ReShader: View-Dependent Highlights for Single Image View-Synthesis

AVINASH PALIWAL, Texas A&M University, USA
BRANDON G. NGUYEN, Texas A&M University, USA
ANDRII TSAROV, Leia Inc., USA
NIMA KHADEMI KALANTARI, Texas A&M University, USA

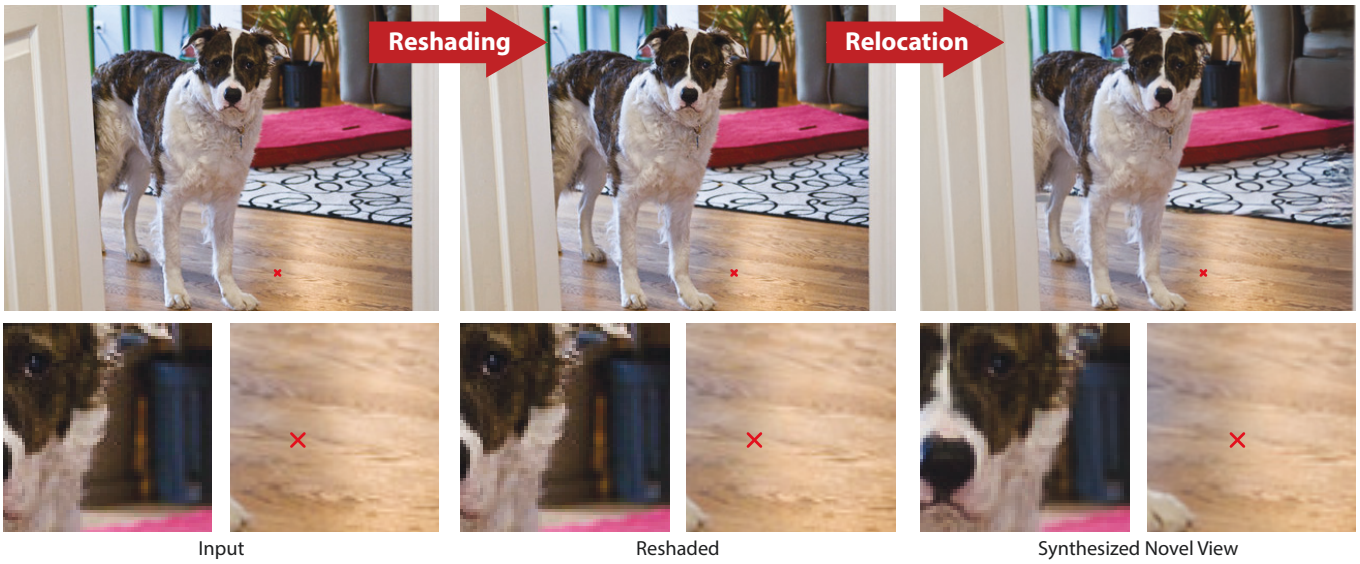

Fig. 1. To properly handle the view-dependent effects, we propose to break down the view synthesis process into two tasks of pixel reshading and relocation. During reshading, we use a neural network to generate a new version of the input image (shown on the left) with the shading computed based on the novel view. As shown on the middle, our reshading network correctly leaves the diffuse areas intact (the dog's head), but moves the highlights on the specular areas (wooden floor). The relocation process takes this reshaded image and generates the novel view image. The red crosses mark the same location on the wooden floor to make it easier to observe the effect of reshading and relocation.

In recent years, novel view synthesis from a single image has seen significant progress thanks to the rapid advancements in 3D scene representation and image inpainting techniques. While the current approaches are able to synthesize geometrically consistent novel views, they often do not handle the view-dependent effects properly. Specifically, the highlights in their synthesized images usually appear to be glued to the surfaces, making the novel views unrealistic. To address this major problem, we make a key observation that the process of synthesizing novel views requires changing the shading of the pixels based on the novel camera, and moving them to appropriate locations. Therefore, we propose to split the view synthesis process into two independent tasks of pixel reshading and relocation. During the reshading process, we take the single image as the input and adjust its shading based on the novel camera. This reshaded image is then used as the input to an existing view synthesis method to relocate the pixels and produce the final novel view image. We propose to use a neural network to perform reshading and generate a large set of synthetic input-reshaded pairs to train our network. We demonstrate that our approach produces plausible novel view images with realistic moving highlights on a variety of real world scenes.

CCS Concepts: • **Computing methodologies** → **Image-based rendering**.

Authors' addresses: Avinash Paliwal, Texas A&M University, College Station, TX, USA, avinashpaliwal@tamu.edu; Brandon G. Nguyen, Texas A&M University, College Station, TX, USA, bgn@tamu.edu; Andrii Tsarov, Leia Inc., Rono-Hills, Menlo Park, CA, USA, andrii.tsarov@leiainc.com; Nima Khademi Kalantari, Texas A&M University, College Station, TX, USA, nimak@tamu.edu.

Additional Key Words and Phrases: View synthesis, neural network, reshading, relocation

1 INTRODUCTION

Creating novel views of a scene from a single image is a compelling way to breathe life into still photographs. When displayed on virtual reality (e.g., HTC vive and Meta Quest) or light field (e.g., Lume Pad [Leia 2023]) devices, these "3D photographs" provide a highly immersive experience for users, allowing them to vividly relive moments captured in still photographs as if they have been transported back in time and place.

The rapid advancements in 3D scene representation and image inpainting techniques have led to remarkable progress in single image view synthesis in recent years. Despite this, the existing techniques focus on producing geometrically consistent novel views and mostly ignore the view-dependent effects. For example, a number of techniques [Jampani et al. 2021; Shih et al. 2020], handle this application in a *modular* manner. These approaches estimate the depth from the input and use it to decompose the scene into multiple layers. These depth layers are then warped to the novel view and composed together to form the final image. Unfortunately, these methods treat the highlights, which are quite common in real scenes, as textures and warp them to the novel views along with other areas. Therefore,

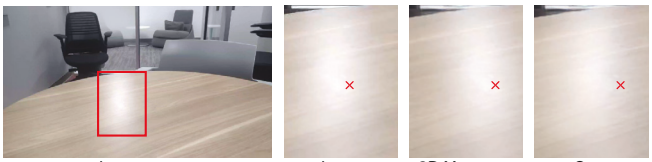

Fig. 2. We compare our results against 3D Moments by Wang et al. [2022]. 3D Moments reconstructs the novel image by moving the input pixels according to their depth values. As such, the highlights are treated as textures and appear to be glued to the wooden table. Our approach, however, is able to properly move the highlights over the table. The red crosses mark the same location on the table. Note that the cross is inside the highlight in the input and 3D Moment's results, but it appears to be outside the highlight in our results.

as shown in Fig. 2, the highlights in their synthesized views appear to be glued to the surfaces, making their results unrealistic.

