Urban Radiance Fields

Konstantinos Rematas[1] Andrew Liu[1] Pratul Srinivasan[1] Jonathan Barron[1]
Andrea Tagliasacchi[1,2] Thomas Funkhouser[1] Vittorio Ferrari[1]

[1] Google Research [2]University of Toronto

Abstract

The goal of this work is to perform 3D reconstruction and novel view synthesis from data captured by scanning platforms commonly deployed for world mapping in urban outdoor environments (e.g., Street View). Given a sequence of posed RGB images and lidar sweeps acquired by cameras and scanners moving through an outdoor scene, we produce a model from which 3D surfaces can be extracted and novel RGB images can be synthesized. Our approach extends Neural Radiance Fields, which has been demonstrated to synthesize realistic novel images for small scenes in controlled settings, with new methods for leveraging asynchronously captured lidar data, for addressing exposure variation between captured images, and for leveraging predicted image segmentations to supervise densities on rays pointing at the sky. Each of these three extensions provides significant performance improvements in experiments on Street View data. Our system produces state-of-the-art 3D surface reconstructions and synthesizes higher quality novel views in comparison to both traditional methods (e.g. COLMAP) and recent neural representations (e.g. Mip-NeRF).

1. Introduction

In this work we investigate neural scene representations for world mapping, with the goal of performing 3D reconstruction and novel view synthesis from data commonly captured by mapping platforms such as Street View [23]. This setting features large outdoor scenes, with many buildings and other objects, natural illumination from the sun, and is generally less controlled than previous work [42, 43]. We focus on street-level mapping: a person carrying a camera rig with a lidar sensor placed on a backpack walking through a city. The camera captures panoramas of the street scene while the lidar sensor reconstructs a 3D point cloud.

Street-level mapping is challenging for neural representations, as the area of interest covers a large area, usually hundreds of square meters. This significantly differs from previous works, which largely focus on either synthetic data [49, 56] or small regions of real scenes [8, 10, 42, 43, 64, 72]. Moreover, the scenes contain a large variety of objects, both in terms of geometry and appearance (e.g. buildings, cars, signs, trees, vegetation). The camera locations are biased towards walking patterns (e.g. walking a straight line) without focusing on any particular part of the scene. This results in parts of the scene being observed by only a small number of cameras, in contrast to other datasets [34, 43, 50, 57] which capture scenes uniformly with a large number of cameras. Furthermore, the sky is visible in most street scenes, introducing an infinitely distant element that behaves differently than the solid structures near the cameras. The images typically have highly varying exposures as the cameras use auto-exposure, and the illumination brightness varies depending on the sun's visibility and position. Combined with auto white balance, this results in the same structure having different colors when observed from different cameras. Finally, the lidar points have lower resolution in distant parts of the scene, and are even completely absent in some parts of the scene (e.g., for shiny or transparent surfaces).

Figure 1. **Overview –** Given a set of panoramas and lidar observations from an urban setting, we estimate a neural representation that can be used for novel view synthesis and accurate 3D reconstruction.

In this paper we extend the popular NeRF [42] model in three ways to tailor it to the unique features of the Street View setting and to tackle the challenges above. First, we incorporate lidar information in addition to RGB signals. By carefully fusing these two modalities, we can compensate for the sparsity of viewpoints in such large scale and complex scenes. We introduce a series of lidar-based losses that allow accurate surface estimation both for solid structures like buildings and for volumetric formations such as

trees/vegetation. Second, we automatically segment sky pixels and define a separate dome-like structure to provide a well-defined supervision signal for camera rays pointing at the sky. Third, our model automatically compensates for varying exposure by estimating an affine color transformation for each camera.

During experiments with real world data from Street View [23], we find that these three NeRF extensions significantly improve over the state-of-the-art both in the quality of synthesized novel views (+19% PSNR over [39]) and 3D surface reconstructions (+0.35 F-score over [30]). We encourage the reader to view the supplementary material for more results and animated visualizations.

2. Related Works

Novel View Synthesis. The 3D reconstruction of urban environments has been studied for decades [45]. Most prior work represents the geometry of a city with raw point clouds, acquired either from structure-from-motion [2] or lidar sensors [26], and provide only a sampled, partial representation from which it is difficult to render high quality novel views. Traditional surface reconstruction methods aggregate the raw data into explicit 3D scene representations, such as textured meshes [52] or primitive shapes [14]. These methods generally utilize hand-crafted reconstruction algorithms that work best for scenes with large, diffuse surfaces — a property that does not hold for most urban environments. Others reconstruct volumetric representations, such as voxels [55], octrees [65], or multi-plane images [18, 58, 76]. However, these approaches usually have limited resolution or suffer due to the large storage requirements of discretized volumes.

NeRF. Neural Radiance Fields represent a scene with a multilayer perceptron (MLP) that maps a 3D position and direction to a density and radiance that can be used to synthesize arbitrary novel views with volumetric rendering [42]. Typically this representation is trained per scene with a loss measuring photometric consistency with respect to a collection of posed RGB images. If the input images are dense and diverse enough, the scene is small enough, the camera poses are accurate enough, the camera exposure parameters are constant, and the scene is static, the original NeRF model can synthesize remarkably detailed and accurate novel views.

NeRF in vitro. Many researchers have investigated extensions to NeRF to overcome some of its limitations [15]. Mip-NeRF [4] proposed a scale-aware scene representation based on conical frustrums instead of rays to compensate for blurring and aliasing artifacts. NSVF [38] reduced the rendering time of radiance fields using an octree representation. Another line of research focused on radiance field estimation from single images [28, 51, 72]. Radiance fields are also used for surface extraction, either using an SDF representation [66, 70, 71] or solid surfaces [46]. However, most of this work has been demonstrated only for input images that are synthetic or captured in a laboratory setting ("in vitro") with complete control over lighting, viewpoint, and scene composition, and thus cannot be used directly in real-world applications where sensors move along constrained trajectories.

NeRF in situ. Some research has been directed towards using NeRF-like models for 3D reconstruction and novel view synthesis from images captured in natural environments ("in situ") [19, 35, 60, 69, 75]. NeRF++ [73] investigates the parameterization of unbounded scenes. Other works have addressed short video inputs [19] or monocular input [35]. Methods like [30, 68] use the 3D points from SfM to guide the training of the radiance field. In work more similar to ours, IMAP [60] performs real-time SLAM from RGB-D images captured with a hand-held camera moving through indoor scenes from the Replica Dataset [59]. The work of Azinović *et al*. [3] performs surface reconstruction of indoor environments, also using images from an RGB-D camera. ObjectNeRF [69] and SemanticNeRF [75] perform novel view synthesis and semantic segmentation from RGB-D videos of the ScanNet dataset [13]. These systems have been demonstrated for RGB-D data captured in *indoor* scenes, which do not exhibit most of the issues we address.

NeRF for world mapping. There has been very little work on using NeRF in outdoor mapping applications. Neural Scene Graphs [48] consider novel view synthesis from images provided with the KITTI Dataset [21], and NeRF in the Wild (NeRF-W) [39] does the same for internet photo collections. Neither system leverages lidar data, which is available in most outdoor mapping platforms [6, 7, 20, 22, 25, 32, 36, 61, 63] nor do they attempt to extract 3D surface reconstructions. In addition to this new functionality, in comparison to NeRF-W we also provide improved methods for handling exposure variations and the challenge posed by the sky.

