arXiv:2301.07668v2 [cs.CV] 28 Feb 2023

Behind the Scenes: Density Fields for Single View Reconstruction

Felix Wimbauer[1] Nan Yang[1] Christian Rupprecht[2] Daniel Cremers[1]
[1]Technical University of Munich [2]University of Oxford
{felix.wimbauer, nan.yang, cremers}@tum.de chrisr@robots.ox.ac.uk

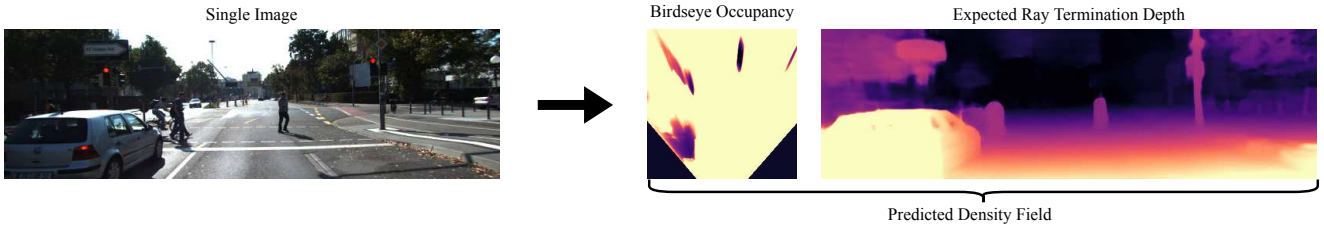

Figure 1. **Predicting a Density Field from a Single Image.** Through a novel "density field" formulation, which decouples geometry from color, architectural improvements, and a novel self-supervised training scheme, our method learns to predict a volumetric scene representation from a single image in challenging conditions. In the birdseye occupancy view you can clearly make out the different objects in true 3D, which is not possible in traditional depth prediction. Please check out our project page at fwmb.github.io/bts/.

Abstract

Inferring a meaningful geometric scene representation from a single image is a fundamental problem in computer vision. Approaches based on traditional depth map prediction can only reason about areas that are visible in the image. Currently, neural radiance fields (NeRFs) can capture true 3D including color but are too complex to be generated from a single image. As an alternative, we introduce a neural network that predicts an implicit density field from a single image. It maps every location in the frustum of the image to volumetric density. Our network can be trained through self-supervision from only video data. By not storing color in the implicit volume, but directly sampling color from the available views during training, our scene representation becomes significantly less complex compared to NeRFs, and we can train neural networks to predict it. Thus, we can apply volume rendering to perform both depth prediction and novel view synthesis. In our experiments, we show that our method is able to predict meaningful geometry for regions that are occluded in the input image. Additionally, we demonstrate the potential of our approach on three datasets for depth prediction and novel-view synthesis.

1. Introduction

The ability to infer information about the geometric structure of a scene from a single image is of high importance for a wide range of applications from robotics to augmented reality. While traditional computer vision mainly focused on reconstruction from multiple images, in the deep learning age the challenge of inferring a 3D scene from merely a single image has received renewed attention.

Traditionally, this problem has been formulated as the task of predicting per-pixel depth values (*i.e*. depth maps). One of the most influential lines of work showed that it is possible to train neural networks for accurate single-image depth prediction in a self-supervised way only from video sequences. [13–15, 28, 42, 49, 53, 54, 56] Despite these advances, depth prediction methods are *not* modeling the true 3D of the scene: they model only a *single* depth value per pixel. As a result, it is not directly possible to obtain depth values from views other than the input view without considering interpolation and occlusion. Further, the predicted geometric representation of the scenes does not allow reasoning about areas that lie *behind* another object in the image (*e.g*. a house behind a tree), inhibiting the applicability of monocular depth estimation to 3D understanding.

Due to the recent advance of 3D neural fields, the related task of novel view synthesis has also seen a lot of progress. Instead of directly reasoning about the scene geometry, the goal here is to infer a representation that allows rendering views of the scene from novel viewpoints. While geometric properties can often be inferred from the representation, they are usually only a side product and lack in quality.

Even though neural radiance field [31] based methods achieve impressive results, they require many training images per scene and do not generalize to new scenes. Efforts

have been made to condition the neural network on global or local scene features to enable generalization. However, this has only been shown to work well on simple scenes, for example scenes containing an object from a single category [41, 52]. Nevertheless, obtaining a neural radiance field from a single image has not been achieved before.

In this work, we tackle the problem of inferring a geometric representation from a single image by generalizing the depth prediction formulation to a continuous density field. Concretely, our architecture contains an encoder-decoder network that predicts a dense feature map from the input image. This feature map locally conditions a density field inside the camera frustum, which can be evaluated at any spatial point through a multi-layer perceptron (MLP). The MLP is fed with the coordinates of the point and the feature sampled from the predicted feature map by reprojecting points into the camera view. To train our method, we rely on simple image reconstruction losses.

Our method achieves robust generalization and accurate geometry prediction even in very challenging outdoor scenes through three key novelties:

1. Color sampling. When performing volume rendering, we sample color values directly from the input frames through reprojection instead of using the MLP to predict color values. We find that only predicting density drastically reduces the complexity of the function the network has to learn. Further, it forces the model to adhere to the multi-view consistency assumption during training, leading to more accurate geometry predictions.

2. Shifting capacity to the feature extractor. In many previous works, an *encoder* extracts image features to condition local appearance, while a high-capacity MLP is expected to generalize to multiple scenes. However, on complex and diverse datasets, the training signal is too noisy for the MLP to learn meaningful priors. To enable robust training, we significantly reduce the capacity of the MLP and use a more powerful *encoder-decoder* that can capture the entire scene in the extracted features. The MLP then only evaluates those features locally.