On the other hand, several approaches [Li and Kalantari 2020; Srinivasan et al. 2017; Yu et al. 2021] handle this problem by learning the process in an *end-to-end* manner. These techniques learn the entire view synthesis pipeline either directly [Srinivasan et al. 2017], or through various scene representations, such as neural radiance field (NeRF) [Yu et al. 2021] and multiplane images (MPI) [Li and Kalantari 2020; Tucker and Snavely 2020]. Although they could potentially handle the view-dependent effects, these techniques often struggle to properly reconstruct the moving highlights.

Our main observation is that both the shading and projected pixel location of a 3D surface point change between the input and novel view images. Modular approaches overlook the view-dependent shading, focusing solely on pixel relocation. The end-to-end approaches, on the other hand, aim to learn to move the pixels and change their shading within a unified system. However, the majority of effort is dedicated to learning pixel relocation, as the contribution of the shading mismatch to their training loss is often minimal.

Guided by this observation, we make a key contribution to break down the novel view synthesis process into two tasks: pixel reshading and relocation (see Fig. 1). During the reshading process, we only adjust the shading of the input image according to the novel camera. We then perform pixel relocation on the reshaded image, using the modular method by Wang et al. [2022], to obtain the final novel view image. We propose to learn the reshading process using a neural network that takes a single image as well as the relative novel camera position as the input and produces the reshaded image. Since there are no publicly available datasets of input-reshaded image pairs, we render a large number of synthetic image pairs for training. We train our reshading network on this newly introduced dataset using a perceptual loss to ensure producing plausible, but detailed reshaded images. We demonstrate that our method produces high-quality novel view images with plausible moving highlights on a wide range of real scenes.

2 RELATED WORK

The problem of view synthesis has been extensively studied and many powerful multi- and single-image methods have been developed [Mildenhall et al. 2020; Shih et al. 2020; Tucker and Snavely 2020; Wizadwongsa et al. 2021]. A complete literature review is beyond the scope of this paper. Here, we mainly focus on approaches that use a single image as the input. We also discuss image relighting methods as they are relevant to the focus of our paper.

2.1 Single Image View Synthesis

We discuss these approaches by categorizing them into two classes of modular and end-to-end. The modular methods [Jampani et al. 2021; Kopf et al. 2019, 2020; Niklaus et al. 2019; Shih et al. 2020; Wang et al. 2022] break down the process into multiple components and address each component separately. Specifically, these techniques divide the view synthesis pipeline into depth estimation, image warping, and image inpainting. The individual methods differ in how they handle each stage of the pipeline. For example, Niklaus et al. [2019] train a depth estimation network and use it to directly reproject the input image to the novel view. On the other hand, Shih et al. [2020] obtain the depth using an existing method [Ranftl et al. 2022] and reconstructs layered depth image (LDI) representation [Shade et al. 1998] to warp the input image to the novel view. These techniques, however, primarily focus on pixel relocation and overlook the pixel reshading process. As a result, they produce results with incorrect view-dependent effects, where the highlights appear to be glued to the surfaces (see Fig. 2).

A category of modular methods focus on handling the view-dependent effects by first decomposing the image(s) into multiple layers (e.g., diffuse and reflective), warping each layer separately, and blending them to generate the final image. However, most of these techniques are either specifically designed for rendering [Lochmann et al. 2014; Zimmer et al. 2015] where ground truth scene information (e.g., geometry and material) is available, or require multiple images [Blake 1985; Roth and Black 2006; Sinha et al. 2012].

In contrast to the modular approaches, a number of techniques [Han et al. 2022; Li and Kalantari 2020; Srinivasan et al. 2017; Tucker and Snavely 2020; Wiles et al. 2020; Yu et al. 2021] attempt to learn the entire view synthesis process in an end-to-end manner. Zhou et al. [2016] propose to estimate optical flows at novel views and use the estimated flow to backward warp the input image. The flow estimation network is trained by minimizing the loss between the synthesized and ground truth novel view images. Srinivasan et al. [2017] propose to estimate a light field from a single image using a convolutional neural network (CNN). Several approaches use a network to estimate intermediate representations, such as point cloud [Wiles et al. 2020], multiplane images (MPI) [Han et al. 2022; Li and Kalantari 2020; Tucker and Snavely 2020], and neural radiance field (NeRF) [Yu et al. 2021]. Since these approaches perform end-to-end training, they could potentially learn to handle the view-dependent effects. However, highlights are usually concentrated in small regions, and thus the shading mismatch does not significantly contribute to the loss function. As such, these methods often are not able to produce results with proper moving highlights.

Recently, several approaches [Chan et al. 2023; Fridman et al. 2023; Gu et al. 2023; Poole et al. 2022; Shue et al. 2022; Watson et al. 2022] have proposed to address this problem using diffusion models [Ho et al. 2020]. Some of these techniques [Shue et al. 2022; Watson et al. 2022] produce novel view images of only single objects or simple scenes. Others [Chan et al. 2023; Fridman et al. 2023] handle complex scenes and produce impressive walkthroughs from a single

image. However, when synthesizing views that are relatively close to the input, the quality of their synthesized images are not on par with the existing modular or MPI-based techniques.

2.2 Image Relighting

Image relighting is the process of reconstructing images of a scene under different illumination. This problem is highly related to inverse rendering where the aim is to estimate the image formation factors (e.g., shape, reflectance, lighting) of a scene. Several methods propose to handle this application either by directly estimating the relit images [Xu et al. 2018], estimating the individual factors [Xu et al. 2019], or by utilizing NeRF [Bi et al. 2020a,b; Boss et al. 2021; Srinivasan et al. 2021; Zhang et al. 2021]. However, these approaches focus on simple scenes or single objects, and require multiple images as the input. For more complex scenes, Philip et al. [2019] propose a relighting approach for outdoor scenes, while Philip et al. [2021] and Wu et al. [2022] focus on indoor scenes. However, both of these techniques use several images of the scene as the input.