2.1. Review of Neural Radiance Fields

Neural radiance fields fit a coordinate-based neural network with parameters $\boldsymbol{\theta}$ to describe a volumetric scene from a set of posed images $\{I_i\}_{i=1}^{N}$; i.e. with known intrinsic and extrinsic calibration. To render an image, NeRF uses ray marching to sample the volumetric radiance field and composites the sampled density and color to render the incoming radiance of a particular ray, and supervises the training of $\boldsymbol{\theta}$ by an L2 photometric reconstruction loss:

$$\mathcal{L}_{\text{rgb}}(\boldsymbol{\theta}) = \sum_i \mathbb{E}_{\mathbf{r}\sim I_i}\left[\left\|\boldsymbol{C}(\mathbf{r}) - \boldsymbol{C}_i^{\text{gt}}(\mathbf{r})\right\|_2^2\right] \qquad (1)$$

where $\boldsymbol{C}_i^{\text{gt}}(\mathbf{r})$ is the ground truth color of ray $\mathbf{r}$ passing through a pixel in image i, and the color $\boldsymbol{C}(\mathbf{r})$ is computed

by integrating the weighted volumetric radiance within the ray's near and far bounds t_n and t_f:

$$\boldsymbol{C}(\mathbf{r}) = \int_{t_n}^{t_f} w(t) \cdot \underbrace{\mathbf{c}(t)}_{\text{radiance}} \, dt \tag{2}$$

and $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ represents a ray with camera origin $\mathbf{o}$ oriented as $\mathbf{d}$, with volume rendering integration weights:

$$w(t) = \underbrace{\exp\left(-\int_{t_n}^{t} \sigma(s)\, ds\right)}_{\text{visibility of } \mathbf{r}(t) \text{ from } \mathbf{o}} \cdot \underbrace{\sigma(t)}_{\text{density at } \mathbf{r}(t)} \tag{3}$$

while the intermediate features $\mathbf{z}(t)$, the volumetric density $\sigma(t)$ and view-dependent radiance fields $\mathbf{c}(t)$ are stored within the parameters $\boldsymbol{\theta}$ of fully connected neural networks:

$$\mathbf{z}(t) = \mathbf{z}(\mathbf{r}(t); \boldsymbol{\theta}) : \mathbb{R}^3 \rightarrow \mathbb{R}^z \tag{4}$$

$$\sigma(t) = \sigma(\mathbf{z}(t); \boldsymbol{\theta}) : \mathbb{R}^z \rightarrow \mathbb{R}^+ \tag{5}$$

$$\mathbf{c}(t) = \mathbf{c}(\mathbf{z}(t), \mathbf{d}; \boldsymbol{\theta}) : \mathbb{R}^z \times \mathbb{R}^3 \rightarrow \mathbb{R}^3 \tag{6}$$

The discretization of the integrals in (2) and (3) follows [44]. As the scenes used in this work are observed at different distances, we use the integrated positional encoding proposed in mip-NeRF [4] for 3D points $\mathbf{r}(t)$ and the original positional encoding [42] for the viewing direction $\mathbf{d}$.

3. Data

This paper investigates how to reconstruct 3D surfaces and synthesize novel views of urban spaces from data commonly collected for autonomous driving and world mapping applications. Though many suitable data sources are available, we focus our experiments on Trekker data from Street View [23], which was acquired from Google with permission via personal communication.

Street View data is particularly interesting because it has been captured for large parts of the world, and thus provides opportunities for visualization and geometry analysis applications at scale. However, Street View differs from other 3D scene reconstruction datasets such as Phototourism [57] in several crucial ways. First, the number of images captured for a particular scene is significantly smaller than those found for popular landmarks. This results in limited diversity of viewpoints. Second these panoramic captures are often accompanied by lidar sensors which provide accurate, but sparse, depth information.

Image Data. Street View imagery is collected by multiple fisheye cameras attached to a *trekker* capturing rig. Each camera is calibrated with estimated intrinsic parameters and poses relative to the trekker. Images are collected from each camera at approximately 2Hz as the trekker moves through the world. Images are posed automatically within a global coordinate system using structure-from-motion and GPS information, allowing us to assemble camera rays with origin $\mathbf{o}$ and direction $\mathbf{d}$ corresponding to each pixel.

Real world urban scenes have moving objects whose positions change as images are captured over time (pedestrians, cars, bicyclists, etc). If unaddressed, these objects can result in trained NeRFs that produce ghosting and blurring. Because dynamics are often tied to semantics, we run a pre-trained semantic segmentation model [9] on every image, and then mask pixels of people, which are the most prominent moving category.

Lidar Data. In addition to imaging sensors, the trekker contains time-of-flight VLP16 lidar sensors which actively emit light to measure distances to surfaces. Unlike the imaging data which represents dense samples of incoming light, the lidar data is a swept sequence of timestamped 3D line segments represented by an origin and termination position. A single lidar segment indicates that during the timestamp, the space traversed by an emitted ray did not intersect an opaque surface. We make a simplifying assumption that most surfaces detected by lidar are stationary like buildings and roads, so we can ignore the timestamp information and assume that empty space is empty throughout the entire capture. This allows us to model lidar rays similar to camera rays, with origin $\mathbf{o}_\ell$, direction $\mathbf{d}_\ell$, and termination distances z_ℓ.

4. Method

We define a Urban Radiance Field (URF) with scene-level neural network parameters $\boldsymbol{\theta}$ as well as per-image exposure parameters $\{\beta_i\}$. Given the image and lidar data for a scene, we optimize a URF by minimizing the following loss:

$$\underset{\boldsymbol{\theta}, \{\beta_i\}}{\arg\min} \; \underbrace{\mathcal{L}_{\text{rgb}}(\boldsymbol{\theta}, \{\beta_i\}) + \mathcal{L}_{\text{seg}}(\boldsymbol{\theta})}_{Sec.\ 4.1} + \underbrace{\mathcal{L}_{\text{depth}}(\boldsymbol{\theta}) + \mathcal{L}_{\text{sight}}(\boldsymbol{\theta})}_{Sec.\ 4.2}$$

4.1. Photometric-based Losses

The photometric loss term is similar to the original NeRF equation (1), but ours also depends on estimated per-image exposure parameters $\{\beta_i\}$:

$$\mathcal{L}_{\text{rgb}}(\boldsymbol{\theta}, \{\beta_i\}) = \sum_i \mathbb{E}_{\mathbf{r} \sim I_i} \left[||\boldsymbol{C}(\mathbf{r}; \beta_i) - \boldsymbol{C}_i^{\text{gt}}(\mathbf{r})||_2^2 \right] \tag{7}$$

We modify the volume rendering equation in two ways, each described in its corresponding sub-section:

$$\boldsymbol{C}(\mathbf{r}; \beta_i) = \int_{t_n}^{t_f} w(t) \cdot \underbrace{\Gamma(\beta_i)}_{\text{Sec. B.2}} \cdot \mathbf{c}(t)\, dt + \underbrace{\mathbf{c}_{\text{sky}}(\mathbf{d})}_{\text{Sec. 4.1.2}} \tag{8}$$

4.1.1 Exposure compensation

Images acquired by mapping systems are usually captured with auto white balance and auto exposure which complicates the computation of $\mathcal{L}_{\text{rgb}}$ in (1). Previous work addresses

this issue using latent codes, learned separately for each image, that map image-independent scene radiance to an image-dependent radiance [3, 40, 53]. One shortcoming with such an approach is that modeling exposure variations with a per-image latent code is overparameterized as it allows the latent codes to compensate for non-exposure related errors. Instead, in (8) we perform an affine mapping of the radiance predicted by the shared network where the affine transformation is a 3x3 matrix decoded from the per-image latent code $\beta_i \in \mathbb{R}^B$:

$$\Gamma(\beta_i) = \Gamma(\beta_i; \boldsymbol{\theta}) : \mathbb{R}^B \rightarrow \mathbb{R}^{3\times 3} \tag{9}$$

This mapping models white balance and exposure variations with a more restrictive function, and thus is less likely to cause unwanted entanglement when the scene radiance parameters $\boldsymbol{\theta}$ and the exposure mappings β are optimized jointly.