3. *Behind the Scenes* loss formulation. The continuous nature of the density field and color sampling allow us to reconstruct a novel view from the colors of any frame, not just the input frame. By applying a reconstruction loss between two frames that both observe areas occluded in the input frame, we train our model to predict meaningful geometry *everywhere* in the camera frustum, not just the visible areas.

We demonstrate the potential of our new approach in a number of experiments on different datasets regarding the aspects of capturing true 3D, depth estimation, and novel view synthesis. On KITTI [11] and KITTI-360 [25], we show both qualitatively and quantitatively that our model can indeed capture true 3D, and that our model achieves state-of-the-art depth estimation accuracy. On RealEstate10K [43] and KITTI, we achieve competitive novel view synthesis results, even though our method is purely geometry-based. Further, we perform thorough ablation studies to highlight the impact of our design choices.

2. Related Work

In the following, we review the most relevant works that are related to our proposed method.

2.1. Single-Image Depth Prediction

One of the predominant formulations to capture the geometric structure of a scene from a single image is predicting a per-pixel depth map. Learning-based methods have proven able to overcome the inherent ambiguities of this task by correlating contextual cues extracted from the image with certain depth values. One of the most common ways to train a method for single-image depth prediction is to immediately regress the per-pixel ground-truth depth values [9,26]. Later approaches supplemented the fully-supervised training with reconstruction losses [20, 51], or specialise the architecture and loss formulation [1, 10, 21, 23, 24]. To overcome the need for ground-truth depth annotations, several papers focused on relying exclusively on reconstruction losses to train prediction networks. Both temporal video frames [56] and stereo frames [12], as well as combinations of both [13, 54] can be used as the reconstruction target. Different followup works refine the architecture and loss [14, 15, 28, 42, 49, 53]. [55] first predicts a discrete density volume as an intermediate step, from which depth maps can be rendered from different views. While they use this density volume for regularization, their focus is on improving depth prediction and their method does not demonstrate the ability to learn true 3D.

2.2. Neural Radiance Fields

Many works have investigated alternative approaches to representing scenes captured from a single or multiple images, oftentimes with the goal of novel view synthesis. Recently, [31] proposed to represent scenes as neural radiance fields (NeRFs). In NeRFs, a multi-layer perceptron (MLP) is optimized per scene to map spatial coordinates to color (appearance) and density (geometry) values. By evaluating the optimized MLP along rays and then integrating the color over the densities, novel views can be rendered under the volume rendering formulation [29]. Training data consists of a large number of images of the same scene from different viewpoints with poses computed from traditional SFM and SLAM methods [4, 38, 39]. The training goal is to reconstruct these images as accurately as possible. NeRF's impressive performance inspired many follow-up works, which improve different parts of the architecture [2, 3, 6, 17, 19, 34, 37].

In the traditional NeRF formulation, an entire scene is captured in a single, large MLP. Thus, the trained network cannot be adapted to a different setting or used for other scenes. Further, the MLP has to have a high capacity, resulting in slow inference. Several methods propose to condition such MLPs on feature grids or voxels [5,27,30,33,36,41,45,52]. Through this, the MLP needs to store less information and can be simplified, resulting in significantly faster inference [5, 27, 33, 45]. Additionally, this allows for some generalization to new scenes [32, 41, 52]. However, generalization is mostly limited to a single object category, or simple synthetic data, where the scenes differ mostly in local details. In contrast, our proposed method can generalize to highly complex outdoor scenes.

2.3. Single Image Novel View Synthesis

While traditional NeRF-based methods achieve impressive performance when provided with enough images per scene, they do not work when there is only a single image of a scene available. In recent years, a number of methods specialized for novel-view synthesis (NVS) from a single image emerged. They often incorporating ideas from both depth prediction and neural radiance fields.

Several methods [7, 8, 47] predict layered depth images (LDI) [40] for rendering. Later approaches [44, 46] directly produce a multiplane image (MPI) [57] (multiple layers at certain depths with color and alpha values) [22] predicts a generalized multiplane image. Instead of directly outputting the discrete layers, the architecture's decoder receives a variable depth value, for which it outputs the layer. In [50], a network predicts both a per-pixel depth and feature map. Novel viewes are synthesized through neural rendering based on the reprojected features. While these methods achieve impressive NVS results, the quality of predicted geometry usually falls short. Some methods even predict novel views without any geometric representation [58].

3. Method

In the following, we describe a neural network architecture that predicts the geometric structure of a scene from a single image $\mathbf{I}_\text{I}$, as shown in Fig. 2. We first cover how we represent a scene as a continuous density field, and then propose a training scheme that allows our architecture to learn geometry even in occluded areas.

3.1. Notation

Let $\mathbf{I}_\text{I} \in [0,1]^{3\times H\times W} = (\mathbb{R}^3)^\Omega$ be the input image, defined on a lattice $\Omega = \{1,\dots,H\} \times \{1,\dots,W\}$. $T_\text{I} \in \mathbb{R}^{4\times4}$ and $K_\text{I} \in \mathbb{R}^{3\times4}$ are the corresponding world-to-camera pose matrix and projection matrix, respectively. During training, we have available an additional set of $N = \{1,2,\dots,n\}$ frames $\mathbf{I}_k, k \in N$ with corresponding world-to-camera pose and projection matrices $T_k, K_k, k \in N$. When assuming homogeneous coordinates, a point $\mathbf{x} \in \mathbb{R}^3$ in world coordinates can be projected onto the image plane of frame k with the following operation: $\pi_k(\mathbf{x}) = K_k T_k \mathbf{x}$

3.2. Predicting a Density Field

We represent the geometric structure of a scene as a function, which maps scene coordinates $\mathbf{x}$ to volume density σ. We term this function "density field". Inference happens in two steps. From the input image $\mathbf{I}_\text{I}$, an encoder-decoder network first predicts a pixel-aligned feature map $\mathbf{F} \in (\mathbb{R}^C)^\Omega$. The idea behind this is that every feature $f_\mathbf{u} = \mathbf{F}(\mathbf{u})$ at pixel location $\mathbf{u} \in \Omega$ captures the distribution of local geometry along the ray from the camera origin through the pixel at $\mathbf{u}$. It also means that the density field is designed to lie inside the camera frustum. For points outside of this frustum, we extrapolate features from within the frustum.

