposed method is path guiding for unbiased rendering. We perform one-hour equal-time comparisons with the PT, GMM, SD-tree and MEMLT methods in various scenes. PT uses 1400 samples per pixel. The adaptive sampling and reconstruction time for our method, the learning time for SD-tree, and the pre-training time for GMM are included. The resolution of these images is 1000×1000.

GMM and SD-tree automatically balance sampling and rendering samples. GMM uses multiple passes (e.g., 30 passes) and distributes 300 K photons and importons within each pass. In the rendering pass, it fits GMM models with reconstruction samples, thus introducing considerable overhead, to guide rendering and acquire final images with 384 rendering samples per pixel (i.e., samples used during the final rendering stage). SD-tree distributes a small number of samples initially and then doubles the number of samples at the next pass with the previous samples used as a guide. In total, there are 508 and 768 reconstruction and rendering samples per pixel, respectively, for SD-tree. In contrast to the multiple pass strategy of GMM and SD-tree, we adopt the relative error estimation strategy (e.g., [Huo et al. 2015]) to control the sampling of samples. The average reconstruction and final rendering SPP are 300 and 1024, respectively. The initial radiance field sampling and reconstruction time is approximately 15 minutes, whereas the neural network inference time is within three minutes. In general, the neural network overhead is significantly smaller than the total computation time of one hour.



(a) PT rMSE 0.0064 (b) MEMLT rMSE 0.0046 (c) GMM rMSE 0.027 (d) SD-tree rMSE 0.0041 (e) Ours rMSE 0.0022 (f) Reference (g) Distribution

Fig. 11. *Classroom* scene rendered in one hour. The pseudocolor figure visualizes the reconstructed PDF distribution of the radiance field at the green dot.



(a) PT rMSE 0.0011 (b) MEMLT rMSE 0.0015 (c) GMM rMSE 0.00072 (d) SD-tree rMSE 0.020 (e) Ours rMSE 0.00035 (f) Reference (g) Distribution

Fig. 12. *Red* scene rendered in one hour. The pseudocolor figure visualizes the reconstructed PDF distribution of the radiance field at the green dot.

The size of each pixel tile is set to 25×25 to balance memory consumption and parallelism between threads. As we can train the R-network for arbitrary tile sizes, this parameter does not influence the reconstruction quality. However, it is the primary parameter that determines the memory consumption of the entire algorithm, because each pixel has its own information of the incident radiance field, while pixels in the tile share the same hierarchy, i.e., the same adaptive partitioning on the field. Overall, each tile takes 1.5 GB of memory.

Classroom (Fig. 11) is an open scene with complex occlusions. Most of the scene is illuminated by indirect illumination. Some parts of the scene, such as the desk drawer, have narrow incident light paths that cause artifacts in most methods. Although there are no obvious artifacts, the result of PT has a higher noise level than most of the others, especially in the ceiling. MEMLT generates smooth results in most areas but it has some artifacts and biases. The other two tested methods produce considerable noise and lose the details of the drawer. Compared with the results of these methods, ours preserves more details and has fewer noises. As seen from the radiance figure, SD-tree and GMM are corrupted by spiky noise, whereas our Q- and R-networks accurately predict the true radiance from noise.

Gloom (Fig. 2) is a highly occluded indoor scene. We seal two windows to increase the complexity of light transport, as adopted by GMM [Vorba et al. 2014]. As a result, this scene is filled with spiky noise for most methods. PT generates considerable noise. Although MEMLT is relatively smooth in the open area, its main problem is spiky artifacts on the boundary of the geometry. GMM and SD-tree methods have similar artifact patterns. The most significant artifact area is under the desk. Although they work well in the open area, they both fail to guide paths within highly occluded and complex geometry because they share the incident radiance field model across nearby pixels that may encounter a resolution problem if the occlusion is complex. By contrast, we use the Q-network to partition the radiance field block adaptively then use the R-network to predict the incident radiance for each pixel. As the neural network considers the visibility information, i.e., incident radiance feature maps $\boldsymbol{R}_\Gamma^1, \boldsymbol{R}_\Gamma^2, \cdots, \boldsymbol{R}_\Gamma^{16}$ shown in Fig. 5, the result is visually satisfactory and contains many occlusions details.