4.1.2 Sky modeling

Outdoor scenes contain sky regions where rays never intersect any opaque surfaces, and thus the NeRF model gets a weak supervisory signal in those regions. To address this issue, our rendering model includes a spherical radiance (environment) map represented as a coordinate-based neural network, similar to the radiance map used in GANcraft [24]

$$\mathbf{c}_{\text{sky}}(\mathbf{d}) = \mathbf{c}_{\text{sky}}(\mathbf{d}; \boldsymbol{\theta}) : \mathbb{R}^3 \rightarrow \mathbb{R}^3 \tag{10}$$

to provide a direction-dependent background color for those regions. To modulate which rays utilize the environment map, we run a pre-trained semantic segmentation model for each image to detect pixels likely to be sky: $\mathcal{S}_i{=}\mathcal{S}(I_i)$, where $\mathcal{S}_i(\mathbf{r}){=}1$ if the ray $\mathbf{r}$ goes through a sky pixel in image i. We then use the sky mask to define an additional loss that encourages at all point samples along rays through sky pixels to have zero density:

$$\mathcal{L}_{\text{seg}}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{r}\sim I_i}\left[\mathcal{S}_i(\mathbf{r}) \int_{t_n}^{t_f} w(t)^2 \, dt\right] \tag{11}$$

Note that whenever $\mathcal{S}_i(\mathbf{r}){=}1$, this will force the $\mathbf{c}_{\text{sky}}$ to explain the pixel for ray $\mathbf{r}$ in (8).

4.2. Lidar losses

Since lidar data is available in our data, we use it to supervise training of the model. We are given a collection of L lidar samples $\mathcal{D}{=}\{(\mathbf{o}_\ell, \mathbf{d}_\ell, z_\ell)_{\ell=1}^L\}$, each corresponding to a ray $\mathbf{r}(z){=}\mathbf{o}_\ell + z\mathbf{d}_\ell$, and the associated 3D measurement $\mathbf{p}_\ell{=}\mathbf{r}(z_\ell)$. Inspired by classical works in 3D reconstruction [27], we break the losses into two different types: ① supervising the expected *depth* value, and ② supervising the free space along the *line-of-sight* from the lidar sensor to the observed position.

Expected depth. We start by supervising the expected depth $\hat{z}$ from a volumetric rendering process (i.e. optical depth [30]) to match the depth of the lidar measurement:

$$\mathcal{L}_{\text{depth}}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{r}\sim\mathcal{D}}\left[(\hat{z} - z)^2\right] \;\; \hat{z} = \int_{t_n}^{t_f} w(t) \cdot t \, dt \tag{12}$$

Line-of-sight priors. For points that are observed by a lidar sensor, a reasonable assumption is that a measured point $\mathbf{p}$ corresponds to a location on a non-transparent surface, and that atmospheric media *does not* contribute to the color measured w.r.t. a lidar ray $\mathbf{r}_\ell = \mathbf{r}(z_\ell)$. Hence, we expect that the radiance is concentrated at a *single* point along the ray, and therefore that a *single* point is responsible for the observed color. In other words, with reference to (2):

$$\boldsymbol{C}(\mathbf{r}_\ell) \equiv \mathbf{c}(\mathbf{r}_\ell) \quad \text{iff} \quad w(t) = \delta(t) \tag{13}$$

where $\delta(.)$ is the continuous Dirac function. We can convert this constraint via the penalty method into a loss:

$$\mathcal{L}_{\text{sight}}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{r}\sim\mathcal{D}}\left[\int_{t_n}^{t_f} \left(w(t) - \delta(z)\right)^2 \, dt\right] \tag{14}$$

and to make this numerically tractable, we can replace the Dirac with a kernel $\mathcal{K}_\epsilon(x)$ that integrates to one (i.e. a distribution) that has a bounded domain parameterized by ϵ. We choose $\mathcal{K}_\epsilon(x){=}\mathcal{N}(0, (\epsilon/3)^2)$, with $\mathcal{N}$ being a truncated Gaussian, and then split the ray integral into three intervals with three corresponding losses:

$$\mathcal{L}_{\text{sight}}(\boldsymbol{\theta}) = \underbrace{\mathcal{L}_{\text{empty}}(\boldsymbol{\theta})}_{t\in[t_n, z-\epsilon]} + \underbrace{\mathcal{L}_{\text{near}}(\boldsymbol{\theta})}_{t\in[z-\epsilon, z+\epsilon]} + \underbrace{\mathcal{L}_{\text{dist}}(\boldsymbol{\theta})}_{t\in[z+\epsilon, t_f]} \tag{15}$$

The second term in the breakdown above will be:

$$\mathcal{L}_{\text{near}}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{r}\sim\mathcal{D}}\left[\int_{z-\epsilon}^{z+\epsilon} \left(w(t) - \mathcal{K}_\epsilon(t - z)\right)^2 \, dt\right] \tag{16}$$

which encourages the representation to increase volumetric density in the neighborhood of $\mathbf{p}$, thereby allowing training to converge more quickly. Note that, as $\mathcal{K}_\epsilon(x)$ has bounded support in $[z - \epsilon, z + \epsilon]$, the first term can be simplified to:

$$\mathcal{L}_{\text{empty}}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{r}\sim\mathcal{D}}\left[\int_{t_n}^{z-\epsilon} w(t)^2 \, dt\right] \tag{17}$$

which requires that the portion of space between the ray origin and the lidar point $\mathbf{p}$ (i.e. the line-of-sight) does not contain any 3D surface. This line of sight information has been a key ingredient in "volume carving" techniques [12, 27, 41].The last term has a similar form:

$$\mathcal{L}_{\text{dist}}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{r}\sim\mathcal{D}}\left[\int_{z+\epsilon}^{t_f} w(t)^2 \, dt\right], \tag{18}$$

however, because this term's only purpose is to ensure that $w(t)$ sums to one, and because NeRF's volume rendering equation only requires that $w(t)$ sums to no more than one, this term can be safely dropped during training. Note that the choice of a smooth kernel $\mathcal{K}_\epsilon(x)$ is critical, as it guarantees continuity across the transition between losses at $z - \epsilon$. Finally, selecting a suitable ϵ plays an important role in the reconstruction accuracy. We discovered that employing a small ϵ hinders performance, especially in the early training phases, and note that a similar behavior has also been observed in somewhat related methods that anneal the bandwidth of importance sampling over time [47]. In our network, we adopt an exponential decay strategy for ϵ, and ablate this decision in supplementary material.

5. Experimental Evaluation

We ran a series of experiments to evaluate our model and to test whether the proposed new ideas enable more accurate renderings of novel views and improve the quality of 3D geometric reconstructions.

