

# Online Learning of Neural Surface Light Fields alongside Real-time Incremental 3D Reconstruction

Yijun Yuan and Andreas Nüchter

**Abstract**—Immersive novel view generation is an important technology in the field of graphics and has recently also received attention for operator-based human-robot interaction. However, the involved training is time-consuming, and thus the current test scope is majorly on object capturing. This limits the usage of related models in the robotics community for 3D reconstruction since robots (1) usually only capture a very small range of view directions to surfaces that cause arbitrary predictions on unseen, novel direction, (2) requires real-time algorithms, and (3) work with growing scenes, e.g., in robotic exploration. The paper proposes a novel Neural Surface Light Fields model that copes with the small range of view directions while producing a good result in unseen directions. Exploiting recent encoding techniques, the training of our model is highly efficient.

In addition, we design Multiple Asynchronous Neural Agents (MANA), a universal framework to learn each small region in parallel for large-scale growing scenes. Our model learns online the Neural Surface Light Fields (NSLF) aside from real-time 3D reconstruction with a sequential data stream as the shared input. In addition to online training, our model also provides real-time rendering after completing the data stream for visualization. We implement experiments using well-known RGBD indoor datasets, showing the high flexibility to embed our model into real-time 3D reconstruction and demonstrating high-fidelity view synthesis for these scenes. The code is available on [github<sup>1</sup>](#).

**Index Terms**—Mapping; SLAM

## I. INTRODUCTION

**I**n robotics, mapping, and 3D reconstruction have long been of great interest and have been studied for decades. Initially, researchers focused on 3D point clouds and voxel grids, and this later shifted towards using Signed Distance Functions (SDF). An SDF, is the starting point of many of the following state-of-the-art papers. It has been well-developed from Point-to-SDF [1] to SDF-to-SDF [2], [3], from explicit voxel field [4] to neural implicit representation [5], [6]. These methods have achieved high-quality 3D reconstructions in real-time while maintaining high regression functionality of the neural models.

Aside from the reconstruction of geometries, neural rendering for colors is also a hot topic, however more in the field of computer graphics [7]. Neural rendering focuses on the synthesis of novel views from 3D models. The goal of this topic is to enhance the immersive experience for users. In the context of robotics, for instance, it is crucial for human-robot interaction or situation awareness of an operator of a telerobot.

Manuscript received: January, 24, 2023; Revised March, 29, 2023; Accepted April, 28, 2023.

This paper was recommended for publication by Editor Civera Javier upon evaluation of the Associate Editor and Reviewers' comments.

All authors are with Informatics XVII : Robotics, Julius-Maximilians-Universität Würzburg, Germany. {yijun.yuan|andreas.nuechter}@uni-wuerzburg.de

<sup>1</sup><https://jarome.github.io/NSLF-OL>

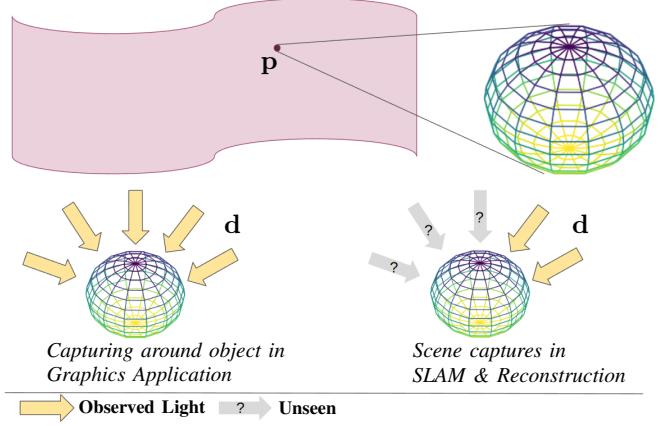


Fig. 1: Light difference between object capturing for graphics applications and scene capturing in robotic SLAM & reconstruction. At 3D point  $p$ , lights are cast from direction  $d$ . The sphere shows the  $S^2$  space for the direction vectors  $d$  that are partially covered by light rays.

In 2020, the Neural Radiance Field (NeRF) [8] was introduced as an innovative approach to high-resolution rendering for view synthesis, activating the trend of neural rendering. It learns a neural radiance field that produces both an occupancy field and a light field via differentiable rendering for custom immersive view synthesis, and it has extended the testing scope of 3D reconstruction to large scene-level [9], [10]. The drawback is the extremely high training time, and it is therefore unsuitable for real-time 3D reconstruction. Nevertheless, it still affects the field of 3D reconstruction. Recent works on 3D reconstructions, i.e., iMAP [11] and NICE-SLAM [12], build on the differentiable rendering from NeRF to approach an online reconstruction with color. However, their visualization of color appears blurred. It is worth mentioning that shape reconstruction has already been extensively studied and has already achieved high-quality 3D reconstruction performance. Aside from the high quality, the robotics community values online capable reconstructions.

However, NeRFs in the graphics community concentrate *more* on (1) object capturing instead of capturing large scene surfaces (captures contain dense view-directions to surface), (2) rendering high-quality images while caring less on depth and surface accuracies (shape-radiance ambiguity) [13], and (3) more on rendering (testing) speed instead of online training. Therefore, we explore the high potential to use surface reconstruction research as the basis of this topic to meet the interest of the robotics community to have a universal way to simulate real environments.

We utilize a real-time reconstruction model to provide surfaces and simplify the problem to online learn the light field on surfaces **aside from** real-time reconstruction as given in Fig. 3. Where our model relies on data from reconstruction without affecting it. During the online learning phase, we employ a colored point cloud. Subsequently, in the inference phase, we apply ray-rasterizing on the reconstruction mesh to obtain the point cloud for color prediction.

The Surface Light Fields (SLF) has been proposed by Wood et al. [14] to model the surface reflectance. For a surface point  $p$ , the radiance of a reflection ray (with direction  $d_o$ ) is computed from an accumulation of incident rays (with direction  $d_i$  for ray  $i$ ). However, due to the inefficient dense sampling requirement of SLF, more work has focused on modeling compression [15]. Recently, Chen et al. [16] and Yu et al. [17] have introduced Deep Surface Light Fields (DSLF), which utilize a neural network to replace the previous handcrafted formulation. Still, these approaches have an extremely slow training speed. Similarly, these techniques are mainly concentrated on capturing a 360-degree view of the object. Large-scale scenes have not been considered.