Red (Fig. 12) has a simpler lighting setting but with many subtle geometric features. These details also cause light spots for MEMLT. GMM works better than SD-tree with the help of the Gaussian model. However, both of them have trouble with subtle occlusions deteriorating their estimation quality. In this scene, PT has a decent visual and numerical result, because most parts of this scene are lit by direct light sources, thus, directly sampling the light source with NEE is efficient. In contrast, the SD-tree and GMM methods synthesize incorrect PDFs around occlusions, resulting in serious local artifacts. For general scenarios, i.e., those in which light sources are occluded, and NEE is less efficient, GMM and SD-tree work better than PT [Müller et al. 2017; Vorba et al. 2014].

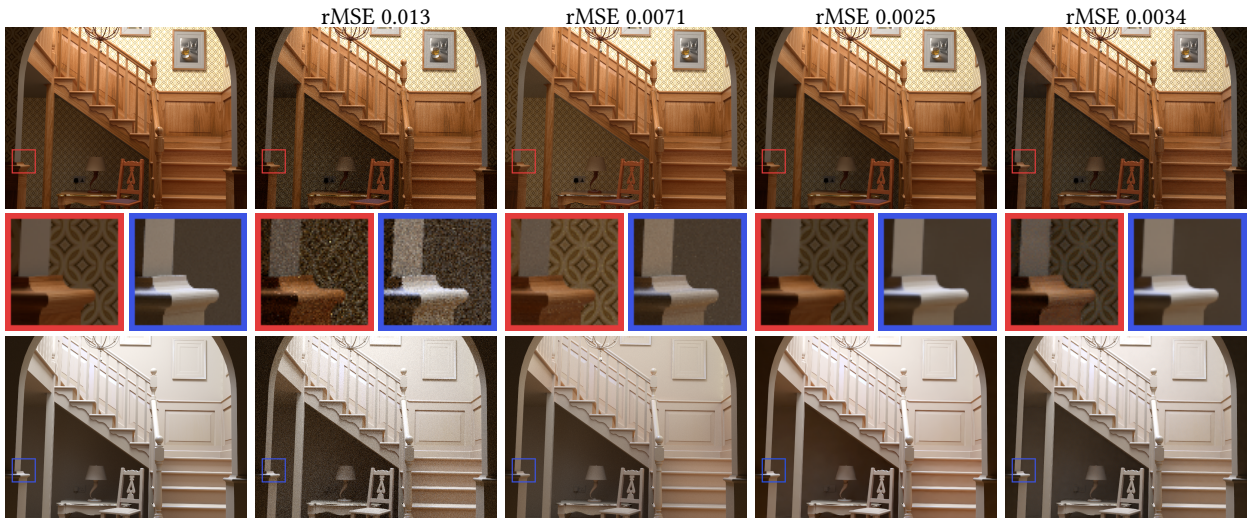


Fig. 13. Biased preview (top row) and irradiance caching (bottom row) tests on *Staircase* scene (30 seconds, 16 SPP and 800×600).

In conclusion, the SD-tree and GMM methods work well in an open area but have problems with subtle occlusions. Our DRL-based path guiding approach shows an advantage in the test scenes, especially under complex environments. Our method can identify the elusive coherence, thus showing strong robustness in most cases.

7.3 Biased Rendering Application

While our approach is mainly designed for unbiased rendering, we also discuss how it behaves for a biased rendering application compared with other techniques. There are many possible applications, such as filtering noise for fast previews, irradiance caching, ambient occlusion, light field rendering and radiance caching. Among all these methods, we test fast previews and irradiance caching to compare the results with state-of-the-art light-field space and image-space methods. However, we should note that the light-field space and image-space methods behave quite differently. Accordingly, this comparison should be considered conditionally.

Filtering for fast preview. Directly filtering the incident radiance field with the BRDF to generate a smooth fast preview is one application of our method. Although many methods reconstruct radiance fields for various reasons, including SD-tree and GMM, few of them can produce smooth, detailed and low-error reconstruction results in the image space. Meanwhile, our method produces a high-quality preview before providing unbiased rendering.