Several techniques [Li et al. 2020, 2022; Sengupta et al. 2019; Wang et al. 2021] propose to estimate all the image formation factors including shape, reflectance, and lighting, from a single image. Sengupta et al. [2019] propose an inverse rendering network to estimate albedo, normal, and a single environment lighting. Li et al. [2020] extend this work to estimate per-pixel lighting, as well as roughness and depth. Wang et al. [2021] further propose to estimate 3D lighting of the scene through volumetric spherical Gaussian. Moreover, Li et al. [2022] present a holistic scene reconstruction system that estimates the reflectance, shape, and parameteric 3D lighting. These techniques demonstrate impressive results for object insertion, material editing, and dramatic lighting change [Li et al. 2022] (e.g., covering a window). While they could potentially be used to perform pixel reshading, these methods do not meet the quality requirement for our application.

3 ALGORITHM

Given a single RGB image I, captured with a camera at location $\mathbf{c}$, our primary goal is to synthesize an image I' from a novel view $\mathbf{c}'$. Similar to most existing methods [Han et al. 2022; Jampani et al. 2021], we assume the depth can be obtained with a reasonable accuracy using single image depth estimation techniques [Ranftl et al. 2022].

We begin by discussing the rendering equation [Kajiya 1986], a reasonably expressive rendering model, to describe the relationship between the input and novel view images. Formally, the rendering equation describes the total outgoing radiance $L_o(\mathbf{x}, \omega_o)$ at a 3D point $\mathbf{x}$ along the viewing direction ω_o as follows:

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_\Omega f_r(\mathbf{x}, \omega_o, \omega_i)\, L_i(\mathbf{x}, \omega_i)\, \cos(\theta_i)\, d\omega_i, \quad (1)$$

where L_e and L_i are the emitted and incoming radiances, respectively, ω_i is the incoming direction, and f_r is the bidirectional reflectance distribution function (BRDF). Moreover, θ_i is the angle between ω_i and the surface normal, and the integral is taken over the entire hemisphere Ω over the surface point.

As shown in Fig. 3, the appearance of a surface point $\mathbf{x}$ in the input and novel images is determined by the outgoing radiance

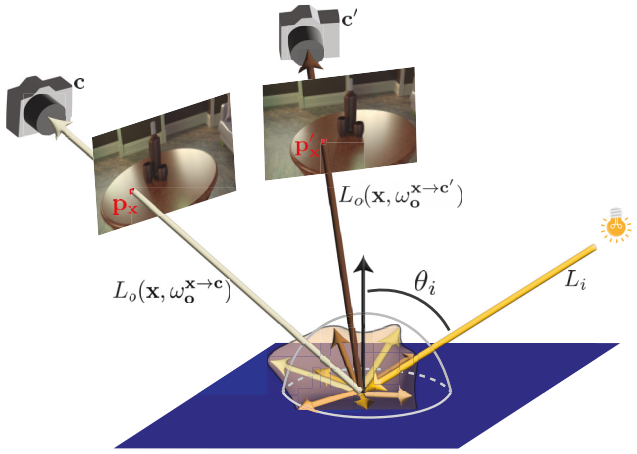

Fig. 3. We visualize the image formation process for the input ($\mathbf{c}$) and novel ($\mathbf{c}'$) cameras. A surface point $\mathbf{x}$ appears at two different locations ($\mathbf{p_x}$ and $\mathbf{p'_x}$) in the input and novel images. Moreover, the shading of point $\mathbf{x}$ in the two images is determined by $L_o(\mathbf{x}, \omega_o^{\mathbf{x}\to\mathbf{c}})$ and $L_o(\mathbf{x}, \omega_o^{\mathbf{x}\to\mathbf{c}'})$, and thus is different. Note that the incoming radiance L_i, surface normal (and consequently θ_i), and the BRDF (shown with curly black line), are the same for both the input and novel view images.

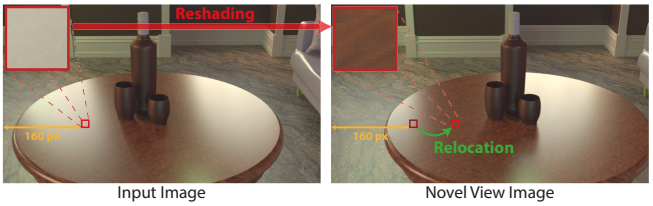

Fig. 4. We show an input and a novel view image. The same point on the table appears at different locations and with different shadings in the input and novel images. Therefore, the view synthesis process can be divided into two tasks of pixel reshading and relocation.

$L_o(\mathbf{x}, \omega_o^{\mathbf{x}\to\mathbf{c}})$ and $L_o(\mathbf{x}, \omega_o^{\mathbf{x}\to\mathbf{c}'})$, respectively. Here, $\omega_o^{\mathbf{x}\to\mathbf{c}}$ is the direction from the surface point to input camera location $\mathbf{c}$, while $\omega_o^{\mathbf{x}\to\mathbf{c}'}$ represents the direction to the novel camera at position $\mathbf{c}'$.

Based on this analysis, we observe that the appearance of point $\mathbf{x}$ in the input and novel images differs in two major ways: **1)** The point $\mathbf{x}$ appears with different shadings in the input and novel images as its appearance is determined by $L_o(\mathbf{x}, \omega_o^{\mathbf{x}\to\mathbf{c}})$ and $L_o(\mathbf{x}, \omega_o^{\mathbf{x}\to\mathbf{c}'})$, respectively. **2)** The location of this point in the two images is different; $\mathbf{p_x}$ and $\mathbf{p'_x}$ in the input and novel images, respectively. This is determined by the intersection of the rays along directions $\omega_o^{\mathbf{x}\to\mathbf{c}}$ and $\omega_o^{\mathbf{x}\to\mathbf{c}'}$ with the image planes of the input and novel cameras, respectively.

Therefore, we can describe the view synthesis process through two tasks of pixel reshading and relocation, as shown in Fig. 4. Existing modular approaches [Jampani et al. 2021; Shih et al. 2020], synthesize novel view images by warping the input image to the novel view using the input depth. As such, they mainly focus on the pixel relocation task and ignore the pixel reshading process, which is responsible for the view-dependent effects. The end-to-end systems [Han et al. 2022; Li and Kalantari 2020], on the other hand, attempt to learn both pixel reshading and relocation processes

by minimizing the loss between the estimated and ground truth novel view images. However, these systems often ignore the pixel reshading task as the contribution of the shading differences to the appearance loss is small; view-dependent highlight are often concentrated in small regions in each scene. As such, these techniques are not able to properly handle the view-dependent effects.

To address this problem, we propose to treat pixel reshading and relocation as two independent tasks. Specifically, we first adjust the shading of the input image according to the novel view camera. We then use the reshaded image as the input to the approach by Wang et al. [2022] to relocate the pixels and produce the final image. Below we discuss our approach in detail.