Unlike the graphics communities, in robotics, SLAM, and 3D reconstruction (1) *capture scene frames that contain only a limited range of surface view-direction* as depicted Fig. 1, apply for (2) *growing large-scale scenes without prior knowledge of their size* and require (3) *real-time processing speed*. These three points are the primary concerns for using Neural Surface Light Fields (NSLF) in robotics. Thus, this paper aims to address the limited view-direction challenge during testing, where most unseen directions will cause arbitrary results. To cope with this problem, we introduce Spherical Harmonics for *decoding*, as depicted in Fig. 2. Furthermore, relying on recent advances in graphics, i.e., the Multi-resolution Hash-encoding, we train grid-latent as an encoder for (3) real-time online learning. We further introduce Multiple Asynchronous Neural Agents (MANA) to handle (2), i.e., we handle large scenes without pre-knowing the size.

We take inspiration from [18] which claims that multiple local MLPs converge faster. However, [18] has to train all local MLPs with one optimizer because NeRF's differentiable rendering accumulates data from different MLPs. While Surface Lighting Field training does not depend on ray integration in NeRF. Therefore, each region has its model and optimizer to learn individually.

To deal with scalable data when data is distributed, we dynamically allocate neural models and optimizers for new regions and run them in a new thread independently as depicted in Fig. 3. Every region has a neural model and an optimizer running independently without synchronizing with others.

The contributions of this paper are as follows:

- Proposing a novel Neural Surface Light Fields model to address the issue of arbitrary prediction on unseen direction causing from the small range of view-direction to surface in SLAM & 3D reconstruction captures,
- Proposing the first framework (MANA) for online-learning neural surface light fields on growing large-scale scenes,

- Implementing MANA aside from real-time reconstruction for experiments. Both online learning and real-time testing are supported. Agents can be distributed to multiple GPUs.

## II. RELATED WORKS

Our work is mainly related to NSLF. Here, we also review the hotter sister topic, NeRF because its development process highly inspires and directs us to use NSLF online for the large scenes.

### A. Neural Radiance Fields (NeRF)

Mildenhall et al. [8] propose to represent a scene as a continuous mapping from five-dimensional coordinates  $(x, y, z, \theta, \phi)$  to volume density and view-dependent RGB color ( $\sigma RGB$ ), which are so-called NeRF representations. The training of NeRF relies on the use of differentiable rendering along rays. NeRF represents a significant advancement in graphics due to its immersive view-synthesis performance. However, NeRF does have some limitations. Firstly, the speed for training and rendering is extremely slow. Secondly, the NeRF model suffers from shape-radiance ambiguity.

1) *Speed-aware NeRF*: Although NeRFs show a very high quality of view synthesis, their computational cost for training and testing is extremely high. Therefore, lots of recent work shows interest in the speed issue. Tewari et al. [7] summarize and show many works exploring speedups using pruning [19], sampling [20], fast integration [21], data structure [22], [23], [24], and so on [18], [25], [26]. Many different data structures are also used to speed up the rendering and the training process benefits. A recently highlighted work from NVIDIA, Instant Neural Graphics Primitives, uses a multi-resolution hash encoding to realize fast NeRF training in a few seconds [24]. It provides an encoder with very fast convergence. We will also employ this encoder in our proposed model. In the most related scope, KiloNeRF, which uses thousands of tiny MLPs to imitate standard NeRF, shows a factor of 2000 speedup in rendering [18]. Real-time rendering already meets the needs of the field of computer graphics (CG).

However, from a robotics perspective, online algorithms such as 3D reconstruction and SLAM are always preferred. In such an environment, online training makes the task of light fields much more challenging.

2) *Depth Aided NeRF*: Another problem with NeRF is shape-radiance ambiguity. NeRF++ [13] points out that given an incorrect shape, there exists a family of radiance fields that perfectly explains the training images, but generalizes poorly to novel views. Although a well-trained NeRF does not show this phenomenon, imperfect depth prediction is common but is neglected since the application targets color prediction. However, this also leads to another research direction, a group of works uses depth as an input to reduce the complexity of geometry modeling. This group of works embed Structure-from-motion (SfM) [27], [28], Multi-view Stereo (MVS) [29], [30] or directly input RGBD data [31].

The above algorithms show a trend that the facilitation of geometry learning well-reduces the complexity of NeRF while

improving the high quality. Moreover, in the robotics community, recent mature real-time 3D reconstructions already provide high-quality online shape reconstruction [5].

Therefore, we consider solving the geometric and color regression separately. Where the geometry is estimated using a mature online 3D reconstruction model. Meanwhile, an NSLF is independently trained for surface colors. Without ray sampling, our model uses only surface points during training and inference. This breaks the constraint of NeRF's rendering loss and allows asynchronous training and inference for each intelligent agent.

### B. Neural Surface Light Fields (NSLF)

NSLF learns a mapping function  $f_{NSLF} : \mathbf{X} \times \mathbf{S}^3 \rightarrow \mathbf{C}$  where  $\mathbf{X} \subseteq \mathbb{R}^3$  denotes the point set on the surface,  $\mathbf{S}^3$  is the unit 3-sphere,  $\mathbf{C}$  denotes the RGB color space.

With the high regression capability of deep learning, Chen et al. [16] propose Deep Surface Light Fields (DSLF) which uses MLPs to replace the accumulation function. This parametric modeling greatly reduces the computational and spatial burden. And thus provides more efficient rendering. Yu et al. [17] have introduced Anisotropic Fourier Features Mapping (AFFM) to encode points. This increases convergence speed and rendering quality.

Similar to the NeRF design, the NSLFs also follow the encoding-decoding pattern. For a given point, its position vector is encoded as  $\mathbf{F}_p = \phi_p(\mathbf{p})$ . In the other branch, the direction vector is also encoded with a non-parameterized encoder, spherical harmonics, or frequency encoding:  $\mathbf{F}_d = \varphi_{sh/freq}(\mathbf{d})$ . Shallow MLPs are then applied to concatenated features (position encoding and direction) to predict color:  $\mathbf{c} = \phi_{dec}(\mathbf{F}_p, \mathbf{F}_d)$ .

However, similar to NeRF the NSLF has not been used in large-scale incremental 3D reconstruction. Thus, in the following, we fill this gap and provide an online learned neural surface light fields function for large scales.