5.1. Evaluation Protocol

The Street View dataset. We collected 10 scenes from cities around the globe covering six continents. Each scene corresponds to a trekker capture of approximately 20 panoramas (each containing 7 images) and 6 million lidar points on average. Each scene covers hundreds of square meters and represents a different urban environment. For the quantitative analysis we report average metrics over the scenes *Taipei, Zurich, New York, Rome*. We split this overall dataset in two different ways in a training and a test set, giving rise to the two experimental settings below.

Setting 1: Held-out Viewpoints. We split each scene into train and test based on the camera locations. We randomly select 20% simultaneous image captures from our trekker rig and use them as test views. As lidar sensors operate in a continuous fashion, we select all lidar rays whose origins are close to a test camera's location as the lidar test set.

Setting 2: Held-out Buildings. We also want to evaluate how well our model reconstructs entire 3D surfaces for which we do not have any lidar. To simulate this, we manually select a building and remove all lidar rays terminating on its surface; these removed rays form the test set. We use the remaining lidar rays and all images as the training set.

5.2. Baseline Methods

We perform comparisons with the following baselines [1]. For each of them, we adjust the model parameters (number of rays, samples, etc) to be comparable:

[1]We experimented also with NerfingMVS [68], but the results were much lower than the other methods.

	Lidar	PSNR ↑	SSIM ↑	LPIPS ↓
NeRF [42]	☐	14.791	0.477	0.569
NeRF-W [40]	☐	17.156	0.455	0.620
Mip-NeRF [4]	☐	16.987	0.516	0.458
DS-NeRF [30]	☑	15.178	0.500	0.537
Ours w/o lidar	☐	19.638	0.541	0.447
Ours	☑	**20.421**	**0.563**	**0.409**

Table 1. **Novel view synthesis** – We report standard image rendering metrics on test views of selected scenes.

- **NeRF [42]** – We use the JAX [5] version, which is a superior re-implementation [16] of the original NeRF paper. This method operates on images only.
- **Mip-NeRF [4]** – an extension of NeRF that uses integrated positional encoding. It also operates only on images and is the method that we build upon.
- **DS-NeRF [30]** – this paper used 3D keypoints from an SfM reconstruction to supervise the NeRF density; here we adjust it to use lidar points (the same we also use).
- **NeRF-W [39]** – NeRF in the Wild has shown impressive results on outdoor scenes and can handle images with different exposures. It operates on images only.

5.3. Novel View Synthesis Results

We first consider novel view synthesis by training and then rendering novel views in the *Held-out Viewpoints* setting (Tab. 1). We evaluate the rendered test views using three standard metrics: PSNR, SSIM, and LPIPS [74]. Similar to the protocol of [39], we evaluate on the right part of the image, as the left is used for test-time optimization of the exposure latent codes for our method and NeRF-W.

Starting from the base model Mip-NeRF, we achieve significant improvements by adding our exposure and sky modeling (second-last row). Additionally, including lidar information improves the renderings even further (last row). The methods we compare against are challenged by Street View data due to its sparsity in terms of viewpoints and exposure variations between images. We outperform all of them, including NeRF-W which was designed for outdoor scenes, and DS-NeRF, which exploits lidar.

In Fig. 2, we show rendered results on test views from various models. Mip-NeRF suffers from its inability to handle exposure variations, resulting in floating artifacts that attempt to explain the differences in exposure between training views. Next we find that our full model shows significantly sharper images when using lidar due to geometrically accurate surface placements and suppression of erroneous, semi-transparent density "floaters". The improvement is more visible in the distant areas like the arcade in Rome and the covered sidewalk in Taipei.

Figure 2. **Qualitative novel view synthesis –** We visualize the output of our model against the ground truth and different methods. Our full model is able to generate more accurate renderings that do not suffer from exposure artifacts and floating elements.

5.4. 3D Reconstruction Results

Next we evaluate the quality of the recovered 3D scene structure, both in terms of depth estimates and point clouds.

Depth estimates. As shown in Fig. 3, our full model is able to use the sparse lidar supervision to reconstruct significantly finer depth detail compared to using only the dense pixel supervision. In particular, we reconstruct crisp depth values even for some surfaces that are difficult to capture, like cars and window frames. Note that the lidar depthmaps are estimated using depth-aware splatting and the missing regions are due to the lidar scanning pattern. For quantitative evaluation, we use the set of test lidar rays' origin $\mathbf{o}_\ell$ and direction $\mathbf{d}_\ell$ to cast rays and ask the model to estimate the expected termination distance $\hat{z}$ by sampling its volumetric density function and accumulating its transmittance, similar to Eq. (12). We compare this model estimate to the ground truth termination distance z and report the average error in meters. We also report accuracy as the number of test rays estimated within 0.1 meters of their ground truth (Acc@0.1m). Looking at the results in Tab. 2, we find that our model outperforms all baselines on both metrics (by $\sim 3\times$).

Point clouds. We generate 3D point clouds directly from the ray parameters and depth estimates. Given a ray origin $\mathbf{o}_\ell$, direction $\mathbf{d}_\ell$ and ground truth termination distance z, the corresponding 3D point is $\mathbf{p}_\ell = \mathbf{o}_\ell + z\mathbf{d}_\ell$. By iterating over all the test lidar rays we estimate the ground truth point cloud of the scene. We do the same for the estimated depth $\hat{z}$, resulting in the predicted point cloud. We compare the two clouds using Chamfer Distance and F-score (threshold $= 0.1$ meters). As Tab. 2 shows, our model has the best performance across all metrics in both held-out settings. The difference is large even over DS-NeRF, that uses lidar, indicating that both our exposure/sky modeling and our combination of losses are important to achieve high accuracy.

Mesh reconstruction. Here we use our full model to generate dense point clouds by casting one ray for each pixel in each training camera, and estimating depth for it as described above. For comparison, we obtain a point cloud with COLMAP [54] estimated by running MVS using the camera parameters provided with the dataset. We also compare to the point cloud defined by the lidar points (the training points that we also feed to our method). For each method, we then reconstruct 3D meshes using Poisson Surface Reconstruction [31].

Fig. 4 shows the meshes derived from our point clouds, from COLMAP and from lidar. Our method is able to estimate accurately the underlying geometry, whereas COLMAP loses fine details and lidar produces artifacts due to limited sampling resolution. Our method also provides denser coverage than the raw lidar, since we also use images. These provide higher resolution observations in some regions (e.g., the cars in the image on the left) and broader coverage (e.g., the missing region of the building on the right), as the scanning pattern of the lidar is far narrower than the image panoramas. Fig. 5 shows more mesh reconstructions from our method. We can accurately reconstruct fine details in areas hundreds of square meters large.

5.5. Ablation Studies

Effects of individual components. Tab. 3 studies the effect of each model component. As we add exposure compensation and sky modeling we see consistent improvements in all 3D reconstruction metrics (second and third rows). When incorporating lidar, the best performing setup is when using all proposed losses at the same time (Sec. 4.2). The strongest contribution comes from the near-surface loss, as when its not activated the performance drops considerably. The empty-space loss, which is primarily used to suppress floating semi-transparent density ("floaters"), harms performance when used without the near-surface loss. This indicates that our decaying margin strategy is suitable way to gather the benefits of both losses. We further analyze this behaviour in the supplementary material.