3.1 Pixel Reshading

Our goal is to take the input image I and produce a reshaded image I_s that has the same shading as the novel view image. This necessitates changing the shading of input pixel $\mathbf{p_x}$ from $L_o(\mathbf{x}, \omega_o^{\mathbf{x}\to\mathbf{c}})$, to the shading of the corresponding pixel in the novel image $\mathbf{p}'_\mathbf{x}$, i.e., $L_o(\mathbf{x}, \omega_o^{\mathbf{x}\to\mathbf{c}'})$. Note that at this stage, we are not interested in pixel relocation and reshading occurs in the input camera frame.

According to the rendering equation (Eq. 1), performing the reshading process requires estimating various components: the lighting L_e (emitters), material properties f_r, incoming radiance from all directions going through the hemisphere L_i, and the normals (to compute θ_i). Once these quantities are estimated, it is possible to recompute the shading of pixel $\mathbf{p_x}$ in the input image, by evaluating the integral in Eq. 1 using the outgoing direction of the corresponding pixel in the novel view image $\omega_o^{\mathbf{x}\to\mathbf{c}'}$. Note that the outgoing direction can be easily inferred from the input depth and the camera positions (provided relatively to avoid the need for estimating the input camera pose).

Unfortunately, estimating all of the aforementioned quantities from a single image is an extremely challenging problem. While there are existing techniques [Li et al. 2020, 2022; Sengupta et al. 2019; Wang et al. 2021] that estimate these various factors to a great extent, the quality of their re-rendered images falls short of the requirements for our view synthesis application.

Therefore, we instead propose to directly learn the reshaded image from the input image using a neural network. Although simple, as shown in Sec. 4 and in the supplementary video, our method is able to handle this challenging problem reasonably well and produce results with plausible moving highlights. In the following sections, we describe our dataset, inputs, architecture, and training process.

3.2 Dataset

To train our reshading network, we need a dataset of input-reshaded image pairs, which is currently not available. Unfortunately, obtaining such a dataset from real scenes is extremely challenging. Capturing the reshaded image necessitates taking a picture of the scene from the input camera view, but with the light rays going towards a different camera. One potential solution is to take a large number of images of a scene and use neural radiance field (NeRF) [Mildenhall et al. 2020] to reconstruct the radiance field of the scene. This radiance field can then be used to produce the reshaded images.

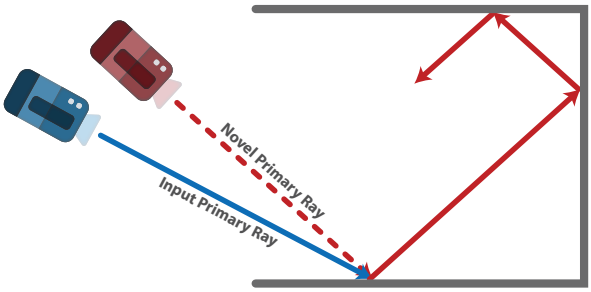

Fig. 5. We visualize our modification to the path tracer to render the reshaded images. We trace a primary ray to find the first intersection from the input camera. We then find the ray from the novel camera to this point (novel primary ray). This ray is then used for shading computation at the intersection point and generation of the secondary ray.

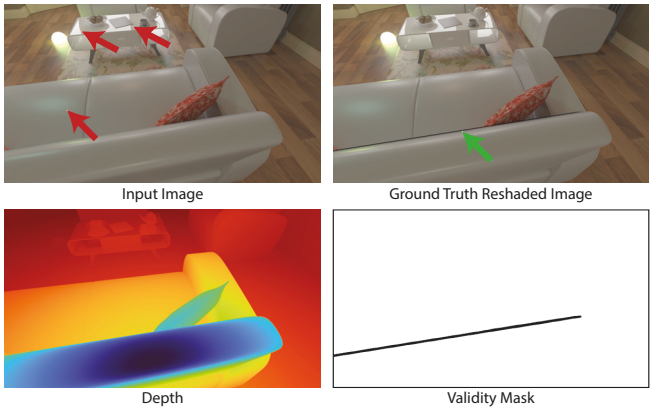

Fig. 6. For each training example in our dataset, we store the input and ground truth reshaded images, as well as the depth and validity mask. The red arrows point to the highlights in the input image that are moved in the reshaded image. Note that the objects in the reshaded image are in the same location as the input image, since reshading happens in the input camera frame. Small areas in the reshaded image (indicated by the green arrow) contain incorrect shading. We mask these out using the validity mask in our training loss.

However, generating a large scale dataset using this approach is difficult. Additionally, even the state-of-the-art approaches [Kerbl et al. 2023; Kopanas et al. 2022; Verbin et al. 2022] struggle to produce high-quality view-dependent effects on arbitrary surfaces.

Therefore, we propose to generate our input-reshaded image pairs synthetically. Specifically, we use the Tungsten renderer [Bitterli 2014] and render our input images using a large number of samples per pixel. We then slightly modify the path tracer to obtain the corresponding reshaded images, as shown in Fig. 5. To do this, we trace primary rays from the input camera (input primary ray) to find the first intersection points. We then calculate the rays connecting the novel camera to these intersection point (novel primary ray). These novel primary rays are then used for shading and generating all the additional secondary rays. An example input-reshaded image pair from our dataset is shown in Fig. 6 (top row).

Note that some regions from the input image are occluded in the novel camera. We could easily detect and mask these areas by performing a visibility check with the novel primary ray. However,

Fig. 7. Scenes used to create the synthetic dataset.

we choose not to do so to provide more content for our network to learn from. Most of the occluded areas will be shaded correctly as if they are not obscured from the camera. However, small regions (see the green arrow in Fig. 6), typically along the boundaries of objects, will be incorrectly shaded. These are the cases where the angle between the surface normal and novel primary ray is greater than 90 degrees. We detect these regions and create a validity mask, as shown in Fig. 6, which is used to mask out such areas when computing our training loss. Note that since we are using Monte Carlo rendering, each pixel is rendered by tracing a large number of rays. We mark a pixel as invalid if any of such rays does not satisfy our constraint. This is why the line in the validity mask appears to be thicker than the problematic region in the reshaded image.