## III. METHODOLOGY

In the following, we introduce our proposed NSLF model, that addresses the problem of limited view directions toward the surfaces. Then we further design a Multiple Asynchronous Neural Agents (MANA) framework that learns the NSLF alongside the reconstruction. The geometry regression that provides the surface relies on the existing incremental 3D reconstruction models.

### A. NSLF Design

In the use of capturing of 360 degrees of an object, for a certain surface point, the trained view directions are densely sampled over a large range and thus, the novel view inference is mostly on the touched view directions. However, in large-scale 3D SLAM and reconstruction, this feature of data acquisition is not guaranteed. Therefore, training on a small range of view directions will lead to arbitrary prediction on unseen views.

Unlike previous encoding schemes that concatenate position and direction encoding, in Fig. 2, we propose to learn Spherical Harmonics (SH) parameters from the position encodings.

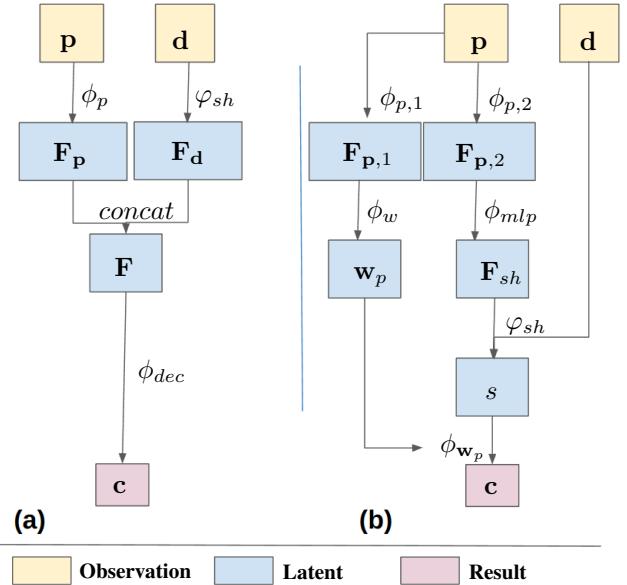


Fig. 2: Patterns of models designed for Neural Light Field. (a) is most widely used. Our (b) learn one sphere for each  $\mathbf{p}$  and use a  $\phi_{\mathbf{w}_p}$  to map the 1D value on direction  $\mathbf{d}$  to 3D RGB.

On the decoding side, we run deterministic formulas with a known SH basis.

To achieve the speedup from current techniques, we use the novel Multiresolution Hash Encoding [24] to bear the main burden of position encoding:  $\mathbf{F}_p = \phi_{p,\theta_{HG}}(\mathbf{p})$ . Such an encoding produces an encoding by interpolation of voxel features, which mitigates the global effect problem of MLPs in DSLF. In addition, the surfaces occupy a small space in a cubic region, and thus the allocation during use is more efficient in space. Considering that scene sequences only cover very limited direction space on surface surface, training on specific positions and directions easily leads to an arbitrary result in unseen view. To prevent arbitrary guessing in the unseen direction, we introduce learning of SH parameters that affect the full direction. We add an MLP-layer to generate the SH parameters:  $\mathbf{F}_{sh} = \phi_{mlp}(\phi_{\theta_{HG}}(\mathbf{p})) = (v_\ell^m)_{\ell:0 \leq \ell \leq \ell_{max}}^{m:l \leq m \leq \ell}$ . So here we are creating one sphere of latent to continuously represent a small space of 1 dimensional color range for corresponding lights on a point. Then, with the known spherical harmonics function  $Y_\ell^m : \mathbf{S}^2 \rightarrow \mathbb{C}$ , we extract the latent in a given direction of the sphere utilizing the SH formula  $\sum_{\ell=0}^{\ell_{max}} \sum_{m=-\ell}^{\ell} v_\ell^m Y_\ell^m(\mathbf{d})$  to deterministically decode on direction  $\mathbf{d}$ :  $s = \mathbf{F}_{sh}^T \varphi_{sh}(\mathbf{d})$ . Meanwhile, the other branch of the encoder computes features  $\mathbf{w}_p$  to parameterize MLPs  $\phi_{\mathbf{w}_p}$ . It maps (or extracts) the color range value  $s$  to the color  $\mathbf{c}$ .

Hash encoding requires normalizing the data into a unit cube [24]. This means that we need to know the scale of the data. Thus, using this model in each region avoids this problem as the regional scale is predefined. In the following, we present a framework that operates on the growing scenes.

### B. NSLF with Region-wise Neural Agents

DSLF [16] uses MLPs to learn an entire scene. But such a model is not robust for the incrementally creation of large

scenes, because newly fed data still changes the scene globally even the non-effected parts. Some ideas in NeRF provide possible solutions: (1) using voxels of multiple MLPs (KiloNeRF [18]) or (2) using a voxel multi-resolution representation (Instant-NGP [24]).

1) *Multiple Asynchronous Neural Agents*: We use RGBD and rasterization to directly approach points on the surface during training and testing respectively.

To speed up training while making it scalable, we divide the space equally into regions and dynamically assign models for each newly touched region. Each region is assigned with an Intelligent Agent (IA) that maintains its own **thread**, **neural model** and **optimizer** and is trained autonomously. In training mode, when data is fed into an agent, it will train continuously until it reaches max-iterations. In evaluation mode, the IA will predict input data and output colors. Since the implementation is done with neural network, we will refer to such an IA as a Neural Agent (NA) in the rest of this paper.

In addition, we assign an optimizer to each NA, making it an independent model that does the training itself. Each region maintains its own neural model and optimizer to train independently. With this feature, our model distinguishes itself from KiloNeRF which synchronizes voxels. And thus, our model is capable of asynchronous training. We call it Multiple Asynchronous Neural Agents (MANA) with each NA  $\mathbf{U} = (\phi_\theta, Optim)$ .