Effect of exposure handling. Finally, we investigate the effect of our affine color transformation. An alternative is to provide the exposure code β_i directly as an input to the network, as done in NeRF-W [39] and [3]. However, in

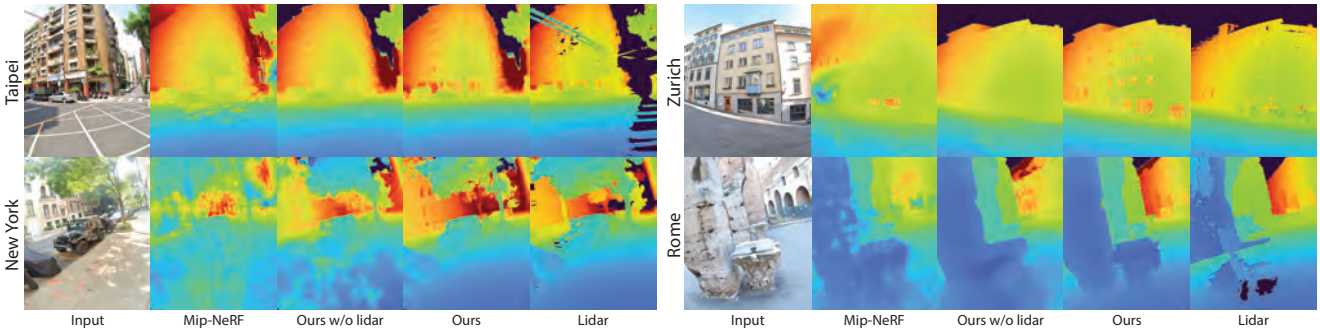

Figure 3. **Qualitative depthmaps –** We visualize the expected depth for our model against other methods and variations. Our full model is able to estimate precisely the extent of the scene, including thin structures such as tree trunks, window frames, etc.

		Held-out Viewpoints				Held-out Building			
	Lidar	Avg Error (m) ↓	Acc↑	CD↓	F↑	Avg Error (m) ↓	Acc↑	CD↓	F↑
NeRF [42]	☐	1.582	0.264	3.045	0.528	1.423	0.274	2.857	0.535
NeRF-W [40]	☐	3.663	0.144	6.165	0.372	1.348	0.207	4.054	0.552
Mip-NeRF [4]	☐	1.596	0.133	2.812	0.363	1.417	0.132	2.508	0.427
DS-NeRF [30]	☑	1.502	0.259	2.571	0.526	1.367	0.294	2.720	0.558
Ours	☑	**0.463**	**0.742**	**0.272**	**0.880**	**0.770**	**0.363**	**2.312**	**0.687**

Table 2. **Reconstruction evaluation** – We compare various NeRF based approaches in two experimental settings (*Held-out Viewpoints*, *Held-out Buildings*). We report two depth estimation metrics (Average Error and Accuracy) and two point cloud metrics (Chamfer Distance CD, and F-score). See main text for details.

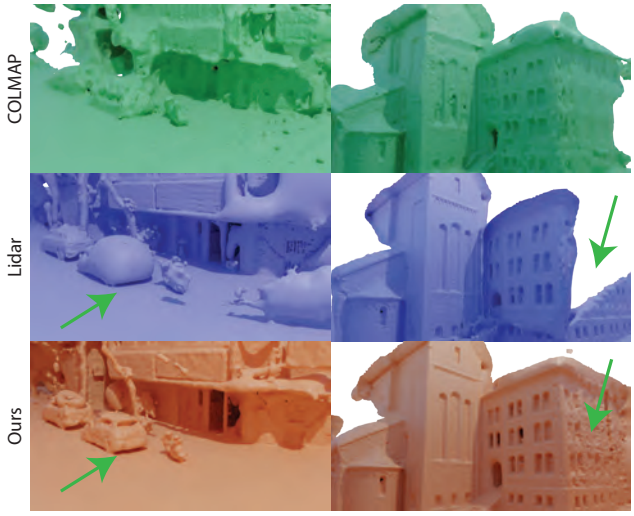

Figure 4. **Surface Reconstruction –** We show the surface reconstruction returned by different approaches. Our method is able to provide dense and accurate depth estimates, which in turn allow detailed mesh reconstructions.

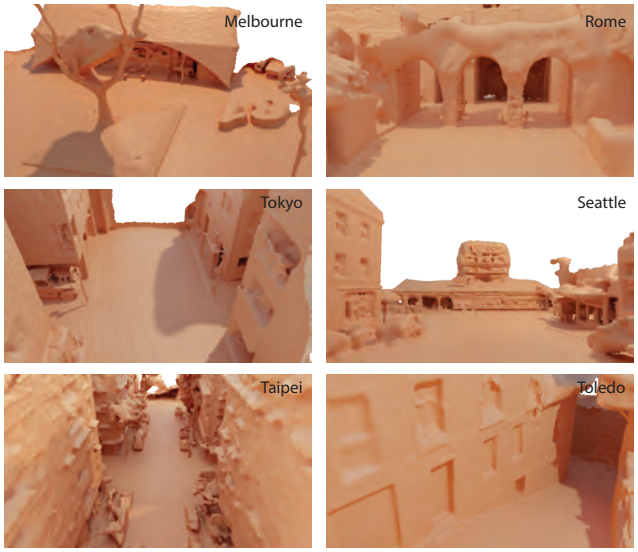

Figure 5. **Additional Reconstructions –** Visualization of extracted meshes for several large scale urban scenes.

this way the exposure code can explain arbitrary appearance elements, and not necessarily those due to exposure/white balance. Fig. 6 illustrates the appearance changes when modifying the latent codes on the *Rome* scene. For our affine model this translate into rendering the same structure with different color tones, while the direct approach generates visible artifacts. These affect also 3D reconstruction performance: our affine model results in F-score of 0.47 vs 0.36 for the direct approach, These artifacts can be reduced by limiting the power of the latent codes to affect rendering, *e.g.* by providing β_i to later layer of the network, or reducing their dimensionality (similar to observations in [73] regarding the viewing direction $\mathbf{d}$). See the supplementary material for more details.

Γ	$\mathcal{L}_{\text{seg}}$	$\mathcal{L}_{\text{depth}}$	$\mathcal{L}_{\text{empty}}$	$\mathcal{L}_{\text{near}}$	Avg↓	Acc↑	CD↓	F↑
☐	☐	☐	☐	☐	1.596	0.133	2.812	0.363
☑	☐	☐	☐	☐	1.226	0.184	1.771	0.424
☑	☑	☐	☐	☐	1.109	0.233	1.190	0.471
☑	☑	☑	☐	☐	0.811	0.284	0.782	0.545
☑	☑	☑	☑	☐	1.136	0.093	0.536	0.306
☑	☑	☐	☑	☑	0.633	0.736	0.726	0.878
☑	☑	☑	☑	☑	**0.463**	**0.742**	**0.272**	**0.880**

Table 3. **Ablation study** – We investigate the effect of the individual components of our model, from the image-based elements (top part) to the lidar losses (bottom part). The first row corresponds to the baseline Mip-NeRF [4] model.

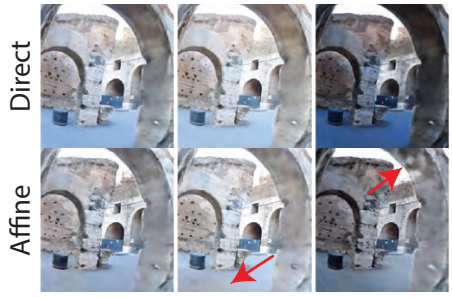

Figure 6. **Exposure modifications –** We visualize the effect of changing exposure codes using our affine color model and a direct approach. The affine model performs a global color transformation, enforcing the disentanglement between exposure codes and scene radiance.