To obtain a density value at a 3D coordinate $\mathbf{x}$, we first project $\mathbf{x}$ onto the input image $\mathbf{u}'_\text{I} = \pi_\text{I}(\mathbf{x})$ and bilinearly sample the feature $f_{\mathbf{u}'} = \mathbf{F}(\mathbf{u}')$ at that position. This feature $f_{\mathbf{u}'}$, along with the positional encoding [31] $\gamma(d)$ of the distance d between $\mathbf{x}$ and the camera origin, and the positional encoding $\gamma(\mathbf{u}'_\text{I})$ of the pixel, is then passed to a multi-layer perceptron (MLP) ϕ. During training, ϕ and $\mathbf{F}$ learn to describe the density of the scene given the input view. We can interpret the feature representation $f_{\mathbf{u}'}$ as a descriptor of the density along a ray through the camera center and pixel $\mathbf{u}'$. In turn, ϕ acts as a decoder, that given $f_{\mathbf{u}'}$ and a distance to the camera, predicts the density at the 3D location $\mathbf{x}$.

$$\sigma_\mathbf{x} = \phi(f_{\mathbf{u}'_\text{I}}, \gamma(d), \gamma(\mathbf{u}'_\text{I})) \tag{1}$$

Unlike most current works on neural fields, we *do not* use ϕ to also predict color. This drastically reduces the complexity of the distribution along a ray as density distributions tend to be simple, while color often contains complex high-frequency components. In our experiments, this makes capturing such a distribution in a single feature, so that it can be evaluated by an MLP, much more tractable.

3.3. Volume Rendering with Color Sampling

When rendering the scene from a novel viewpoint, we do not retrieve color from our scene representation directly. Instead, we sample the color for a point in 3D space from the available images. Concretely, we first project a point $\mathbf{x}$ into a frame k and then bilinearly sample the color $c_{\mathbf{x},k} = \mathbf{I}_k(\pi_k(\mathbf{x}))$.

By combining $\sigma_\mathbf{x}$ and $c_{\mathbf{x},k}$, we can perform volume rendering [18,29] to synthesise novel views. We follow the discretization strategy of other radiance field-based methods, *e.g.* [31]. To obtain the color $\hat{c}_k$ for a pixel in a novel view, we emit a ray from the camera and integrate the color along the ray over the probability of the ray ending at a certain distance. To approximate this integral, density and color are evaluated at S discrete steps $\mathbf{x}_i$ along the ray. Let δ_i be the

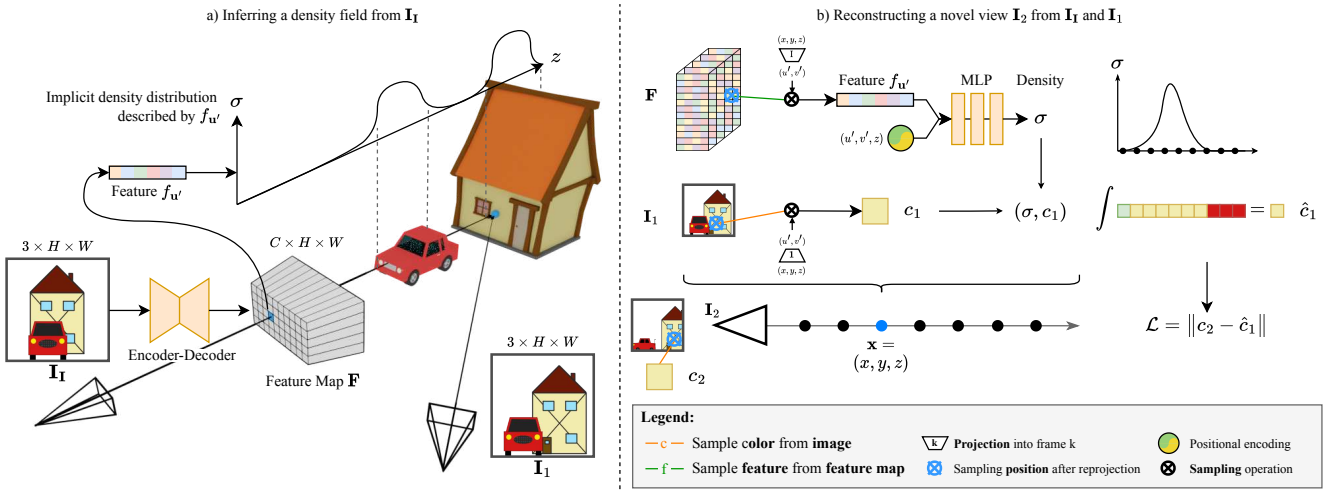

Figure 2. **Overview.** *a)* Our method first predicts a pixel-aligned feature map $\mathbf{F}$, which describes a density field, from the input image $\mathbf{I}_\mathbf{I}$. For every pixel $\mathbf{u}'$, the feature $f_{\mathbf{u}'}$ implicitly describes the density distribution along the ray from the camera origin through $\mathbf{u}'$. Crucially, this distribution can model density even in occluded regions (*e.g.* the house). *b)* To render novel views, we perform volume rendering. For any point $\mathbf{x}$, we project $\mathbf{x}$ into $\mathbf{F}$ and sample $f_{\mathbf{u}'}$. This feature is combined with positional encoding and fed into an MLP to obtain density σ. We obtain the color c by projecting $\mathbf{x}$ into one of the views, in this case $\mathbf{I}_1$, and directly sampling the image.