Then following KiloNeRF, we use an axis-aligned bounding box (AABB) to enclose the scene. We preset  $\mathbf{b}_{min}$  and  $\mathbf{b}_{max}$  as the minimum and maximum bound of AABB and discretize the space uniformly with the resolution  $\mathbf{r} = (r_x, r_y, r_z)$ . We assign to each grid cell the index  $\mathbf{i} = (i_x, i_y, i_z)$  for the NA  $\mathbf{U}_i$ . For growing scenes, we define extremely large box that

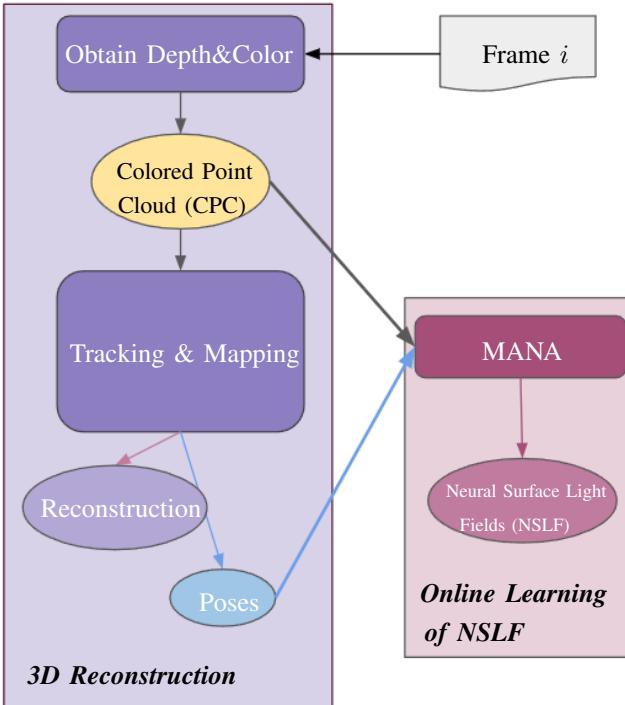


Fig. 3: MANA learns online a NSLF by serving as an **external function** to 3D reconstruction.

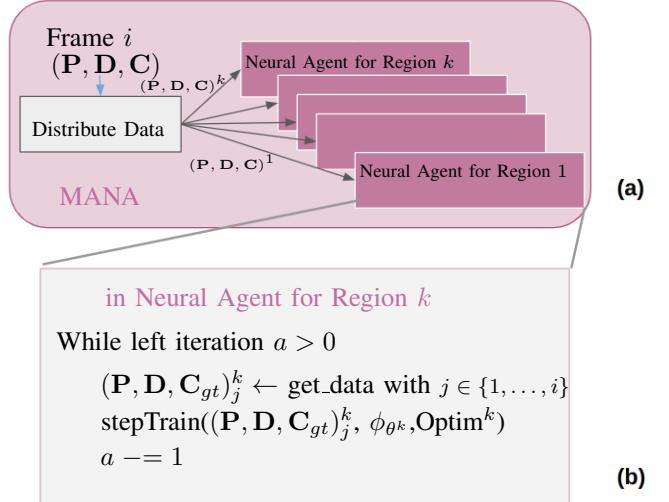


Fig. 4: Online Learning of NSLF. (a) shows MANA distributing data into different Neural Agents by region. Each Neural agent maintains its thread and optimizes individually as (b). cost almost nothing.

Given a RGBD frame, we unproject it to 3D space as a point cloud  $\mathbf{P} \in \mathbb{R}^{N_{\mathbf{P}} \times 3}$  and their corresponding color  $\mathbf{C} \in \mathbb{R}^{N_{\mathbf{P}} \times 3}$ . The corresponding direction  $\mathbf{D} \in \mathbb{R}^{N_{\mathbf{P}} \times 3}$  is also obtained for light fields. Each point  $\mathbf{p} \in \mathbf{P}$  is assigned to the corresponding region with

$$v(\mathbf{p}) = \lfloor (\mathbf{p} - \mathbf{b}_{min}) / (\mathbf{b}_{max} - \mathbf{b}_{min}) / \mathbf{r} \rfloor. \quad (1)$$

Then the color is predicted with the point  $\mathbf{p}_j \in \mathbf{P}$  and its direction vector  $\mathbf{d}_j \in \mathbf{D}$  as input:

$$\mathbf{c}_{j,pred} = \phi_{\theta(v(\mathbf{p}_j))}(\mathbf{p}_j, \mathbf{d}_j) \quad (2)$$

Thus, for each agent  $\mathbf{i}$ , it optimizes

$$\min_{\theta(\mathbf{i})} \sum_{j, v(\mathbf{p}_j)=\mathbf{i}} \|\mathbf{c}_j - \mathbf{c}_{j,pred}\|_2^2 \quad (3)$$

independently with  $\text{Optim}_i$ .

### C. Online Learning of Surface Light Fields

As indicated in Fig. 3, our MANA works aside from the real-time 3D reconstruction of the model. Note that, our model uses colored point clouds and poses sequentially from the reconstruction. But the training of MANA does not depend on the reconstruction result. The reconstruction mesh is only used during the testing, i.e., view rendering, where the surface is needed to determine the intersection points of rays. In our implementation, the main thread is used to feed data into MANA and to perform the reconstruction.

We plot the diagram of MANA as a frame is fed into the pink window of Fig. 4.

1) *Distribution Module*: When a frame arrives, we generate a colored point cloud with direction  $(\mathbf{P}, \mathbf{C}, \mathbf{D})$ . The distribution module feeds the points to their corresponding regions with their position. Together with the feed data, to ensure equal training of each region, our distribution module assigns iterations to less trained models, while setting zero iterations to more trained models.

Since the distribution module and agents run in different threads, the data distribution is sufficient for real-time data streaming while the training of the color models is optimized independently in the background.

2) *Asynchronous Neural Agents*: When training data of region  $k$  is passed to agent  $k$ , it is appended to the data memory stack. Each agent maintains its own thread to independently train the color field model  $\theta$  with memory data. As described in Fig. 4 (b), a thread of NA maintains the optimization of the neural model. The newly distributed data for an agent is appended into its own memory stack. Meanwhile, left iterations  $a$  is modified for more iterations (in the experiments  $a$  is set large enough such that it does not block the whole algorithm). Then a signal is given to continue training.

#### D. View Rendering

After a complete online pass of the data stream, a learned NSLF and a surface are obtained.

By rasterizing with a surface mesh and camera pose as input, we obtain from the unprojected surface point  $P$ . Rendering is then implemented by assigning points to different agents and synchronously predicting using Eq. (2).

We show in the experiments, that in addition to the real-time training, the rendering is also done in real-time.

## IV. RESULTS AND EXPERIMENTS

Our experiments are on the scenario of **Real-time Incremental** sequences, which run the online learning modules (MANA) aside from reconstructing.