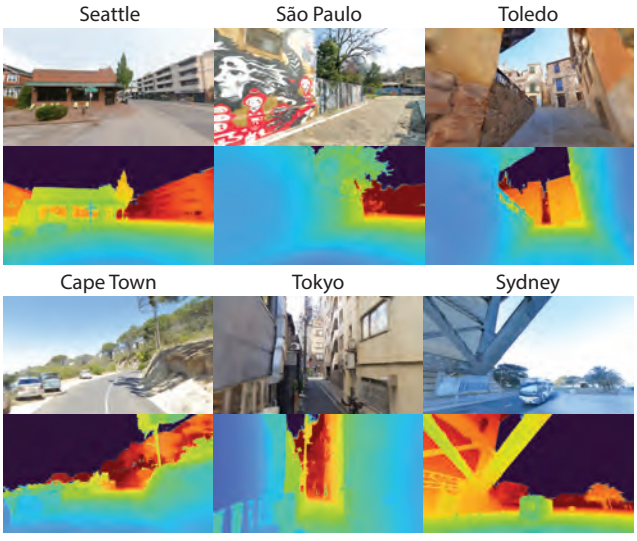

Figure 7. **Novel views –** After training our model for a particular scene, we are able to estimate novel views for a new camera with different intrinsics and trajectory path than those used for training.

Figure 8. **Colored mesh –** Using our model we can also accurately reconstruct a colored scene mesh.

5.6. Qualitative results

A benefits of having precise 3D reconstruction as part of a NeRF-based model is that it enables great flexibility in placing a virtual camera for novel view synthesis. In Fig. 7 we visualize rendered images and depth maps by our model from camera positions substantially different from those along the trekkers acquisition path (on average 1.7 meters away from the closest training camera).

In Sec. 5.4 we showed how we can estimate accurately the scene geometry as meshes. In Fig. 8, we illustrate how we can use our model to query the colors of the mesh vertices, resulting in precise textured meshes that are compatible with traditional rendering software. Note that the exposure is already compensated, while traditional pipelines need to perform screened Poisson image integration to compensate for the color variation [17],

5.7. Limitations

Our system has limitations. First, it assumes that good camera parameters are given through an SfM pipeline, but they are sometimes noisy in practice — a joint optimization of the camera parameters along with the network parameters may improve the reconstructions for those cases [37, 67]. Second, it has been demonstrated only for snippets of data cut out of longer scanning sequences — models learned from multiple snippets would have to be *stitched* together to produce a coherent model of the spatially continuous world. These limitations, plus failure cases and potential negative societal impacts, are discussed further in the supplemental material.

6. Conclusion

We present a system for 3D reconstruction and novel view synthesis from data captured by mobile scanning platforms in urban environments. Our approach extends recent work on Neural Radiance Fields with new ideas to leverage asynchronously captured lidar data, to account for differences in exposures between captured images, and to leverage predicted image segmentations to supervise the density of rays pointing towards the sky. Experimental results on Street View data demonstrate that each of these three ideas significantly improves performance on its own, and that they combine to produce better synthesized novel views (+19% PSNR over [39]) and 3D surface reconstructions (+0.35 F-score over [30]). We hope this paper inspires future work to take further steps towards deploying coordinate-based neural networks in outdoor mapping applications.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 12

[2] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building rome in a day. *Communications of the ACM*, 2011. 2

[3] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural rgb-d surface reconstruction. *arXiv*, 2021. 2, 4, 6, 12

[4] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. *ICCV*, 2021. 2, 3, 5, 7, 8, 12

[5] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. 5

[6] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. *arXiv:1903.11027*, 2019. 2

[7] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. *CVPR*, 2019. 2

[8] Anpei Chen and Zexiang Xu. MVSNeRF: Fast Generalizable Radiance Field Reconstruction From Multi-View Stereo. In *ICCV*, October 2021. 1

[9] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *ECCV*, 2018. 3

[10] Julian Chibane, Aayush Bansal, Verica Lazova, and Gerard Pons-Moll. Stereo radiance fields (srf): Learning view synthesis for sparse views of novel scenes. *CVPR*, 2021. 1

[11] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 13

[12] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. *SIGGRAPH*, 1996. 4

[13] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. *CVPR*, 2017. 2

[14] Paul E Debevec, Camillo J Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. *SIGGRAPH*, 1996. 2

[15] Frank Dellaert and Yen-Chen Lin. Neural volume rendering: Nerf and beyond. *CoRR*, abs/2101.05204, 2021. 2

[16] Boyang Deng, Jonathan T. Barron, and Pratul P. Srinivasan. JaxNeRF: an efficient JAX implementation of NeRF, 2020. 5

[17] Arnaud Dessein, William AP Smith, Richard C Wilson, and Edwin R Hancock. Seamless texture stitching on a 3d mesh by poisson blending in patches. *ICIP*, 2014. 8

[18] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. Deepstereo: Learning to predict new views from the world's imagery. *CVPR*, 2016. 2

[19] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. *CVPR*, 2021. 2

[20] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 2013. 2

[21] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. *CVPR*, 2012. 2

[22] Jakob Geyer, Yohannes Kassahun, Mentar Mahmudi, Xavier Ricou, Rupesh Durgesh, Andrew S. Chung, Lorenz Hauswald, Viet Hoang Pham, Maximilian Mühlegg, Sebastian Dorn, Tiffany Fernandez, Martin Jänicke, Sudesh Mirashi, Chiragkumar Savani, Martin Sturm, Oleksandr Vorobiov, Martin Oelker, Sebastian Garreis, and Peter Schuberth. A2D2: Audi Autonomous Driving Dataset, 2020. 2

[23] Google. Street view, 2007. www.google.com/streetview/. 1, 2, 3

[24] Zekun Hao, Arun Mallya, Serge Belongie, and Ming-Yu Liu. GANcraft: Unsupervised 3D Neural Rendering of Minecraft Worlds. *ICCV*, 2021. 4

[25] Xinyu Huang, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin, and Ruigang Yang. The apolloscape dataset for autonomous driving. *CVPR Workshops*, 2018. 2

[26] Veli Ilci and Charles Toth. High definition 3d map creation using gnss/imu/lidar sensor integration to support autonomous vehicle navigation. *Sensors*, 2020. 2

[27] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. *UIST*, 2011. 4

[28] Wongbong Jang and Lourdes Agapito. CodeNeRF: Disentangled Neural Radiance Fields for Object Categories. In *ICCV*, October 2021. 2

[29] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert

Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. In-datacenter performance analysis of a tensor processing unit. *SIGARCH Comput. Archit. News*, 45(2), 2017. 12

[30] Jun-Yan Zhu Kangle Deng, Andrew Liu and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. *arXiv:2107.02791*, 2021. 2, 4, 5, 7, 8

[31] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. *SGP*, 2006. 6

[32] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet. Lyft level 5 perception dataset 2020. https://level5.lyft.com/dataset/, 2019. 2

[33] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 12

[34] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM TOG*, 2017. 1

[35] Jiaxin Li and Zijian Feng. MINE: Towards Continuous Depth MPI With NeRF for Novel View Synthesis. In *ICCV*, October 2021. 2