The high-quality reconstruction of incident radiance field indicates the feasibility of integrating the reconstructed radiance field with the BRDF to generate a fast preview. The first row of Fig. 13 shows the preview result, which is rendered with 16 SPP and 800×600 in 30 seconds. As a comparison, we also test a previous light field reconstruction work, REC, targeting filtering light fields and synthesizing image results [Lehtinen et al. 2012], and an image-space CNN-based filtering method, KPCN [Bako et al. 2017], in an equal amount of time. REC reconstructs only indirect illumination, and thus we use another pass to render direct illumination. The timing for rendering direct illumination is not counted. The KPCN network is based on the code released by its author, but it is trained by a dataset synthesized by Mitsuba and our scenes to prevent numerical inconsistency.

As shown in the upper row of Fig. 13, our preview faithfully reflects the geometry, texture and lighting details relative to the previous light field methods. Although our approach achieves better results than REC, the state-of-the-art image-space filtering method, KPCN, achieves better numerical results compared with our radiance field method given the setting of 16 SPP because our radiance field partitioning and reconstruction approaches focus on incident radiance recovery and not considering the BRDF term, which requires an additional criterion on partitioning and reconstruction to adapt to various materials, especially glossy materials. In future works, we will improve the radiance field filtering performance by considering the BRDF.

Filtering for irradiance caching. Another possible biased rendering application generates irradiance caching, in which the BRDF does not need to be considered. We also test REC, KPCN, and our method to calculate the irradiance on every pixel. These results are shown in the bottom row of Fig. 13. Similar to the fast preview results, our method visually and numerically outperforms the previous light-field method, REC, which produces fairly smooth results but contains bias. However, compared with KPCN, our method achieves a slightly better rMSE because the irradiance calculation does not consider the first-bounce BRDF term.

Overall, while it is not primarily designed for biased rendering applications, our method offers a reasonably high quality even for this application compared with state-of-the-art image-space filtering techniques.

7.4 Comparison between Biased and Unbiased Rendering

Biased and unbiased rendering methods usually have different applications. Whereas biased rendering methods, e.g., image-space filtering, can generate very smooth images in a few minutes or even seconds at the expense of accuracy and detail, unbiased rendering methods, e.g., path guiding, mathematically guarantee convergence to the GT result without any bias, and thus can be applied for dataset generation, physical simulations, as well as designs and high-quality rendering tasks.

Fig. 14 shows the result of an equal-time comparison between KPCN and the proposed path guiding method to illustrate the difference between biased and unbiased approaches. The rendering times allocated for the three rows from top to bottom are 1, 2, 8, 30, 60 and 120 minutes. The resolution is 500×500. The SPP values are 32, 128, 700, 2900, 5900, and 11900 for Ours, 40, 140, 800, 3000, 6000 and 12000 for KPCN. As expected, the biased-filtering approach can generate noiseless results within only a few minutes, while the result of the unbiased approach is still heavily corrupted. With approximately half an hour, the numerical errors between the two approaches become similar. The biased result appears to be smoother but contains visual artifacts and loses some geometrical details. By contrast, the unbiased approach gradually suppresses the noise without introducing bias, which results in better geometry results. With more time, the unbiased approach continues to decrease the numerical error, which implies that it gradually approaches the GT. However, the numerical error of the biased approach shows no significant decline and visual artifacts remain in the chess pieces, demonstrating the common features of biased techniques.

7.5 Failure Case

One limitation of the proposed method is that the BRDF term is not considered. On the one hand, focusing on the reconstruction of incident radiance makes the proposed method easily compatible with various applications, e.g., MIS and light field rendering. On the other hand, not considering BRDFs as part of the reconstruction target negatively affects the application of biased filtering for fast previews, especially for glossy materials. Although BRDFs can be integrated into the final image after radiance field reconstruction, a radiance field resolution introduces a frequency limitation on the glossy lobe. As shown in Fig. 15, compared with the unbiased path guiding result, the biased fast preview result fails to capture the highly glossy edges and has a bias when the grazing angle is viewed.

Another limitation is that our method focuses on first-bounce radiance field reconstruction. This idea can be extended to multi-bounce shading points, but it requires a new clustering approach to represent multiple-bounce vertices as a grid for the network input. Rendering transparent or specular objects causes performance problems because they require multi-bounce shading points. Our method performs radiance reconstruction on only one point, e.g., the first nonspecular surface point observed from a mirror. The other points are simply sampled by their own BRDFs. In addition, its performance is limited by the training dataset and it cannot generate smooth results with a small number of SPP, unlike other image-space filtering methods.