#### A. Dataset

a) *ICL-NUIM*: ICL-NUIM [32] is one of the most widely used RGBD datasets for SLAM and reconstruction purposes. ICL-NUIM contains two synthetic scenes, i.e., room and office.

b) *Replica*: The Replica dataset [33] provides synthetic indoor space reconstructions that contain clean dense geometry, high resolution, and dynamic textures. It is suitable for our surface light field purpose.

#### B. Baselines

Our comparisons are mainly on *online* learning of Neural Surface Light Fields given incremental reconstruction frames in real-time. The reconstruction model that we work aside is a current SOTA, a large-scale real-time incremental reconstruction model, DiFusion [5], which is based on Neural Implicit Maps.

First, we compare *NSLF models* under our online learning framework (MANA) to demonstrate the advances of our proposed model. The baselines are the recent Deep Surface Light Field models DSLF [16], AFFM [17], and HashGrid (HG), an SLF model migrated from NeRF based on Instant-NGP [24].

Then we evaluate the model and framework *as a whole* to compare the photometric performance with the latest incremental neural implicit reconstruction SOTA, NICE-SLAM [12].

#### C. Implementation Details

The implementations of DSLF and AFFM are taken from [17]. For the HG baseline, we use a multi-resolution hash grid with *resolution*=512 to encode position, and spherical harmonics to encode direction [24]. The decoder operates on the concatenated feature using 4 layers of MLPs with *net-width*=32. The sigmoid at the end is used to normalize value in  $[0, 1]$  as normalized color. Our model is also implemented with the same multi-resolution hash grid settings. Instead of encoding directions, we learn spherical harmonics parameters and extract a value directly in that direction. Then, the extracted latent is operated on another MLPs with learned parameters at that position for color extraction. For all models, learning rates are set to  $lr = 1e - 3$  to train with the Adam optimizer.

For online learning of growing scenes, we set the region scale to  $4m$ . Our MAMA works alongside the recent 3D reconstruction models (DiFusion [5]) for reconstruction support. The main thread processes the data and passes it to reconstruction and MANA. In the MANA, the *data\_feeding* function in the main thread takes 0.01 s for each frame. The agents run asynchronously in their own threading. On the sequence, we skip and take every 20th frame into DiFusion and MANA. During the evaluation, we infer all frames.

For comparison, we use peak signal-to-noise ratio (PSNR), structural similarity index measure (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS) to evaluate the produced image quality.

Experiments run on a computer with an AMD Ryzen 9 5950X 16-Core Processor CPU and a GTX3090Ti GPU.

#### D. Online NSLFs alongside Real-time 3D Reconstruction

We implement MANA alongside Real-time 3D reconstruction for Online NSLF. We first run MANA and the reconstruction on the real-time sequence ICL-NUIM 1rkt0n. For a fair comparison, we should use the same mesh for all four models. Since ICL-NUIM does not provide a ground-truth mesh, we pre-run DiFusion to obtain the same mesh for all four NSLF models. Four NSLF models are embedded respectively into MANA. Then, we sequentially feed those frames into MANA. The real-time learning skips every 20 frames and the main thread sleeps 1 s when every 20th frame arrives. The evaluation is done on all frames.

Table I shows the photometric quantitative evaluation. DSLF and AFFM perform much worse than HG and ours. Our model shows the best performance on all metrics. We also find this in Fig. 5 that both DSLF and AFFM show blur or incorrect image prediction. While HG and ours show a more realistic quality on the painting and sofa.

We find that HG has results, very similar to ours. Therefore, we select HG and our model for further testing and investigation on the Replica dataset. For the experiment on Replica which provides a ground truth mesh, to mitigate other effects, we use the ground truth mesh for prediction. Similarly, MANA learns in real-time the sequences that skip every 20 frames and sleep 1 s then. In Table II, HG and ours also provide very similar quality.

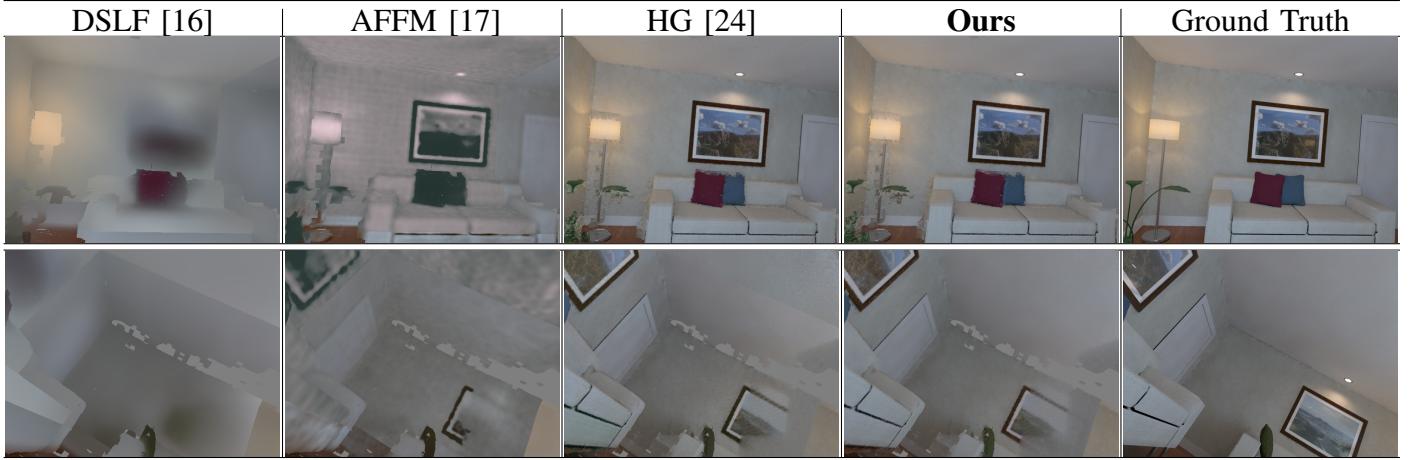


Fig. 5: Demonstration on ICL-NUIM dataset lrkt0n sequence.

TABLE I: Comparison on ICL-NUIM sequence lrkt0n.