[36] Yiyi Liao, Jun Xie, and Andreas Geiger. KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *arXiv.org*, 2109.13410, 2021. 2

[37] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. BARF: Bundle-Adjusting Neural Radiance Fields. In *ICCV*, October 2021. 8

[38] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Sua, and Christian Theobalt. Neural Sparse Voxel Fields. In *Adv. Neural Inform. Process. Syst.*, 2020. 2

[39] Ricardo Martin-Brualla, Noha Radwan, Mehdi Sajjadi, Jonathan Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*, 2021. 2, 5, 6, 8, 13

[40] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. *CVPR*, 2021. 4, 5, 7

[41] Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven J Gortler, and Leonard McMillan. Image-based visual hulls. *SIGGRAPH*, 2000. 4

[42] Ben Mildenhall, Pratul Srinivasan, Matthew Tancik, Jonathan Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*, pages 405–421. Springer, 2020. 1, 2, 3, 5, 7, 12

[43] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019. 1

[44] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *ECCV*, 2020. 3

[45] Przemyslaw Musialski, Peter Wonka, Daniel G Aliaga, Michael Wimmer, Luc Van Gool, and Werner Purgathofer. A survey of urban reconstruction. *Computer graphics forum*, 2013. 2

[46] Michael Oechsle, Songyou Peng, and Andreas Geiger. UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction. In *ICCV*, October 2021. 2

[47] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. *ICCV*, 2021. 5

[48] Julian Ost, Fahim Mannan, Nils Thürey, Julian Knodt, and Felix Heide. Neural Scene Graphs for Dynamic Scenes. In *CVPR*, pages 2856–2865, June 2021. 2

[49] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. *CVPR*, 2019. 1

[50] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. *International Conference on Computer Vision*, 2021. 1

[51] Konstantinos Rematas, Ricardo Martin-Brualla, and Vittorio Ferrari. ShaRF: Shape-conditioned Radiance Fields from a Single View. In *ICML*, 2021. 2

[52] Andrea Romanoni, Daniele Fiorenti, and Matteo Matteucci. Mesh-based 3d textured urban mapping. *IROS*, 2017. 2

[53] Darius Rückert, Linus Franke, and Marc Stamminger. Adop: Approximate differentiable one-pixel point rendering. *arXiv*, 2021. 4

[54] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. *ECCV*, 2016. 6

[55] Inwook Shim, Yungeun Choe, and Myung Jin Chung. 3d mapping in urban environment using geometric featured voxel. *URAI*, 2011. 2

[56] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *NeurIPS*, 2019. 1

[57] Noah Snavely, Steven Seitz, and Richard Szeliski. Photo tourism: exploring photo collections. *ACM TOG*, 2006. 1, 3

[58] Pratul P Srinivasan, Richard Tucker, Jonathan T Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. *CVPR*, 2019. 2

[59] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl

Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv:1906.05797*, 2019. 2

[60] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew Davison. iMAP: Implicit Mapping and Positioning in Real-Time. In *ICCV*, October 2021. 2

[61] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset, 2019. 2

[62] three.js. three.js, 2015. 13

[63] Guofeng Tong, Yong Li, Dong Chen, Qi Sun, Wei Cao, and Guiqiu Xiang. Cspc-dataset: New lidar point cloud dataset and benchmark for large-scale scene semantic segmentation. *IEEE Access*, 2020. 2

[64] Alex Trevithick and Bo Yang. GRF: Learning a General Radiance Field for 3D Scene Representation and Rendering. In *ICCV*, October 2021. 1

[65] Linh Truong-Hong and Debra F Laefer. Octree-based, automatic building facade generation from lidar data. *Computer-Aided Design*, 2014. 2

[66] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*, 2021. 2

[67] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. NeRF–: Neural Radiance Fields Without Known Camera Parameters. *https://arxiv.org/abs/2102.07064*, 2021. 8

[68] Yi Wei. NerfingMVS: Guided Optimization of Neural Radiance Fields for Indoor Multi-View Stereo. In *ICCV*, October 2021. 2, 5

[69] Bangbang Yang. Learning Object-Compositional Neural Radiance Field for Editable Scene Rendering. In *ICCV*, October 2021. 2

[70] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *arXiv:2106.12052*, 2021. 2

[71] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronan Basri, and Yaron Lipman. Multiview Neural Surface Reconstruction by Disentangling Geometry and Appearance. In *Adv. Neural Inform. Process. Syst.*, 2020. 2

[72] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural Radiance Fields from One or Few Images. In *CVPR*, 2021. 1, 2

[73] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv:2010.07492*, 2020. 2, 7, 13

[74] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. *CVPR*, 2018. 5

[75] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew Davison. In-Place Scene Labelling and Understanding with Implicit Scene Representation. In *ICCV*, October 2021. 2

[76] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo Magnification: Learning View Synthesis using Multiplane Images. In *Siggraph*, 2018. 2

URF: Urban Radiance Fields

Supplementary Material

The following supplemental material contains additional implementation details, ablation studies and qualitative results.

A. Additional Implementation Details

A.1. Network architecture

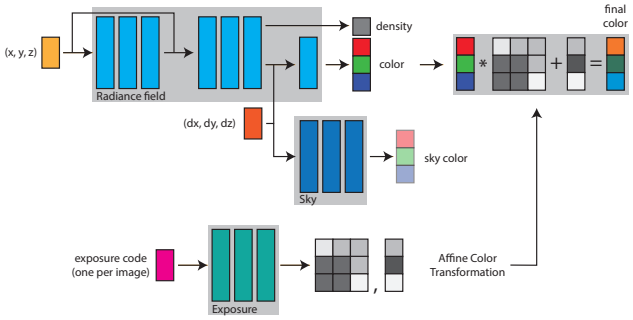

Figure 9. **Network architecture**

Our network architecture is illustrated in Fig. 9 and has three components. The first component is the neural radiance field network, which is similar to the original NeRF [42]. It consists of a series of fully connected layers of width 256 that take as input the 3D location of a point x, y, z and the viewing direction dx, dy, dz and output the RGB color and the density at that point. The second component is the sky network, which takes as an input the direction dx, dy, dz of a ray pointing at a sky point, and outputs its color. Finally, the third component is an exposure compensation network that takes as an input an exposure latent code and estimates the affine transformation to be applied to the color values output by the radiance field network. There is a different affine transformation per image. This compensates for the different exposures across input images. All three network are trained jointly so that the final colors output by the model will match the pixel colors in the input images.

A.2. Training Protocol

We train a separate network for each baseline model (Sec. 5.2) and each variant of our model, applies to each scene. Every network is trained with the same protocol, detailed here. We use a TPU v2 architecture with 128 cores [29] using Tensorflow 2 [1]. We used the Adam optimizer [33] with a learning rate scheduler that included two stages. The warm up stage lasts 50 epochs, with the learning rate starting at 0.0005 and growing linearly until 0.005. After warm up, the main stage lasts 500 epochs, with the learning rate starting at 0.005 and then decaying exponentially with exponent 0.98. The ray batch size was set to 2048 per core and the total training time was about one day per network.

For ray sampling, we use a stratified strategy where the intervals are evenly spaced in log scale, and we sample 1024 samples per ray. We did not perform hierarchical sampling. Each batch contains rays randomly sampled from all images (and similarly for the lidar points)

The 3D location of a point is described using integrated positional encoding [4] with $L = 10$ frequencies. For the viewing direction we use the original positional encoding representation with 4 frequencies.