distance between $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$, and α_i be the probability of a ray ending between $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$. From the previous α_js, we can compute the probability T_i that the ray does not terminate before $\mathbf{x}_i$, *i.e.* the probability that $\mathbf{x}_i$ is not occluded.

$$\alpha_i = \exp(1 - \sigma_{\mathbf{x}_i} \delta_i) \qquad T_i = \prod_{j=1}^{i-1} (1 - \alpha_j) \tag{2}$$

$$\hat{c}_k = \sum_{i=1}^{S} T_i \alpha_i c_{\mathbf{x}_i, k} \tag{3}$$

Similarly, we can also retrieve the expected ray termination depth, which corresponds to the depth in a depth map. Let d_i be the distance between $\mathbf{x}_i$ and the ray origin.

$$\hat{d} = \sum_{i=1}^{S} T_i \alpha_i d_i \tag{4}$$

This rendering formulation is very flexible. We can sample the color values from any frame, and, crucially, it can be a different frame from the input frame. It is even possible to obtain multiple colors from multiple different frames for a single ray, which enables reasoning about occluded areas during training. Note that even though different frames can be used, the density is always based on features from the input image and does not change. During inference, color sampling from different frames is not necessary, everything can be done based on a single input image.

3.4. *Behind the Scenes* Loss Formulation

Our training goal is to optimize both the encoder-decoder network and ϕ to predict a density field only from the input image, such that allows reconstructing other views.

Similar to radiance fields and self-supervised depth prediction methods, we rely on an image reconstruction loss. For a single sample, we first compute the feature map $\mathbf{F}$ from $\mathbf{I}_\mathbf{I}$ and randomly partition *all* frames $\hat{N} = \{\mathbf{I}_\mathbf{I}\} \cup N$ into two sets $N_{\text{loss}}, N_{\text{render}}$. Note that the input image can end up in any of the two sets. We reconstruct the frames in N_{loss} by sampling colors from N_{render} using the camera poses and the predicted densities. The photometric consistency between the reconstructed frames and the frames in N_{loss} serves as supervision signal for the density field. In practice, we randomly sample p *patches* P_i to use patch-wise photometric measurement. For every patch P_i in N_{loss}, we obtain a reconstructed patch $\hat{P}_{i,k}$ from *every* frame $k \in N_{\text{render}}$. We aggregate the costs between P_i and every $\hat{P}_{i,k}$ by taking the per-pixel *minimum* across the different frames k, similar to [13]. The intuition behind this is that for every patch, there is a frame in N_{render}, which "sees" the same surface. Therefore, if the predicted density is correct, then it results in a very good reconstruction and a low error.

For the final loss formula, we use a combination of L1 and SSIM [48] to compute the photometric discrepancy, as well as an edge-aware smoothness term. Let $d_i^\star$ denote the inverse, mean-normalized expected ray termination depth of patch P_i. Both $\mathcal{L}_{\text{ph}}$ and $\mathcal{L}_{\text{eas}}$ are computed per (x, y) element of the patch, thus resulting in 2D loss maps. They are then aggregated when computing $\mathcal{L}$.

$$\mathcal{L}_{\text{ph}} = \min_{k \in N_{\text{render}}} \left(\lambda_{\text{L1}} \text{L1}(P_i, \hat{P}_{i,k}) + \lambda_{\text{SSIM}} \text{SSIM}(P_i, \hat{P}_{i,k}) \right) \tag{5}$$

$$\mathcal{L}_{\text{eas}} = |\delta_x d_i^\star| \, e^{-|\delta_x P_i|} + |\delta_y d_i^\star| \, e^{-|\delta_y P_i|} \tag{6}$$

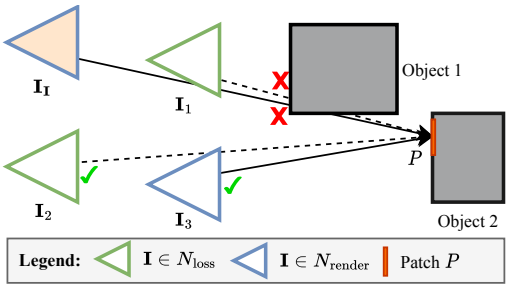

Figure 3. **Loss in Occluded Regions.** Patch P on Object 2 is occluded by Object 1 in the input frame $\mathbf{I}_\mathbf{I}$ and $\mathbf{I}_1$. In order to correctly reconstruct P in $\mathbf{I}_2$ from $\mathbf{I}_3$, the network needs to predict density for Object 2 *behind* Object 1.

$$\mathcal{L} = \sum_{i=1}^{p} \sum_{x,y \in P} \left(\mathcal{L}_{\text{ph}} + \lambda_{\text{eas}} \mathcal{L}_{\text{eas}}\right)(x, y) \tag{7}$$

Learning true 3D. Our loss formula Eq. (7) is the same as for self-supervised depth prediction methods, like [13]. The key difference, however, is that depth prediction methods can only densely reconstruct the input image, for which the per-pixel depth was predicted.

In contrast, our density field formulation allows us to reconstruct *any* frame from *any other* frame. Consider an area of the scene, which is occluded in the input $\mathbf{I}_\text{I}$, but visible in two other frames $\mathbf{I}_\text{k}, \mathbf{I}_{\text{k+1}}$, as depicted in Fig. 3: During training, we aim to reconstruct this area in $\mathbf{I}_\text{k}$. The reconstruction based on colors sampled from $\mathbf{I}_{\text{k+1}}$ will give a clear training signal to correctly predict the geometric structure of this area, even though it is occluded in $\mathbf{I}_\text{I}$. Note, that in order to learn geometry about occluded areas, we require at least **two additional** views besides the input during training, *i.e.* to look *behind the scenes*.