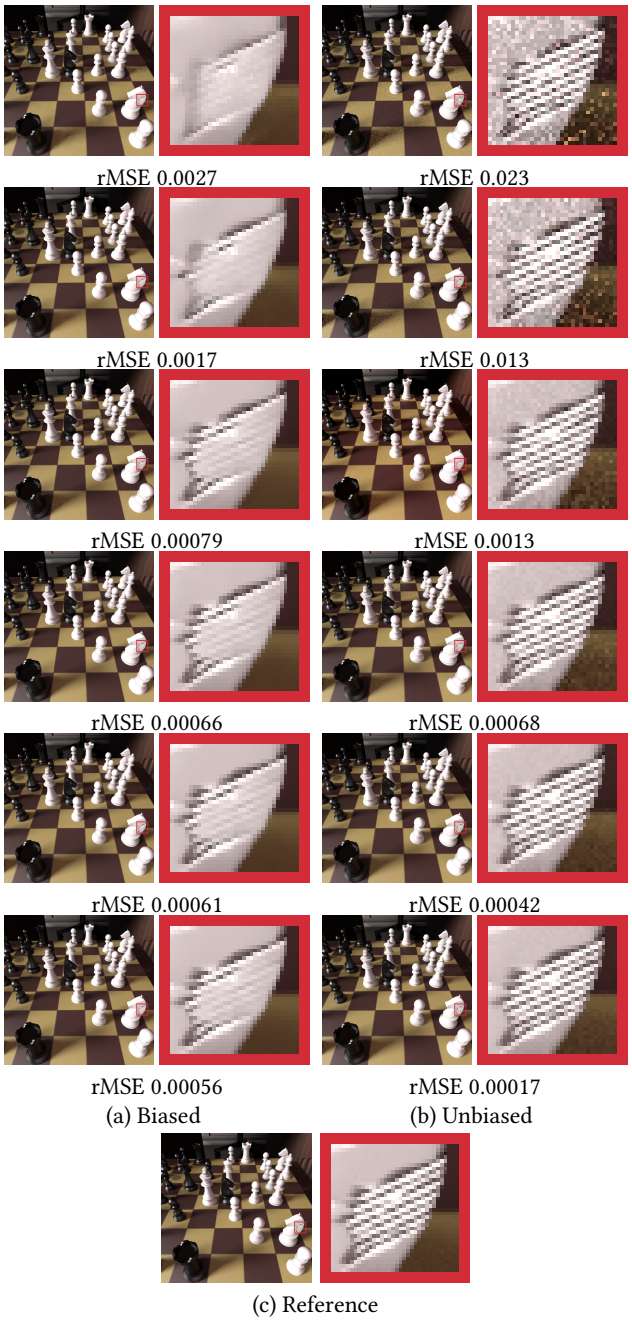
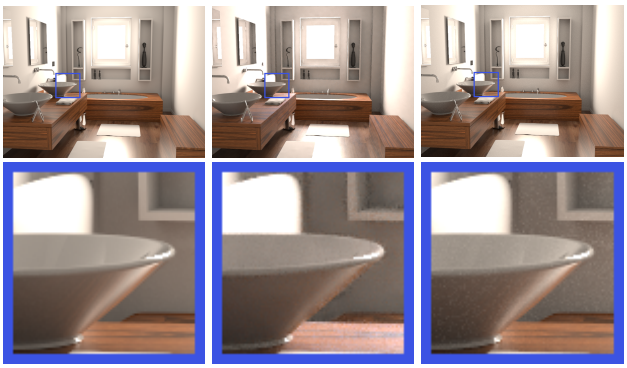


Fig. 14. Equal-time comparisons between biased filtering (KPCN) and unbiased path guiding (Ours) within 1, 2, 8, 30, 60 and 120 minutes from top to bottom.

8 CONCLUSION

In this study, we proposed two novel deep-learning networks, the R-network and the Q-network, for the adaptive sampling and reconstruction of incident radiance fields. We also applied these approaches to various rendering applications and demonstrated their



(a) Reference (b) Preview (c) Path Guiding

Fig. 15. Failure case illustration for the *Bathroom* scene (800 × 600). The preview result fails to capture highlight details. The rMSE values are 0.082 (16 SPP) and 0.0015 (1024 SPP) for the fast preview and unbiased path guiding, respectively.

benefits against state-of-the-art techniques. Our method demonstrated excellent capability for reconstructing the first-bounce incident radiance field.