We use the above approach to generate our synthetic dataset using 9 scenes, shown in Fig. 7, provided by Bitterli [2016]. For each scene, we render 200 input-reshaded pairs by randomly placing the input and novel cameras inside the scene. We randomly choose the novel cameras inside a sphere, centered on the input camera, with radii ranging from 0.1 to 0.3. Note that since all the scenes have similar global scale, the chosen radius range corresponds to a reasonable and uniform camera movement in all the training scenes. For every image pair, we randomly change the texture and material properties of the objects in the scene. By default, most scenes only use the environment map as the light source. To increase the robustness of our approach, we add multiple random colored orbs into the scene at random locations. We render 1280×720 high dynamic range (HDR) images with 8K samples per pixel and for each example, we store the input and reshaded images, as well as the depth, validity mask, and the metadata of the cameras. Our training data for one example is shown in Fig. 6.

3.3 Inputs

For our network to be able to properly reshade an input image, we need to provide the depth information along with the novel camera position to our network. The novel camera position is a 3-channel vector containing position of the novel camera relative to the input camera. Similar to most current single image view synthesis methods, we estimate the depth map using an existing single image depth estimation method (Ranftl et al.'s approach [2021; 2022] in our implementation). Instead of passing the depth to our network, however, we first convert it to disparity. We then scale it by a factor of 1/4 and clamp it to one. This ensures that the disparity

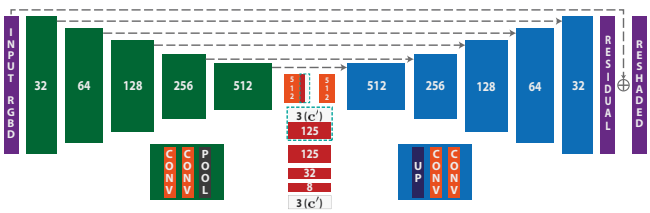

Fig. 8. We show the architecture of our reshading network. Each convolutional layers (shown in orange) is followed by a LeakyReLU activation, except the last layer that uses tanh activation. We use average 2×2 pooling for downsampling, while we use bilinear upsampling to increase the resolution. We use an MLP to convert the 3 channel novel camera position vector to a 125-channel feature vector. We then concatenate the original camera position vector with this feature vector. The result is then replicated and attached to the bottleneck feature map. The dashed lines represent skip connections. Note that our network estimate the residual image which is added to the input to obtain the reshaded image.

is in the range [0, 1] and it covers the depth from 0.25 to infinity. Moreover, we apply frequency encoding [Mildenhall et al. 2020] with 5 frequencies (11 channels; original plus 5 sines and 5 cosines) to the input disparity to allow the network to effectively use the disparity, particularly for far away regions. Frequency encoding essentially increases the resolution of the disparity, while remaining in the range [0, 1]; similar disparity values will have significantly different representation in the frequency domain.

To summarize, we use the input RGB image, frequency encoded disparity map, and the relative novel camera position as the input to our network to produce the reshaded image. The effect of using the disparity map and frequency encoding are shown in Figs. 11 and 12, respectively.

3.4 Architecture

We utilize a UNet [Ronneberger et al. 2015] style encoder-decoder style architecture consisting of 5 downsampling/upsampling layers. The encoder takes the input image and frequency encoded disparity (3+11 channels) and produces a bottleneck feature map of size $H/32 \times W/32 \times 512$, where H and W are the height and width of the input image, respectively. The three channel novel camera position vector is converted to a 125-channel feature vector using a multilayer perceptron (MLP) with a series of fully connected layers. This feature vector is then concatenated with the original 3-channel camera position vector to produce our novel camera features. This is then replicated and concatenated with the bottleneck feature map from the encoder (map of size $H/32 \times W/32 \times 640$). The concatenated feature map is then used as the input to the decoder to produce a 3-channel residual image. The residual is then added to the input to produce the reshaded image. Our architecture is shown in Fig. 8.

3.5 Training

We perform a series of augmentations to improve the generalization ability of our network. We take 384×384 random crops of the HDR synthetic dataset and convert the input and ground truth reshaded pairs to low dynamic range images by applying random exposure (scale factor between 3 and 10) and gamma correction (γ between 2.2 and 5). In addition, we randomly scale the disparity by a factor

of f and the camera position by a factor of $1/f$ simultaneously. This increases the range of scene scales in our training data.

Since this problem is highly ill-posed, we perform the training using a combination of $\mathcal{L}_1$ and perceptual losses. Specifically, our loss consists of the following three terms:

$$\mathcal{L} = \lambda_1 \mathcal{L}_1 + \lambda_{\text{vgg}} \mathcal{L}_{\text{VGG}} + \lambda_{\text{style}} \mathcal{L}_{\text{style}}, \tag{2}$$

where the first term is the $\mathcal{L}_1$ loss between the estimated and ground truth reshaded images and is defined as follows:

$$\mathcal{L}_1 = \|\tilde{I}_s - I_s\|_1. \tag{3}$$

Moreover, the second term is a perceptual VGG-based loss and is defined as:

$$\mathcal{L}_{\text{VGG}} = \|\phi(\tilde{I}_s) - \phi(I_s)\|_2^2, \tag{4}$$

where ϕ represents the output features from the conv4_4 layer of VGG-19 [Simonyan and Zisserman 2014]. Furthermore, the third term is a perceptual VGG-based style loss and is defined as:

$$\mathcal{L}_{\text{style}} = \|G(\phi(\tilde{I}_s)) - G(\phi(I_s))\|_2^2, \tag{5}$$

where G computes the Gram matrix of the VGG features extracted from the estimated and ground truth reshaded images. Finally, λ_1, λ_{vgg}, and λ_{style} define the weight of each term in Eq. 2 and we set them to 1e-1, 1e-2, and 1, respectively. Note that we multiply the estimated and ground truth reshaded images by the validity mask before computing each loss term.

3.6 Pixel Relocation

Once our reshading network is trained, we can use it to reshade the input image during inference. We then use the reshaded image as the input to the approach by Wang et al. [2022] to reconstruct the final novel view image. This approach is designed to perform view and time interpolation using near duplicate photos. However, all the operations related to view synthesis utilize a single image. Therefore, we isolate the view synthesis component and use it to generate novel views from a single image.

The view synthesis component of this approach is an enhanced version of the technique by Shih et al. [2020]. Specifically, using the depth, this method first constructs a layered depth image (LDI) representation [Shade et al. 1998]. It then inpaints the occluded regions and produces LDI features using a network. The LDI features are then warped to the novel view and combined using a subsequent network to produce the final image. Note that our reshaded image is different for each view, which could potentially change the inpainting results, and consequently affect coherency of the synthesized views. However, we did not observe this effect in practice. As shown in the supplementary video, our results are coherent.