Handling invalid samples. While the frustums of the different views overlap for the most part, there is still a chance of a ray leaving the frustums, thus sampling invalid features, or sampling invalid colors. Such invalid rays lead to noise and instability in the training process. Therefore, we propose a policy to detect and remove invalid rays. Our intuition is that when the amount of contribution to the final aggregated color, that comes from invalidly sampled colors or features, exceeds a certain threshold τ, the ray should be discarded. Consider a ray that is evaluated at positions $\mathbf{x}_i, i \in [1, 2, \ldots, S]$ and reconstructed from frames K: $O_{i,k}, k \in \{\text{I}\} \cup K$ denotes the indicator function that $\mathbf{x}_i$ is outside the camera frustum of frame k. Note that we always sample features from the input frame. We define $\text{IV}(k)$ to be the function indicating that the rendered color based on frame k is invalid as:

$$\text{IV}(k) = \sum_{i=1}^{S} T_i \alpha_i \left(O_{i,\text{I}} \vee O_{i,k}\right) > \tau \tag{8}$$

Only if $\text{IV}(k)$ is true for *all* frames the ray was reconstructed from, we ignore the ray when computing the loss value. The reasoning behind this is that non-invalid rays will still lead to the lowest error. Therefore, the $\min$ operation in Eq. (5) will ignore the invalid rays.

3.5. Implementation Details

We implement our model in PyTorch [35] on a single Nvidia RTX A40 GPU with 48GB memory. The encoder-decoder network follows [13] using a ResNet encoder [16] and predicts feature maps with 64 channels. The MLP ϕ is made lightweight with only 2 fully connected layers and 64 hidden nodes each. We use a batch size of 16 and sample 32 patches of size 8×8 from the images for which we want to compute the reconstruction loss. Every ray is sampled at 64 locations, based on a linear spacing in inverse depth. For more details, *e.g.* exact network architecture and further hyperparameters, please refer to the supplementary material.

4. Experiments

To demonstrate the abilities and advantages of our proposed method, we conduct a wide range of experiments. First, we demonstrate that our method is uniquely able to capture a holistic geometry representation of the scene, even in areas that are occluded in the input image. Additionally, we also show the effect of different data setups on the prediction quality. Second, we show that our method, even though depth maps are only a side product of our scene representation, achieves depth accuracy on par with other state-of-the-art self-supervised methods, that are specifically designed depth prediction. Third, we demonstrate that, even though our representation is geometry-only, our method can be used to perform high-quality novel view synthesis from a single image. Finally, we conduct thorough ablation studies based on occupancy estimation and depth prediction to justify our design choices.

4.1. Data

For our experiments, we use three different datasets: KITTI [11], KITTI-360 [25] (autonomous driving), and RealEstate10K [57] (indoor). RealEstate10K only has monocular sequences, while KITTI and KITTI-360 provide stereo. KITTI-360 also contains fisheye camera frames facing left and right. For monocular data, we use three timesteps per sample, for stereo sequences (possibly with fisheye frames), we use two timesteps. The fisheye frames are offset by one second to increase the overlap of the different camera frustums.[1] Training is performed for 50 epochs on KITTI (approx. 125k steps), 25 epochs on KITTI-360 (approx. 143k steps), and 360k iterations on RealEstate10K.

[1] More details on offsets, pose data, and data splits in the supp. mat.

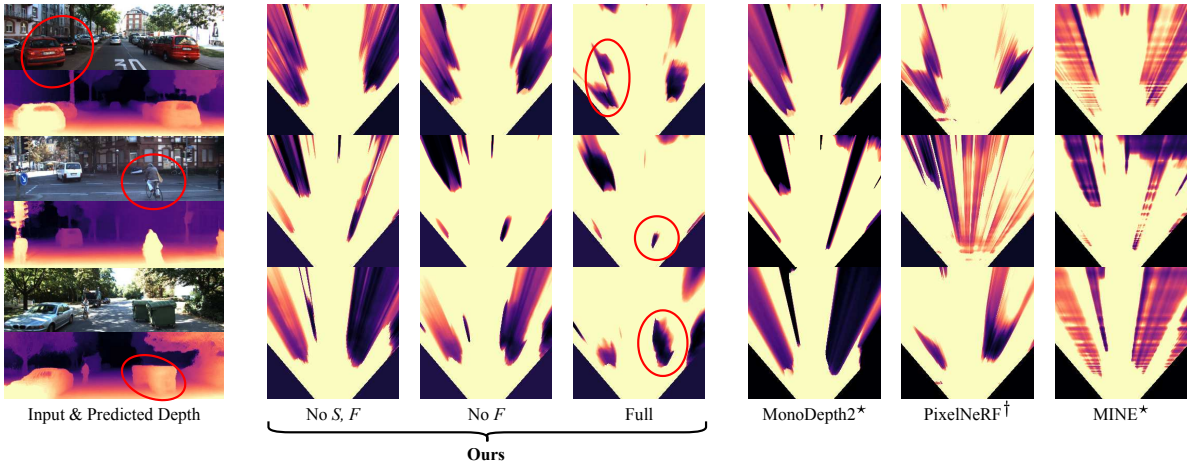

Figure 4. **Occupancy Estimation.** Top-down visualization of predicted occupancy volumes. We show an area of $x = [-9m, 9m], z = [3m, 21m]$ and $y = [0m, 1m]$ (just above the road). Our method produces an accurate volumetric reconstruction, even for occluded regions. Training with more views improves the quality. Depth prediction methods like MonoDepth2 [13] do not predict a full 3D volume. Thus, objects cast "occupancy shadows" behind them. Volumetric methods like PixelNeRF [52] and MINE [22] produce noisy predictions. Inference is from a single image. **Legend**: *S*: Stereo, *F*: Fisheye, ⋆: official checkpoint, †: trained in same setup as our Full variant.