B. Additional Ablation Studies

B.1. Effect of margin ϵ

	Avg Error↓	CD↓	Acc↑	F↑
Fixed	1.007	2.195	0.814	0.871
Stepwise	0.776	1.818	0.849	0.905
Linear	0.238	0.508	0.903	0.961
Exponential	0.249	0.863	0.901	0.966

Table 4. **Margin decay** (ϵ) – We evaluate different decay strategies for the margin ϵ during training in the Rome scene. The margin controls the contribution of the lidar losses $\mathcal{L}_{\text{near}}$ and $\mathcal{L}_{\text{empty}}$. Having a fixed margin results in lower performance, while gradually decreasing it performs the best.

As we observed in Sec. 5.5 of the main paper, the empty-space loss can actually decrease 3D reconstruction performance as it introduces a strong preference for empty space. Using the near-surface loss, which is complementary to empty-space by construction, alleviates this effect. In Tab. 4 we vary the margin ϵ in Eq. (16) and Eq. (17) during training using different strategies: Fixed: keep a constant margin throughout training (as in [3]); Stepwise: start with a large margin (thus only the near-surface loss is activated) and after $N = 50$ epochs the margin suddenly becomes small; Linear/exponential: gradually reduce the margin from large to small with a linear or an exponential schedule. For all methods, the smallest value ϵ was set to 20cm. The linear and exponential strategies perform similarly and much better than the fixed and stepwise ones, indicating that the empty-space loss is best applied after the training process manages to infer a good initial version of the scene structure.

B.2. Effect of exposure handling

In Tab. 5 we expand the ablation experiment in Sec. 5.5 of the main paper, comparing our affine transformation model

	D	Avg Error↓	CD↓	Acc↑	F↑
Direct	48	1.071	1.28	0.159	0.362
Affine		0.98	1.007	0.253	0.47
Direct	4	1.049	2.062	0.247	0.477
Affine		0.885	1.564	0.262	0.524

Table 5. **Exposure handling** – We compare our affine transformation model with the direct input of the exposure code to the network. Using an explicit color transformation for the different exposures results in better reconstruction.

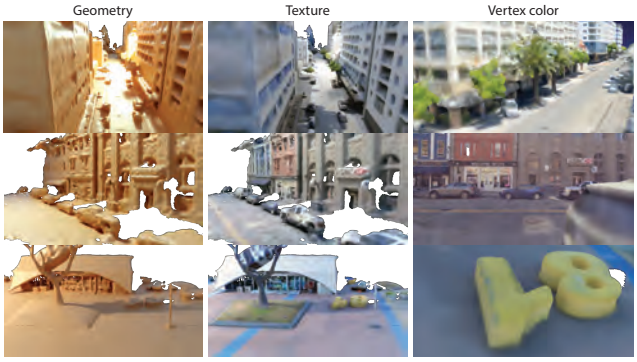

Figure 10. **Rendering colored meshes –** After extracting a color mesh using our model, we can render its geometry and texture in a 3D environment, or render the vertex colors in real time on a browser.

versus directly providing the exposure latent code to the network for the Rome scene. We experiment with two different dimensions for the latent codes, $D = 48$ as in NeRF-W [39] and a much smaller one $D = 4$. As Tab. 5 shows, our affine transformation approach performs better in all 3D reconstruction metrics, for both values of D. We also observe that both the affine and the direct approach perform better when the exposure latent code has smaller dimensionality, thus reduced capacity. For the direct approach this is in accordance to the observations in NeRF++ [73] about the viewing direction: implicit regularization (limiting the capacity) of the latent code can increase the performance. For the affine case, this indicates that there exist a compact latent space that can describe the color transformations appearing in the dataset and it is easier to learn.

C. Additional Qualitative Results

In Fig. 10 we show more visualizations of extracted colored meshes for different scenes. The color of every vertex is estimated by querying the radiance field network in that particular location. This representation can be used in common 3D editing software such as Blender [11] (first and second column in Fig. 10) and it allows for real time rendering on the browser, *e.g.* using ThreeJS [62] (third column in Fig. 10). Note that this way of rendering is different than the volumetric rendering in NeRF models, which is continuous and incorporates implicitly the view dependent appearance changes. Finally, we present additional results for novel view synthesis in Fig. 11.

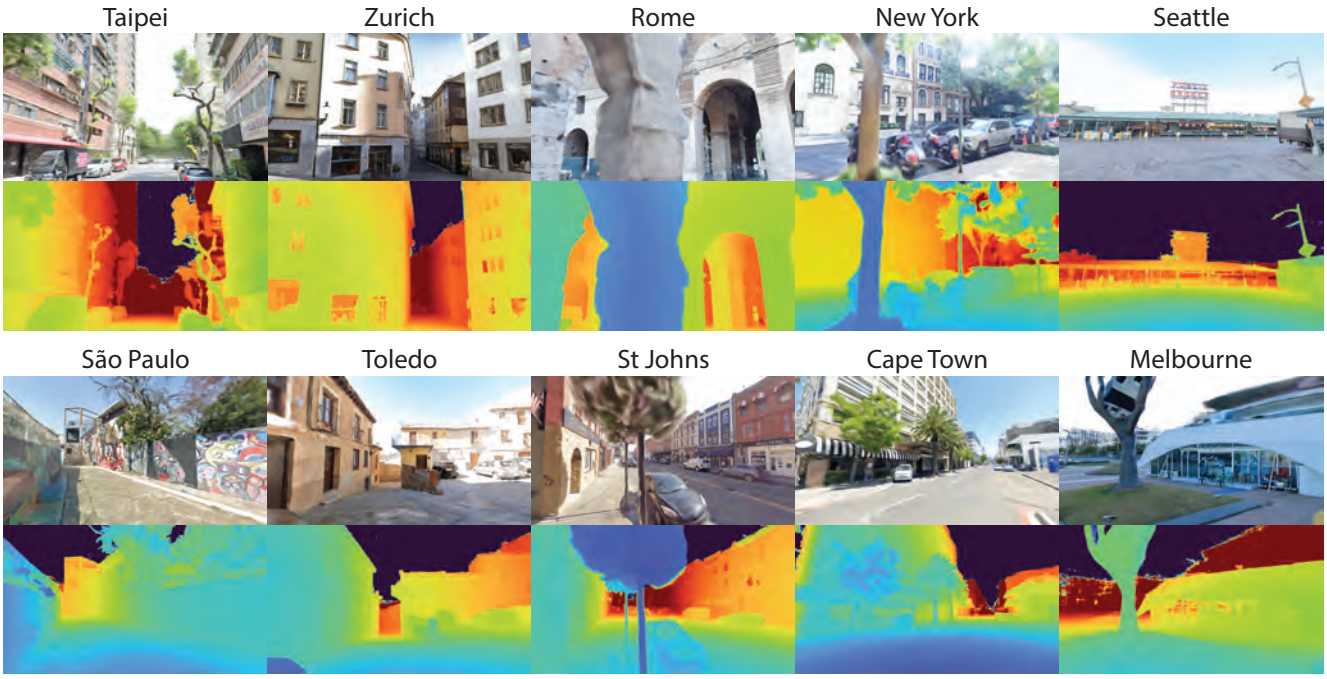

Figure 11. **Novel views**