For our future studies, we will explore deep learning by addressing a significant challenge caused by sparse and irregular points. Finally, we are interested in exploring even higher dimensions, including temporal problems and participating media.

ACKNOWLEDGEMENT

We would like to acknowledge the following projects. Prof. Yoon is supported in part by MSIT/NRF (No. 2019R1A2C3002833) and (NRF-2017M3C4A7066317). Also, Prof. Wang is partially supported by National Key R&D Program of China (No. 2017YFB1002605), NSFC (No. 61872319), Zhejiang Provincial NSFC (No. LR18F020002).

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).

Steve Bako, Thijs Vogels, Brian Mcwilliams, Mark Meyer, Jan NováK, Alex Harvill, Pradeep Sen, Tony Derose, and Fabrice Rousselle. 2017. Kernel-predicting convolutional networks for denoising Monte Carlo renderings. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 97.

Benedikt Bitterli, Fabrice Rousselle, Bochang Moon, José A Iglesias-Guitián, David Adler, Kenny Mitchell, Wojciech Jarosz, and Jan Novák. 2016. Nonlinearly Weighted First-order Regression for Denoising Monte Carlo Renderings. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 107–117.

BC Budge, John C Anderson, and Kenneth I Joy. 2008. Caustic forecasting: Unbiased estimation of caustic lighting for global illumination. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 1963–1970.

Chakravarty R Alla Chaitanya, Anton S Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. 2017. Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 98.

Ken Dahm and Alexander Keller. 2017a. Learning light transport the reinforced way. In *ACM SIGGRAPH 2017 Talks*. ACM, 73.

Ken Dahm and Alexander Keller. 2017b. Learning Light Transport the Reinforced Way. *arXiv preprint arXiv:1701.07403* (2017).

Ken Dahm and Alexander Keller. 2017c. Machine Learning and Integral Equations. *arXiv preprint arXiv:1712.06115* (2017).

Kevin Egan, Frédo Durand, and Ravi Ramamoorthi. 2011a. Practical filtering for efficient ray-traced directional occlusion. In *ACM Transactions on Graphics (TOG)*, Vol. 30. ACM, 180.

Kevin Egan, Florian Hecht, Frédo Durand, and Ravi Ramamoorthi. 2011b. Frequency analysis and sheared filtering for shadow light fields of complex occluders. *ACM Transactions on Graphics (TOG)* 30, 2 (2011), 9.

Kevin Egan, Yu-Ting Tseng, Nicolas Holzschuch, Frédo Durand, and Ravi Ramamoorthi. 2009. Frequency analysis and sheared reconstruction for rendering motion blur. In *ACM Transactions on Graphics (TOG)*, Vol. 28. ACM, 93.

John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. 2016. Deepstereo: Learning to predict new views from the world's imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5515–5524.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 249–256.

Toshiya Hachisuka, Wojciech Jarosz, Richard Peter Weistroffer, Kevin Dale, Greg Humphreys, Matthias Zwicker, and Henrik Wann Jensen. 2008. Multidimensional adaptive sampling and reconstruction for ray tracing. In *ACM Transactions on Graphics (TOG)*, Vol. 27. ACM, 33.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

Sebastian Herholz, Oskar Elek, Jiří Vorba, Hendrik Lensch, and Jaroslav Křivánek. 2016. Product importance sampling for light transport path guiding. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 67–77.

Binh-Son Hua and Kok-Lim Low. 2015. Guided path tracing using clustered virtual point lights. In *SIGGRAPH Asia 2015 Posters*. ACM, 43.

Yuchi Huo, Rui Wang, Shihao Jin, Xinguo Liu, and Hujun Bao. 2015. A matrix sampling-and-recovery approach for many-lights rendering. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 210.

Wenzel Jakob. 2010. Mitsuba renderer. (2010). http://www.mitsuba-renderer.org.

W. Jakob and S. Marschner. 2012. Manifold exploration: a markov chain monte carlo technique for rendering scenes with difficult specular transport.. In *ACM Transactions on Graphics (Proc. SIOGGRAPH)*, Vol. 31. 5:1–5:19.

Henrik Wann Jensen. 1995. Importance driven path tracing using the photon map. In *Rendering TechniquesâĂŹ 95*. Springer, 326–335.

James T. Kajiya. 1986. The rendering equation. In *Proc. SIGGRAPH '86*. 143–150.