Method	O_{acc} ↑	IE_{acc} ↑	IE_{rec} ↑
Depth† [13]	**0.95**	n/a	n/a
Depth† + *4m* [13]	0.92	0.64	0.23
PixelNeRF† [52]	0.93	0.64	<u>0.41</u>
Ours (No *S, F*)	0.93	0.69	0.10
Ours (No *F*)	<u>0.94</u>	<u>0.73</u>	0.16
Ours	**0.95**	**0.82**	**0.47**

Table 1. **3D Scene Occupancy Accuracy on KITTI-360.** We evaluate the capability of the model to predict occupancy *behind* objects in the image. Ground truth occupancy maps are computed from 20 consecutive Lidar scans per frame. Depth prediction [13] naturally has no ability to predict behind occlusions. PixelNeRF [52] can predict free space in occluded regions, but produces poor overall geometry. Our method improves when training with more views. Inference from a single image. Samples are evenly spaced in a cuboid $w = [-4m, 4m], h = [-1m, 0m], d = [3m, 20m]$ relative to the camera. **Legend**: ref. Fig. 4.

We use a resolution of 640×192 for KITTI and KITTI-360, and follow [22] in using a resolution of 384×256 for RealEstate10K.

4.2. Capturing true 3D

Evaluation of fully geometric 3D representations like density fields is difficult. Real-world datasets usually only provide ground truth data captured from a single viewpoint, *e.g.* RGB-D frames and Lidar measurements. Nevertheless, we aim to evaluate and compare this key advantages of our method both qualitatively and quantitatively. Through our proposed training scheme, our networks are able to learn to also predict meaningful geometry in occluded areas.

To overcome the lack of volumetric ground truth, we accumulate Lidar scans to build reference occupancy maps for KITTI-360. Consider a single input frame for which we want to evaluate an occupancy prediction: As KITTI-360 is a driving dataset with a forward moving camera, the consecutive Lidar scans captured a short time later measure different areas within the camera frustum. Note that these Lidar measurements can reach areas that are occluded in the input image. To determine whether a point is occupied, we check whether it is *in front* of the measured surface for any of the Lidar scans. Intuitively, every Lidar measurement "carves out" unoccupied areas in 3D space. By accumulating enough Lidar scans, we obtain a reliable occupancy measurement of the entire camera frustum. Whether a point is visible in the input frame can be checked using the Lidar scan corresponding to the input frame.[2]

For every frame, we sample points in a cuboid area in the camera frustum and compute the following metrics: 1. Occupancy accuracy (O_{Acc}), 2. Invisible and empty accuracy (IE_{Acc}), and 3. Invisible and empty recall (IE_{Rec}). O_{Acc} evaluates the occupancy predictions across the whole scene volume. IE_{Acc} and IE_{Rec} specifically evaluate invisible regions, evaluating performance beyond depth prediction.

We train a MonoDepth2 [13] model to serve as a baseline representing ordinary depth prediction methods. Here, we consider all points behind the predicted depth to be occupied. Additionally, we evaluate a version in which we consider points only up to 4 meters (average car length) behind the predicted depth as occupied. As a second baseline, we train a PixelNeRF [52] model, one of the most prominent NeRF variants that also has the ability to generalize.

To demonstrate that our loss formulation generates

[2]More details on the exact procedure and examples in the supp. mat.

Model	Volum.	Split	Abs Rel ↓	RMSE ↓	$\alpha < 1.25$ ↑
PixelNeRF [52]	✓	Eigen [9]	0.130	5.134	0.845
EPC++ [28]	✗		0.128	5.585	0.831
MonoDepth2 [13]	✗		0.106	4.750	0.874
PackNet [15]	✗		0.111	4.601	0.878
DepthHint [49]	✗		0.105	4.627	0.875
FeatDepth [42]	✗		0.099	4.427	0.889
DevNet [55]	(✓)		**0.095**	**4.365**	**0.895**
Ours	✓		0.102	4.407	0.882
MINE [22]	✓	Tuls. [47]	0.137	6.592	0.839
Ours	✓		**0.132**	**6.104**	**0.873**

Table 2. **Depth Prediction on KITTI.** While our goal is full volumetric scene understanding, we compare to state-of-the-art self-supervised depth estimation method. Our approach achieves competitive performance while clearly improving over other volumetric approaches like PixelNeRF [52] and MINE [22]. DevNet [55] performs better, but does not show any results of their volume.

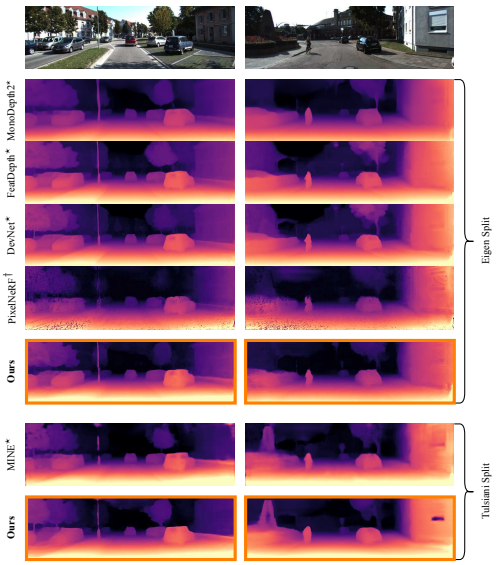

Figure 5. **Depth Prediction on KITTI.** Expected ray termination depth compared with depth prediction results of other state-of-the-art methods [13,22,42,52,55] on both the Eigen [9] and [47] split. Our predictions are very detailed and sharp, and capture the structure of the scene, even when trained on a smaller split like like Tulsiani. Visualizations for DevNet and FeatDepth are taken from [55]. **Legend**: ref. Fig. 4.

strong training signals for occluded regions, given the right data, we train our model in several different data configurations. By removing the fisheye, respectively fisheye and stereo frames, the training signal for occluded areas becomes much weaker. Tab. 1 reports the obtained results.