We note that our approach can be combined with any view synthesis technique that focuses on pixel relocation. We demonstrate this in Table 1, where we examine the performance of our approach using Shih et al.'s method [2020] (3D Photo) for pixel relocation.

4 RESULTS

We implement our approach in PyTorch and use Adam [Kingma and Ba 2015] with the default parameters for training. We use a learning rate of 1e-4 for 300K iterations and 1e-5 for another 200K iterations. Our training takes 5 days on an Nvidia 2080 Ti GPU.

Fig. 9. We show comparisons against SVMPI [Tucker and Snavely 2020] and 3D Moments [Wang et al. 2022]. Only our approach is able to move the highlights in different views. Note that we carefully select the insets to cover roughly the same regions in the two views to be able to demonstrate the view dependent effects.

We compare our approach against single image view synthesis approaches by Tucker and Snavely [2020] (SVMPI) and Wang et al. [2022] (3D Moments). SVMPI is trained in an end-to-end manner on a multi-view dataset and ideally should be able to handle the view-dependent effects. On the other hand, 3D Moments, which we use for pixel relocation, is a modular approach that is not able to

Table 1. We show numerical comparisons against the other approaches on three synthetic scenes by evaluating the error between the ground truth and novel view images in terms of PSNR, SSIM, and LPIPS.

Scene	Method	PSNR↑	SSIM↑	LPIPS↓
Veach Ajar	SVMPI	22.72	0.877	0.0428
	3D Photo	30.06	**0.962**	0.0200
	3D Photo + Ours	**30.70**	**0.962**	0.0198
	3D Moments	29.78	**0.962**	0.0149
	3D Moments + Ours	30.41	**0.962**	**0.0147**
Bathroom	SVMPI	20.27	0.602	0.1255
	3D Photo	29.96	0.907	0.0329
	3D Photo + Ours	30.84	0.910	0.0323
	3D Moments	32.03	0.951	0.0284
	3D Moments + Ours	**33.12**	**0.953**	**0.0281**
Modern Hall	SVMPI	22.73	0.763	0.0759
	3D Photo	32.63	0.950	0.0230
	3D Photo + Ours	**32.99**	0.951	0.0229
	3D Moments	30.98	0.951	0.0197
	3D Moments + Ours	31.21	**0.953**	**0.0196**

move the highlights. We use the code provided by the authors to generate the results. We use images from several datasets, including Holopix50K [Hua et al. 2020], Open Images V7 [Kuznetsova et al. 2020] and Shiny [Wizadwongsa et al. 2021]. Here, we show the image results, but the differences can be better observed in the supplementary video.

In Fig. 9, we show comparisons against the other techniques on five scenes. For each scene, we show the results for two different views. We have carefully selected the insets, so they roughly cover the same region in the two views. Therefore, each approach's ability to adjust the shading based on the view can be observed by comparing the two views. Overall, 3D Moments produce results where the shading of the two views are almost identical. In some cases, SVMPI slightly alters the position of the highlights, but when doing so, it disturbs the texture underneath. Additionally, it produces slightly overblurred results. Our approach, on the other hand, produces detailed images with moving highlights. For example, in the first and fourth rows, our approach moves the highlight to the right and left, respectively, when transitioning from view 1 to 2. Note that our method does not leak the highlights to the dark region in the top row and the diffuse key fob in the fourth row. In the second row, our method produces results with slightly darker shading in the second view, while keeping the underlying texture intact. Finally, in the third and last rows, our approach is able to properly move the highlights (to the left from view 1 to 2) on the red structure and the burger bun, respectively.

Furthermore, we numerically compare our view synthesis results against the other techniques on three synthetic scenes in Table 1. To demonstrate that our approach can be used with any pixel relocation method, we show results with both 3D moments [Wang et al. 2022] and Shih et al.'s approach [2020] (3D Photo). As seen, our approach improves the performance of both modular relocation methods. Note

Table 2. We numerically evaluate the effect of reshading in isolation. Our reshading network produces results that are closer to the ground truth than the input.

Scene	Method	PSNR↑	SSIM↑	LPIPS↓
Veach Ajar	Input	35.45	0.993	0.0012
	Ours	**40.10**	**0.994**	**0.0008**
Bathroom	Input	36.50	0.991	0.0024
	Ours	**41.20**	**0.992**	**0.0020**
Modern Hall	Input	39.99	0.989	0.0015
	Ours	**42.71**	**0.989**	**0.0012**

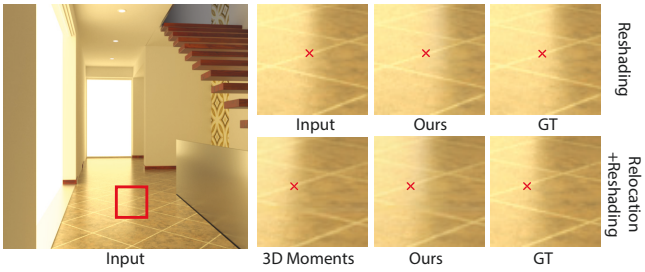

Fig. 10. We show our reshading (top) and view synthesis (bottom) results on the Modern Hall scene. Our approach is able to properly move the highlights during the reshading process (top) and produce novel view images that better match the ground truth than existing techniques (bottom).

that SSIM and LPIPS are highly sensitive to the textures, but are not sensitive to the smooth highlights. As such, these metrics do not fully reflect our quality improvement. Moreover, we evaluate our reshading network in isolation (see Table 2), by measuring the error between our synthesized and ground truth reshaded images. By appropriately moving the highlights, our approach produces results that are significantly closer to the ground truth than the input images (without reshading). This is shown visually in Fig. 10 for the Modern Hall example. Our approach properly moves the highlights (top row), and thus is able to synthesize a novel view image that better matches the ground truth than 3D Moments (bottom row).

Next, we discuss the effect of several design choices in our approach numerically (Table 3) and visually (Figs. 11, 12, and 13). In Fig. 11, we demonstrate that without the disparity as the input, our reshading network is not able to detect the depth discontinuities and smears the shading of the tomato on the bowl. Moreover, as shown in Fig. 12, without frequency encoding, our network has difficulty handling the objects that are far away and incorrectly changes their shading. Finally, in Fig. 13 we show the result of directly concatenating the camera pose with the bottleneck features (w/o MLP). As seen, without the MLP, our network cannot effectively utilize the camera information and incorrectly changes the shading of the background areas.