	DSLF [16]	AFFM [17]	HG [24]	Ours
PSNR $\uparrow$	21.74	21.29	28.46	<b>28.73</b>
SSIM $\uparrow$	0.836	0.846	0.908	<b>0.915</b>
LPIPS <sub>alex</sub> $\downarrow$	0.523	0.578	0.285	<b>0.277</b>
LPIPS <sub>vgg</sub> $\downarrow$	0.616	0.652	0.409	<b>0.403</b>

However, Table I and Table II are compared on the whole sequence, which is very close to the selected trained frames. To better demonstrate the advantage of our model, we show in Fig. 6 the unseen direction of the observed surface. For HG, these surfaces show the correct color when we capture them in the trained direction. But we find that HG gives arbitrary results for the unseen direction of the observed surface. For example, in the first row, HG lost the color of the painting and the sofa cushion. In the second and third rows, HG shows arbitrary colors on the sofa and desk. In the fourth row, HG incorrectly predicts the color of the carpet. This problem happens to HG because the novel directions on those points are not learned. While our model works fine. Our model learns a sphere on each point instead of just for the observed directions.

To better support this, we further use object Chicken data from DSLF [16], which gives a dense view of the object. We perform the experiment on the Chicken test set (200 frames). We train both HG and ours using a single frame (135th) for 1000 iterations and render on all. The rendered video is demonstrated in our project page<sup>2</sup>. Where HG shows high color bias when viewing angle changes while our model gives a better prediction.

From the video, most of the surface in the data is not trained. This means that quantitative evaluation of the whole sequence can hardly find the result differences. Therefore, we compute the angle between the view directions of the inferences frame and the trained frame (135th). Then we threshold the angle to count only the frame result in a certain range. In Table IV, we use three options  $\leq 15^\circ$ ,  $\leq 30^\circ$ ,  $\leq 60^\circ$ . Where we find ours

are always better. In addition, the large angle causes inference on the unlearned surfaces and will give a more similar score. Which is revealed by the smaller gap in the larger angle threshold.

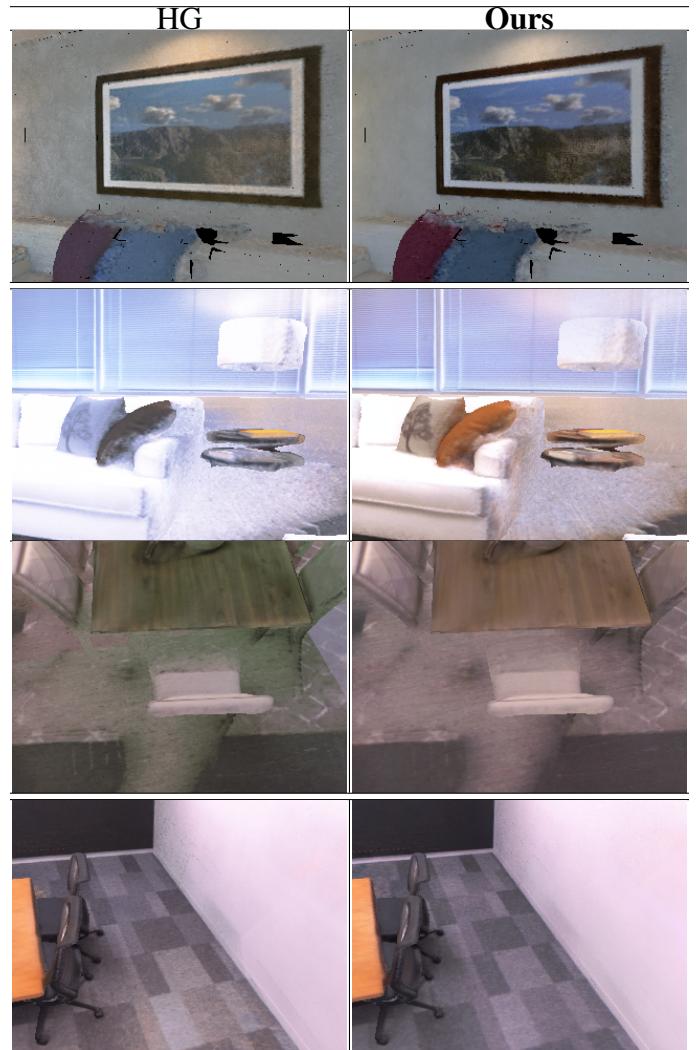
Fig. 6: Demonstration on **unseen direction of observed surface**.<sup>2</sup><https://jarrome.github.io/NSLF-OL>

TABLE II: PSNR comparison on Replica sequences.

	Ofc0	Ofc1	Ofc2	Ofc3	Ofc4	Rm0	Rm1	Rm2
HG	38.12	37.63	28.38	26.64	35.02	30.06	33.51	34.37
Ours	38.13	37.85	28.57	26.89	35.17	30.20	33.47	34.35

TABLE III: Full model comparison on Replica sequences. Header indicates scene names.

	Office0	Office1	Office2	Office3	Office4	Room0	Room1	Room2
NICE-SLAM [12]	PSNR $\uparrow$	28.38	30.68	23.90	24.88	25.18	23.46	23.97
	SSIM $\uparrow$	0.908	0.935	0.893	0.888	0.902	0.798	0.838
	LPIPS <sub>alex</sub> $\downarrow$	0.386	0.278	0.330	0.287	0.326	0.443	0.401
	LPIPS <sub>vgg</sub> $\downarrow$	0.455	0.403	0.433	0.405	0.430	0.496	0.486
DiFusion[5]+MANA	PSNR $\uparrow$	28.59	26.70	21.10	21.89	25.74	23.24	25.68
	SSIM $\uparrow$	0.913	0.879	0.863	0.847	0.893	0.816	0.883
	LPIPS <sub>alex</sub> $\downarrow$	0.371	0.497	0.362	0.368	0.401	0.371	0.308
	LPIPS <sub>vgg</sub> $\downarrow$	0.395	0.446	0.421	0.416	0.425	0.417	0.415
NICE-SLAM mesh +MANA inference	PSNR $\uparrow$	30.76	30.52	22.48	22.57	25.94	24.08	25.43
	SSIM $\uparrow$	0.942	0.927	0.891	0.872	0.911	0.819	0.879
	LPIPS <sub>alex</sub> $\downarrow$	0.214	0.216	0.265	0.263	0.304	0.345	0.310
	LPIPS <sub>vgg</sub> $\downarrow$	0.346	0.371	0.386	0.377	0.406	0.416	0.420