The depth prediction baselines achieve a strong overall accuracy, but are, by design, not able to predict meaningful free space in occluded areas. PixelNeRF can predict free space in occluded areas, but produces poor overall geometry. Our model achieves strong overall accuracy, while it is also able to recover the geometry of the occluded parts of the scene. Importantly, our model becomes better at predicting *free space in occluded areas* when training with more views, naturally providing a better training signal for occluded areas. To qualitatively visualize these results we sample the camera frustum in horizontal slices from the center of the image downwards and aggregate the density in Fig. 4. This shows the layout of the scene, similar to the birds-eye perspective but for density. In the Full variant, the strong signal lets our model learn sharp object boundaries, as can be seen for several cars in the examples. For depth prediction, all objects cast occupancy shadows along the viewing direction. PixelNeRF predicts a volumetric representation with free space in occluded regions. However, the results are noisy and geometry is inaccurate. MINE [22] also specializes on predicting a volumetric representation from a single image. However, it does not produce meaningful density prediction behind objects. Instead, similar to depth prediction, all objects cast occupancy shadows along the viewing direction.

4.3. Depth Prediction

While our method does not predict depth maps directly, they can be synthesized as a side product from our representation through the expected ray termination depth $\hat{d}$. To demonstrate that our predicted representation achieves high accuracy, we train our model on KITTI sequences and compare to both self-supervised depth prediction methods and volume reconstruction methods.

As can be seen in Tab. 2 and Fig. 5, our method performs on par with the current state-of-the-art methods for self-supervised depth prediction. Our synthesized depth maps capture finer details and contain fewer artifacts, as often seen with depth maps obtained from neural radiance field-based methods, like PixelNeRF [52] and MINE [22]. Overall, we achieve competitive performance, even though depth prediction is not the main objective of our approach.

4.4. Novel View Synthesis from a Single Image

As we obtain a volumetric representation of a scene from a single image, we are able to synthesize images from novel viewpoints by sampling color form the input image. Thus, we also evaluate novel view synthesis from a single image. To demonstrate the variability of our approach, we train two models, one on RealEstate10K [57], and one on the KITTI (Tulsiani split [47]). As Tab. 4 shows, our method achieves strong performance on both datasets, despite the fact, that we only predict geometry and obtain color by sampling the input image. Our results are comparable with many recent methods, that were specifically designed for this task, and of which some even use sparse depth supervision during training for RealEstate10K (MPI, MINE). MINE [22] achieves

	Configuration			Occupancy Estimation			Depth Prediction		
Method	Features	MLP	Predicts	O_{acc} ↑	IE_{acc} ↑	IE_{rec} ↑	Abs Rel ↓	RMSE ↓	$\alpha < 1.25$ ↑
PixelNeRF [52]	Enc	Big	$\sigma + c$	0.93	0.64	0.41	0.130	5.134	0.845
	E+D	Big	$\sigma + c$	0.93	0.57	0.44	0.149	5.441	0.800
	Enc	Small	$\sigma + c$	0.93	0.68	0.35	0.112	4.897	0.860
	E+D	Small	$\sigma + c$	0.93	0.68	0.16	0.109	4.758	0.864
	Enc	Small	σ	**0.95**	0.81	**0.47**	0.105	4.590	0.872
Ours	**E+D**	**Small**	σ	**0.95**	**0.82**	**0.47**	**0.102**	**4.407**	**0.882**
Ours	Keep invalid rays			0.93	0.74	0.45	0.108	4.493	0.875

Table 3. **Ablation Studies.** Evaluation of variants with different contributions (predicting only density σ and sampling color, shifting capacity from the MLP to the feature extractor, discarding invalid rays) turned on / off. Occupancy estimation results on KITTI-360 and depth prediction results on KITTI. The variant using only an encoder, big MLP, and color prediction corresponds exactly to PixelNeRF [52]. **Legend**: *Enc* Encoder, *E+D* Encoder-Decoder, σ density, c color.

	KITTI			RealEstate10K		
Model	LPIPS ↓	SSIM ↑	PSNR ↑	LPIPS ↓	SSIM ↑	PSNR ↑
SynSin [50]	n/a	n/a	n/a	1.180	0.740	22.3
Tulsiani [47]	n/a	0.572	16.5	0.176	0.785	23.5
MPI [46]	n/a	0.733	19.5	n/a	n/a	n/a
MINE [22]	**0.112**	**0.828**	**21.9**	**0.156**	**0.822**	**24.5**
PixelNeRF [52]	0.175	0.761	20.1	n/a	n/a	n/a
Ours	0.144	0.764	20.1	0.194	0.755	24.0

Table 4. **Novel View Synthesis.** We test the NVS ability on KITTI (Tulsiani split [47]) and RealEstate10K (MINE split [22], target frame randomly sampled within 30 frames). Even though our method does not predict color, we still achieve strong results.

slightly better accuracy. This can be attributed to them being able to predict color and thereby circumventing issues arising from imperfect geometry.

4.5. Ablation Studies

Our architectural design choices are critically important for the strong performance of our method. To quantify the impact of the different contributions, we conduct ablation studies based on occupancy estimation on KITTI-360 and depth prediction on KITTI. PixelNeRF [52] can be seen as a basis, which we modify step-by-step to reach our proposed model. Namely, we 1. shift capacity from the MLP to the feature extractor and 2. introduce color sampling as an alternative to predicting the color alongside density.