5 LIMITATIONS

Although we have demonstrated that our simple network can produce reasonable results, this is an extremely challenging problem and, as shown in Fig. 14, our approach has several limitations. For

Table 3. We show numerical comparisons against variations of our approach without disparity, frequency encoding, and MLP. The results are averaged over the three synthetic scenes. As shown, all these components are necessary to achieve the best results.

Method	PSNR↑	SSIM↑	LPIPS↓
w/o disparity	30.98	0.950	0.0213
w/o FE	31.11	0.950	0.0210
w/o MLP	31.17	0.951	0.0211
Ours	**31.58**	**0.956**	**0.0208**

Fig. 11. We evaluate the effect of using disparity as the input to our shading network.

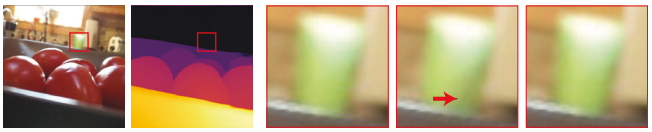

Fig. 12. We compare our results against a version of our approach where we do not apply frequency encoding to the input disparity.

Fig. 13. We compare our results against a version of our approach where the camera position is directly concatenated to the bottleneck features of the UNet.

example, we are currently not able to handle highly specular surfaces, such as mirrors. As shown in Fig. 14 (mirror on the right wall), our technique is not able to correctly move the content inside the mirror between the two reshaded images. Additionally, in cases where the light sources are very close to diffuse surfaces, they create strong saturated regions (see the area underneath the mirror). In these cases, our reshading network interpret these as highlights and moves them between different views.

6 CONCLUSION

We have presented a method to handle view dependent effects in single image novel view synthesis. Specifically, we propose to split the task of view synthesis into pixel reshading and relocation processes and treat them independently. We use a network to adjust the shading of the input image according to the novel camera. We then use the reshaded image as the input to an existing view synthesis method to perform the pixel relocation task. We demonstrate that our method produces plausible results with view-dependent highlights that are better than the existing methods.

Fig. 14. We show the input image as well as two reshaded images corresponding to different views. As seen, our method is not able to properly move the content of the mirror on the right wall in the two reshaded images. Additionally, while our method correctly moves the highlights on the ground, it detects the strong saturated regions under the mirror as highlights and move them in the reshaded images.

ACKNOWLEDGMENTS

The authors would like to thank the reviewers for their comments and suggestions. This work was funded by Leia Inc. (contract #415290). Nima Khademi Kalantari was in part supported by CAREER Award (#2238193). Portions of this research were conducted with the advanced computing resources provided by Texas A&M High Performance Research Computing.

REFERENCES

Sai Bi, Zexiang Xu, Pratul Srinivasan, Ben Mildenhall, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. 2020a. Neural reflectance fields for appearance acquisition. *arXiv preprint arXiv:2008.03824* (2020).

Sai Bi, Zexiang Xu, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. 2020b. Deep reflectance volumes: Relightable reconstructions from multi-view photometric images. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*. Springer, 294–311.

Benedikt Bitterli. 2014. Tungsten Renderer. https://github.com/tunabrain/tungsten.

Benedikt Bitterli. 2016. Rendering resources. https://benedikt-bitterli.me/resources/.

A. Blake. 1985. Specular Stereo. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence - Volume 2* (Los Angeles, California) *(IJCAI'85)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 973–976.

Mark Boss, Raphael Braun, Varun Jampani, Jonathan T Barron, Ce Liu, and Hendrik Lensch. 2021. Nerd: Neural reflectance decomposition from image collections. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 12684–12694.

Eric R. Chan, Koki Nagano, Matthew A. Chan, Alexander W. Bergman, Jeong Joon Park, Axel Levy, Miika Aittala, Shalini De Mello, Tero Karras, and Gordon Wetzstein. 2023. GeNVS: Generative Novel View Synthesis with 3D-Aware Diffusion Models. In *arXiv*.

Rafail Fridman, Amit Abecasis, Yoni Kasten, and Tali Dekel. 2023. Scenescape: Text-driven consistent scene generation. *arXiv preprint arXiv:2302.01133* (2023).

Jiatao Gu, Alex Trevithick, Kai-En Lin, Josh Susskind, Christian Theobalt, Lingjie Liu, and Ravi Ramamoorthi. 2023. NerfDiff: Single-image View Synthesis with NeRF-guided Distillation from 3D-aware Diffusion. In *International Conference on Machine Learning*.

Yuxuan Han, Ruicheng Wang, and Jiaolong Yang. 2022. Single-View View Synthesis in the Wild with Learned Adaptive Multiplane Images. In *ACM SIGGRAPH 2022 Conference Proceedings* (Vancouver, BC, Canada) *(SIGGRAPH '22)*. Association for Computing Machinery, New York, NY, USA, Article 14, 8 pages. https://doi.org/10.1145/3528233.3530755

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems* 33 (2020), 6840–6851.

Yiwen Hua, Puneet Kohli, Pritish Uplavikar, Anand Ravi, Saravana Gunaseelan, Jason Orozco, and Edward Li. 2020. Holopix50k: A Large-Scale In-the-wild Stereo Image Dataset. In *CVPR Workshop on Computer Vision for Augmented and Virtual Reality, Seattle, WA, 2020*.

Varun Jampani, Huiwen Chang, Kyle Sargent, Abhishek Kar, Richard Tucker, Michael Krainin, Dominik Kaeser, William T Freeman, David Salesin, Brian Curless, and Ce Liu. 2021. SLIDE: Single Image 3D Photography with Soft Layering and Depth-aware Inpainting. In *Proceedings of the IEEE International Conference on Computer Vision*.

James T Kajiya. 1986. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. 143–150.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (July 2023). https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Georgios Kopanas, Thomas Leimkühler, Gilles Rainer, Clément Jambon, and George Drettakis. 2022. Neural Point Catacaustics for Novel-View Synthesis of Reflections. *ACM Transactions on Graphics* 41, 6 (2022), Article–201.

Johannes Kopf, Suhib Alsisan, Francis Ge, Yangming Chong, Kevin Matzen, Ocean Quigley, Josh Patterson, Jossie Tirado, Shu Wu, and Michael F. Cohen. 2019. Practical 3D photography. *CVPR Workshop on Computer Vision for AR/VR* (2019).