Nima Khademi Kalantari, Steve Bako, and Pradeep Sen. 2015. A machine learning approach for filtering Monte Carlo noise. *ACM Trans. Graph.* 34, 4 (2015), 122.

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), 231.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Donald E Kirk. 2012. *Optimal control theory: an introduction*. Courier Corporation.

Jaakko Lehtinen, Timo Aila, Jiawen Chen, Samuli Laine, and Frédo Durand. 2011. Temporal light field reconstruction for rendering distribution effects. *ACM Transactions on Graphics (TOG)* 30, 4 (2011), 55.

Jaakko Lehtinen, Timo Aila, Samuli Laine, and Frédo Durand. 2012. Reconstructing the indirect light field for global illumination. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 51.

Soham Uday Mehta, Brandon Wang, Ravi Ramamoorthi, and Fredo Durand. 2013. Axis-aligned filtering for interactive physically-based diffuse indirect lighting. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 96.

Soham Uday Mehta, JiaXian Yao, Ravi Ramamoorthi, and Fredo Durand. 2014. Factored axis-aligned filtering for rendering multiple distribution effects. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 57.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

Bochang Moon, Steven McDonagh, Kenny Mitchell, and Markus Gross. 2016. Adaptive polynomial rendering. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 40.

Thomas Müller, Markus Gross, and Jan Novák. 2017. Practical Path Guiding for Efficient Light-Transport Simulation. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 91–100.

Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. 2018. Neural Importance Sampling. *CoRR* abs/1808.03856 (2018). arXiv:1808.03856 http://arxiv.org/abs/1808.03856

Christos H Papadimitriou and Kenneth Steiglitz. 1998. *Combinatorial optimization: algorithms and complexity*. Courier Corporation.

Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. 2012. Adaptive rendering with non-local means filtering. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 195.

Fabrice Rousselle, Marco Manzi, and Matthias Zwicker. 2013. Robust denoising using feature and color information. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 121–130.

Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty R Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, Aaron Lefohn, and Marco Salvi. 2017. Spatiotemporal variance-guided filtering: real-time reconstruction for

path-traced global illumination. In *Proceedings of High Performance Graphics*. ACM, 2.

Christoph Schied, Christoph Peters, and Carsten Dachsbacher. 2018. Gradient Estimation for Real-Time Adaptive Temporal Filtering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1, 2 (2018), 24.

Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural networks* 61 (2015), 85–117.

Pradeep Sen and Soheil Darabi. 2012. On filtering the noise from the random parameters in Monte Carlo rendering. *ACM Trans. Graph.* 31, 3 (2012), 18.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484–489.

Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep Reinforcement Learning with Double Q-Learning.. In *AAAI*, Vol. 16. 2094–2100.

Eric Veach. 1998. *Robust monte carlo methods for light transport simulation*. Ph.D. Dissertation. Stanford, CA, USA. Advisor(s) Guibas, Leonidas J. AAI9837162.

Petr Vévoda, Ivo Kondapaneni, and Jaroslav Křivánek. 2018. Bayesian online regression for adaptive direct illumination sampling. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 125.

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), 124.

Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. 2014. On-line learning of parametric mixture models for light transport simulation. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 101.

Jiří Vorba and Jaroslav Křivánek. 2016. Adjoint-driven Russian roulette and splitting in light transport simulation. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 42.

Junyuan Xie, Linli Xu, and Enhong Chen. 2012. Image denoising and inpainting with deep neural networks. In *Advances in neural information processing systems*. 341–349.

Ling-Qi Yan, Soham Uday Mehta, Ravi Ramamoorthi, and Fredo Durand. 2015. Fast 4d sheared filtering for interactive rendering of distribution effects. *ACM Transactions on Graphics (TOG)* 35, 1 (2015), 7.

Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. 2017. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing* 26, 7 (2017), 3142–3155.

Quan Zheng and Matthias Zwicker. 2018. Learning to Importance Sample in Primary Sample Space. *CoRR* abs/1808.07840 (2018). arXiv:1808.07840 http://arxiv.org/abs/1808.07840

Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, Bochang Moon, Ravi Ramamoorthi, Fabrice Rousselle, Pradeep Sen, Cyril Soler, and S-E Yoon. 2015. Recent advances in adaptive sampling and reconstruction for monte carlo rendering. In *Computer graphics forum*, Vol. 34. Wiley Online Library, 667–681.