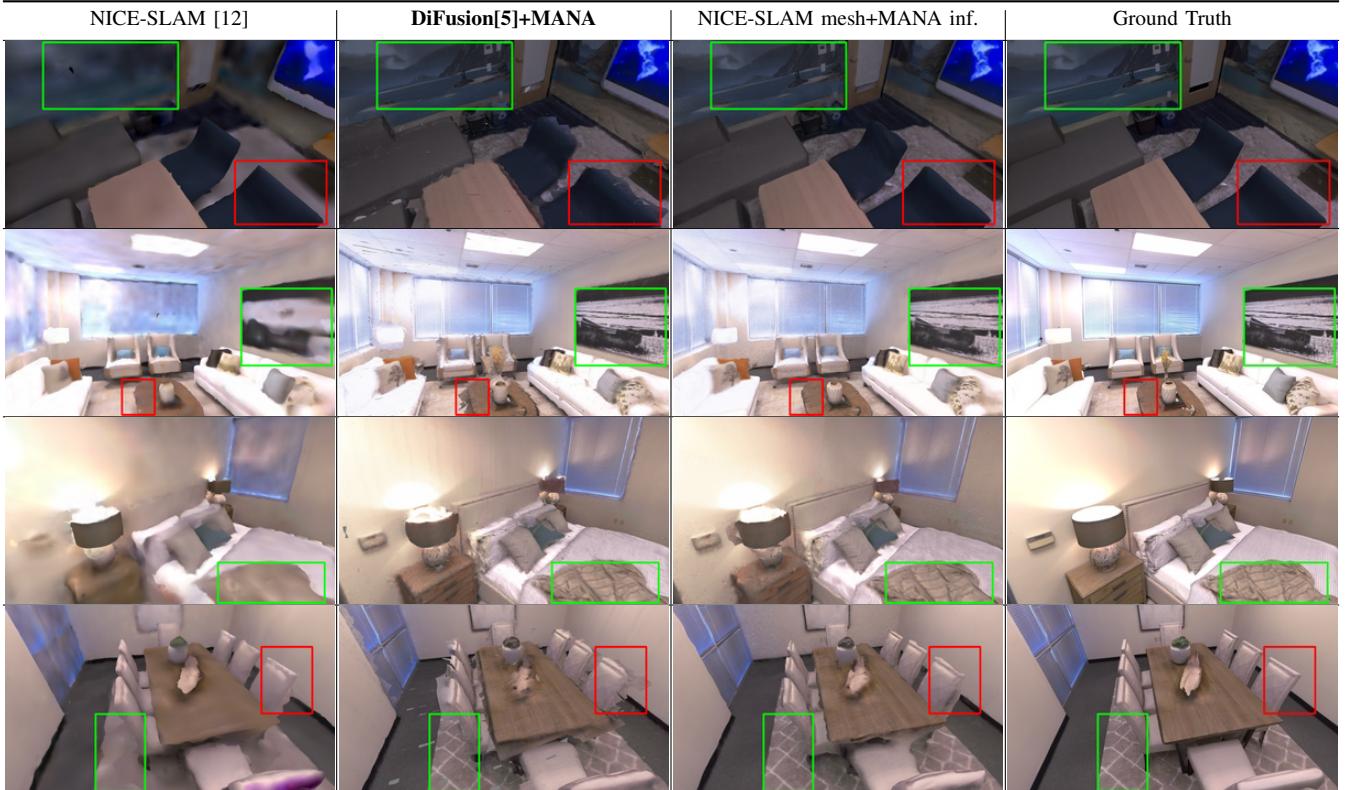


Fig. 7: Demonstration of full model comparison. The green box emphasized our better texture. With the red box we shows the mesh difference between Di-Fusion and NICE-SLAM.

TABLE IV: PSNR comparison on object sequences.

	$\leq 15^\circ$	$\leq 30^\circ$	$\leq 60^\circ$
HG	26.97	22.87	21.45
Ours	27.63	23.13	21.55

#### E. Comparing with Incremental Surface & Color Reconstruction

In the task of *real-time incremental reconstruction*, surface and color are not necessarily decoupled. For example, a recent Neural Implicit Reconstruction SOTA, NICE-SLAM provides high-quality surface and color reconstruction at the same time as online reconstruction. Therefore, we compare it to the *SOTA of real-time incremental reconstruction* to demonstrate the advantages of our setting. Table III shows the quantitative evaluation; we find that NICE-SLAM shows close performance to DiFusion combined with our MANA. However, we find

in Fig. 7 that this is not the case. Our photometric result is more realistic compared to NICE-SLAM. The reason for this phenomenon is that DiFusion’s reconstruction quality is worse than NICE-SLAM, MANA score is weakened. For example, in the first two rows, DiFusion+MANA shows a clear plot on the wall painting while NICE-SLAM shows a very blurry result. Also in the third and fourth rows, MANA shows fine texture on the quilt and carpet, while NICE-SLAM fails to find the detail. However, DiFusion produces a worse reconstruction (shown in the edge of the tables and chairs), which causes defects to the NSLF rendering.

To better demonstrate the advantage of our work along the reconstruction, we use the same learned NSLFs to infer on NICE-SLAM’s mesh. This shows a better result in most of the scenes of Table III. The specific detail is also shown in the colored region of Fig. 7.

### F. Time Efficiency

Real-time reconstruction while online learning NSLF on ICL-NUIM `lrkt0n` (1508 frames) sequence in Section IV-D takes about 28 s. This means that our online learning framework incrementally trains this fixed amount of time. Rendering a  $480 \times 640$  image in ICL-NUIM data takes on average 0.07 s. Therefore, our model easily achieves both real-time learning (training) and rendering (inference) in a similar size.

## V. CONCLUSION

In this paper, we have proposed an online learning method for neural surface light fields during real-time incremental 3D reconstruction on large scenes.

We have proposed a novel Neural Surface Light Fields model to address the challenge that in a SLAM and reconstruction scenario the captured surface directions are very limited, the learned model easily produces arbitrary predictions from unseen directions.

For online learning in growing scenes where we do not pre-know the boundaries in advance, we have designed Multiple Asynchronous Neural Agents to work alongside real-time incremental 3D reconstruction.

The performance of the proposed method has been demonstrated in our experiments. Our implementation achieves real-time learning of Neural Surface Light Fields alongside real-time incremental reconstruction.