Johannes Kopf, Kevin Matzen, Suhib Alsisan, Ocean Quigley, Francis Ge, Yangming Chong, Josh Patterson, Jan-Michael Frahm, Shu Wu, Matthew Yu, Peizhao Zhang, Zijian He, Peter Vajda, Ayush Saraf, and Michael Cohen. 2020. One Shot 3D Photography. *ACM Trans. Graph.* 39, 4, Article 76 (aug 2020), 13 pages. https://doi.org/10.1145/3386569.3392420

Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, Tom Duerig, and Vittorio Ferrari. 2020. The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV* (2020).

Leia. 2023. Leia Inc. https://www.leiainc.com/.

Qinbo Li and Nima Kalantari. 2020. Synthesizing Light Field From a Single Image with Variable MPI and Two Network Fusion. *ACM Transactions on Graphics* 39, 6 (2020). https://doi.org/10.1145/3414685.3417785

Zhengqin Li, Mohammad Shafiei, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. 2020. Inverse Rendering for Complex Indoor Scenes: Shape, Spatially-Varying Lighting and SVBRDF From a Single Image. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhengqin Li, Jia Shi, Sai Bi, Rui Zhu, Kalyan Sunkavalli, Miloš Hašan, Zexiang Xu, Ravi Ramamoorthi, and Manmohan Chandraker. 2022. Physically-Based Editing of Indoor Scene Lighting from a Single Image. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VI*. Springer, 555–572.

Gerrit Lochmann, Bernhard Reinert, Tobias Ritschel, Stefan Müller, and Hans-Peter Seidel. 2014. Real-time Reflective and Refractive Novel-view Synthesis, Jan Bender, Arjan Kuijper, Tatiana von Landesberger, Holger Theisel, and Philipp Urban (Eds.). Eurographics Association, Darmstadt, Germany, 9–16. https://doi.org/10.2312/vmv.20141270

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*.

Simon Niklaus, Long Mai, Jimei Yang, and Feng Liu. 2019. 3D Ken Burns Effect from a Single Image. *ACM Trans. Graph.* 38, 6, Article 184 (nov 2019), 15 pages. https://doi.org/10.1145/3355089.3356528

Julien Philip, Michaël Gharbi, Tinghui Zhou, Alexei A. Efros, and George Drettakis. 2019. Multi-View Relighting Using a Geometry-Aware Network. *ACM Trans. Graph.* 38, 4, Article 78 (jul 2019), 14 pages. https://doi.org/10.1145/3306346.3323013

Julien Philip, Sébastien Morgenthaler, Michaël Gharbi, and George Drettakis. 2021. Free-viewpoint Indoor Neural Relighting from Multi-view Stereo. *ACM Transactions on Graphics* (2021). http://www-sop.inria.fr/reves/Basilic/2021/PMGD21

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. 2022. DreamFusion: Text-to-3D using 2D Diffusion. *arXiv* (2022).

René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. 2021. Vision Transformers for Dense Prediction. *ICCV* (2021).

René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. 2022. Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-Shot Cross-Dataset Transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 3 (2022).

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 234–241.

S. Roth and M.J. Black. 2006. Specular Flow and the Recovery of Surface Structure. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, Vol. 2. 1869–1876. https://doi.org/10.1109/CVPR.2006.290

Soumyadip Sengupta, Jinwei Gu, Kihwan Kim, Guilin Liu, David W. Jacobs, and Jan Kautz. 2019. Neural Inverse Rendering of an Indoor Scene From a Single Image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. 1998. Layered Depth Images. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*. Association for Computing Machinery, New York, NY, USA, 231–242. https://doi.org/10.1145/280814.280882

Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 2020. 3D Photography using Context-aware Layered Depth Inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

J Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 2022. 3D Neural Field Generation using Triplane Diffusion. *arXiv preprint arXiv:2211.16677* (2022).

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

Sudipta N. Sinha, Johannes Kopf, Michael Goesele, Daniel Scharstein, and Richard Szeliski. 2012. Image-Based Rendering for Scenes with Reflections. *ACM Trans. Graph.* 31, 4, Article 100 (jul 2012), 10 pages.

Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. 2021. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7495–7504.

Pratul P Srinivasan, Tongzhou Wang, Ashwin Sreelal, Ravi Ramamoorthi, and Ren Ng. 2017. Learning to synthesize a 4D RGBD light field from a single image. In *Proceedings of the IEEE International Conference on Computer Vision*. 2243–2251.

Richard Tucker and Noah Snavely. 2020. Single-view View Synthesis with Multiplane Images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

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).

Qianqian Wang, Zhengqi Li, Brian Curless, David Salesin, Noah Snavely, and Janne Kontkanen. 2022. 3D Moments from Near-Duplicate Photos. In *CVPR*.

Zian Wang, Jonah Philion, Sanja Fidler, and Jan Kautz. 2021. Learning indoor inverse rendering with 3d spatially-varying lighting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 12538–12547.

Daniel Watson, William Chan, Ricardo Martin-Brualla, Jonathan Ho, Andrea Tagliasacchi, and Mohammad Norouzi. 2022. Novel view synthesis with diffusion models. *arXiv preprint arXiv:2210.04628* (2022).

Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. 2020. SynSin: End-to-end View Synthesis from a Single Image. In *CVPR*.

Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. 2021. NeX: Real-time View Synthesis with Neural Basis Expansion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xiuchao Wu, Jiamin Xu, Zihan Zhu, Hujun Bao, Qixing Huang, James Tompkin, and Weiwei Xu. 2022. Scalable neural indoor scene rendering. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–16.

Zexiang Xu, Sai Bi, Kalyan Sunkavalli, Sunil Hadap, Hao Su, and Ravi Ramamoorthi. 2019. Deep view synthesis from sparse photometric images. *ACM Transactions on Graphics (ToG)* 38, 4 (2019), 1–13.

Zexiang Xu, Kalyan Sunkavalli, Sunil Hadap, and Ravi Ramamoorthi. 2018. Deep image-based relighting from optimal sparse samples. *ACM Transactions on Graphics (ToG)* 37, 4 (2018), 1–13.

Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. 2021. pixelNeRF: Neural Radiance Fields from One or Few Images. In *CVPR*.

Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. 2021. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–18.

Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A. Efros. 2016. View Synthesis by Appearance Flow. In *Computer Vision – ECCV 2016*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (Eds.). Springer International Publishing, Cham, 286–301.

Henning Zimmer, Fabrice Rousselle, Wenzel Jakob, Oliver Wang, David Adler, Wojciech Jarosz, Olga Sorkine-Hornung, and Alexander Sorkine-Hornung. 2015. Path-space Motion Estimation and Decomposition for Robust Animation Filtering. *Computer Graphics Forum* 34, 4 (2015), 131–142.