## REFERENCES

- [1] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers, “Real-time camera tracking and 3d reconstruction using signed distance functions,” in *Robotics: Science and Systems*, vol. 2, 2013, p. 2.
- [2] M. Slavcheva, W. Kehl, N. Navab, and S. Ilic, “Sdf-2-sdf: Highly accurate 3d object reconstruction,” in *European Conference on Computer Vision*. Springer, 2016, pp. 680–696.
- [3] Y. Yuan and A. Nüchter, “Indirect point cloud registration: Aligning distance fields using a pseudo third point set,” *IEEE Robotics and Automation Letters*, 2022.
- [4] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, “Real-time 3d reconstruction at scale using voxel hashing,” *ACM Transactions on Graphics (ToG)*, vol. 32, no. 6, pp. 1–11, 2013.
- [5] J. Huang, S.-S. Huang, H. Song, and S.-M. Hu, “Di-fusion: Online implicit 3d reconstruction with deep priors,” in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8932–8941.
- [6] Y. Yuan and A. Nüchter, “An algorithm for the se (3)-transformation on neural implicit maps for remapping functions,” *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7763–7770, 2022.
- [7] A. Tewari, J. Thies, B. Mildenhall, P. Srinivasan, E. Tretschk, W. Yifan, C. Lassner, V. Sitzmann, R. Martin-Brualla, S. Lombardi *et al.*, “Advances in neural rendering,” in *Computer Graphics Forum*, vol. 41, no. 2. Wiley Online Library, 2022, pp. 703–735.
- [8] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, “Nerf: Representing scenes as neural radiance fields for view synthesis,” in *European Conf. on computer vision*. Springer, 2020, pp. 405–421.
- [9] M. Tancik, V. Casser, X. Yan, S. Pradhan, B. Mildenhall, P. P. Srinivasan, J. T. Barron, and H. Kretzschmar, “Block-nerf: Scalable large scene neural view synthesis,” in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8248–8258.
- [10] X. Zhang, S. Bi, K. Sunkavalli, H. Su, and Z. Xu, “Nerfusion: Fusing radiance fields for large-scale scene reconstruction,” in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5449–5458.
- [11] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, “imap: Implicit mapping and positioning in real-time,” in *Proc. of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6229–6238.
- [12] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys, “Nice-slam: Neural implicit scalable encoding for slam,” in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12786–12796.
- [13] K. Zhang, G. Riegler, N. Snavely, and V. Koltun, “Nerf++: Analyzing and improving neural radiance fields,” *arXiv preprint arXiv:2010.07492*, 2020.
- [14] D. N. Wood, D. I. Azuma, K. Aldinger, B. Curless, T. Duchamp, D. H. Salesin, and W. Stuetzle, “Surface light fields for 3d photography,” in *Proc. of the 27th annual Conf. on Computer graphics and interactive techniques*, 2000, pp. 287–296.
- [15] E. Miandji, J. Kronander, and J. Unger, “Learning based compression of surface light fields for real-time rendering of global illumination scenes,” in *SIGGRAPH Asia 2013 Technical Briefs*, 2013, pp. 1–4.
- [16] A. Chen, M. Wu, Y. Zhang, N. Li, J. Lu, S. Gao, and J. Yu, “Deep surface light fields,” *Proc. of the ACM on Computer Graphics and Interactive Techniques*, vol. 1, no. 1, pp. 1–17, 2018.
- [17] H. Yu, A. Chen, X. Chen, L. Xu, Z. Shao, and J. Yu, “Anisotropic fourier features for neural image-based rendering and relighting,” 2022.
- [18] C. Reiser, S. Peng, Y. Liao, and A. Geiger, “Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps,” in *Proc. of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14335–14345.
- [19] L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, and C. Theobalt, “Neural sparse voxel fields,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 15651–15663, 2020.
- [20] T. Neff, P. Stadlbauer, M. Parger, A. Kurz, J. H. Mueller, C. R. A. Chaitanya, A. Kaplanyan, and M. Steinberger, “Donerf: Towards real-time rendering of compact neural radiance fields using depth oracle networks,” in *Computer Graphics Forum*, vol. 40, no. 4. Wiley Online Library, 2021, pp. 45–59.
- [21] D. B. Lindell, J. N. Martel, and G. Wetzstein, “Autoint: Automatic integration for fast neural volume rendering,” in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14556–14565.
- [22] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, “Plenoctrees for real-time rendering of neural radiance fields,” in *Proc. of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5752–5761.
- [23] P. Hedman, P. P. Srinivasan, B. Mildenhall, J. T. Barron, and P. Debevec, “Baking neural radiance fields for real-time view synthesis,” in *Proc. of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5875–5884.
- [24] T. Müller, A. Evans, C. Schied, and A. Keller, “Instant neural graphics primitives with a multiresolution hash encoding,” *arXiv preprint arXiv:2201.05989*, 2022.
- [25] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, “Tensorf: Tensorial radiance fields,” *arXiv preprint arXiv:2203.09517*, 2022.
- [26] V. Sitzmann, S. Rezchikov, B. Freeman, J. Tenenbaum, and F. Durand, “Light field networks: Neural scene representations with single-evaluation rendering,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 19313–19325, 2021.
- [27] K. Deng, A. Liu, J.-Y. Zhu, and D. Ramanan, “Depth-supervised nerf: Fewer views and faster training for free,” *arXiv preprint arXiv:2107.02791*, 2021.
- [28] Q. Xu, Z. Xu, J. Philip, S. Bi, Z. Shu, K. Sunkavalli, and U. Neumann, “Point-nerf: Point-based neural radiance fields,” *arXiv preprint arXiv:2201.08845*, 2022.
- [29] A. Chen, Z. Xu, F. Zhao, X. Zhang, F. Xiang, J. Yu, and H. Su, “Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo,” in *Proc. of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14124–14133.
- [30] J. Zhang, Y. Yao, and L. Quan, “Learning signed distance field for multi-view surface reconstruction,” in *Proc. of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6525–6534.
- [31] D. Azinović, R. Martin-Brualla, D. B. Goldman, M. Nießner, and J. Thies, “Neural rgb-d surface reconstruction,” *arXiv preprint arXiv:2104.04532*, 2021.
- [32] A. Handa, T. Whelan, J. McDonald, and A. J. Davison, “A benchmark for rgb-d visual odometry, 3d reconstruction and slam,” in *2014 ICRA*.
- [33] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma *et al.*, “The replica dataset: A digital replica of indoor spaces,” *arXiv preprint arXiv:1906.05797*, 2019.