As Tab. 3 shows, reducing the MLP capacity and using a more powerful encoder-decoder rather than encoder as feature extractor allows the model to learn significantly more precise overall geometry. We conjecture that a powerful feature extractor is more suited to generalize to unseen scenes based on a single input image, than a high-capacity MLP. The feature extractor outputs a geometry representation (*i.e.* the feature map) of the full scene in a single forward pass. During training, it receives gradient information from all points sampled in the camera frustum, conditioned on the input image. Thus, potential noise from small visual details gets averaged out. On the other hand, the MLP outputs density based on the coordinates and is conditioned on a local feature. The coordinates and feature are different for every sampled point, rather than per scene. Consequently, noise will affect the MLP training significantly more.

Introducing the sampling of color from the input frames further boosts accuracy, especially for occupancy estimation in occluded areas. We hypothesize that only predicting density simplifies the training task significantly. Crucially, the network does not have to hallucinate colors in occluded regions. Additionally, color sampling enforces strict multi-view consistency. The network cannot compensate imperfect geometry by predicting the correct color.

Finally, the results show that our policy of discarding invalid rays during training improves accuracy by reducing noise in the training signal. This mainly affects the border regions of the frustum.

5. Conclusion

In this paper, we introduced a new approach for learning to estimate the 3D geometric structure of a scene from a single image. Our method predicts a continuous density field, which can be evaluated at any point in the camera frustum. The key contributions in our paper are 1. color sampling, 2. architecture improvements, and 3. a new self-supervised loss formulation. This enables us to train a network on large in-the-wild datasets with challenging scenes, such as KITTI, KITTI-360, and RealEstate10K. We show that our method is able to capture geometry in occluded areas. We evaluate depth maps synthesized from the predicted representation achieving comparable results to state-of-the-art methods. Despite only predicting geometry, our model even achieves high accuracy for novel view synthesis from a single image. Finally, we justify all of our design choices through detailed ablation studies.

Acknowledgements. This work was supported by the ERC Advanced Grant SIMULACRON, the GNI project AI4Twinning and the Munich Center for Machine Learning. C. R. is supported by VisualAI EP/T028572/1 and ERC-UNION-CoG-101001212.

References

[1] Shubhra Aich, Jean Marie Uwabeza Vianney, Md Amirul Islam, and Mannat Kaur Bingbing Liu. Bidirectional attention network for monocular depth estimation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11746–11752. IEEE, 2021. 2

[2] 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. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. 2

[3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022. 2

[4] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021. 2, 13

[5] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. *arXiv preprint arXiv:2203.09517*, 2022. 3

[6] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12882–12891, 2022. 2

[7] Helisa Dhamo, Nassir Navab, and Federico Tombari. Object-driven multi-layer scene decomposition from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5369–5378, 2019. 3

[8] Helisa Dhamo, Keisuke Tateno, Iro Laina, Nassir Navab, and Federico Tombari. Peeking behind objects: Layered depth prediction from a single image. *Pattern Recognition Letters*, 125:333–340, 2019. 3

[9] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, 27, 2014. 2, 7, 13, 15, 18

[10] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2002–2011, 2018. 2

[11] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 2, 5, 12, 13

[12] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 270–279, 2017. 2

[13] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3828–3838, 2019. 1, 2, 4, 5, 6, 7, 12, 13, 15, 18

[14] Juan Luis GonzalezBello and Munchurl Kim. Forget about the lidar: Self-supervised depth estimators with med probability volumes. *Advances in Neural Information Processing Systems*, 33:12626–12637, 2020. 1, 2

[15] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3d packing for self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2485–2494, 2020. 1, 2, 7, 15

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5, 12

[17] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5885–5894, 2021. 2

[18] James T Kajiya and Brian P Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH computer graphics*, 18(3):165–174, 1984. 3

[19] Mijeong Kim, Seonguk Seo, and Bohyung Han. Infonerf: Ray entropy minimization for few-shot neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12912–12921, 2022. 2

[20] Yevhen Kuznietsov, Jorg Stuckler, and Bastian Leibe. Semi-supervised deep learning for monocular depth map prediction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6647–6655, 2017. 2

[21] Sihaeng Lee, Janghyeon Lee, Byungju Kim, Eojindl Yi, and Junmo Kim. Patch-wise attention network for monocular depth estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1873–1881, 2021. 2

[22] Jiaxin Li, Zijian Feng, Qi She, Henghui Ding, Changhu Wang, and Gim Hee Lee. Mine: Towards continuous depth mpi with nerf for novel view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12578–12588, 2021. 3, 6, 7, 8, 12, 13, 15, 18

[23] Zhenyu Li, Zehui Chen, Xianming Liu, and Junjun Jiang. Depthformer: Exploiting long-range correlation and local information for accurate monocular depth estimation. *arXiv preprint arXiv:2203.14211*, 2022. 2

[24] Zhenyu Li, Xuyang Wang, Xianming Liu, and Junjun Jiang. Binsformer: Revisiting adaptive bins for monocular depth estimation. *arXiv preprint arXiv:2204.00987*, 2022. 2

[25] Yiyi Liao, Jun Xie, and Andreas Geiger. Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 2, 5, 12, 13

[26] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE transactions on pattern*

analysis and machine intelligence, 38(10):2024–2039, 2015. 2

[27] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020. 3

[28] Chenxu Luo, Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, Ram Nevatia, and Alan Yuille. Every pixel counts++: Joint learning of geometry and motion with 3d holistic understanding. *IEEE transactions on pattern analysis and machine intelligence*, 42(10):2624–2641, 2019. 1, 2, 7, 15

[29] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995. 2, 3

[30] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019. 3

[31] 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. *Communications of the ACM*, 65(1):99–106, 2021. 1, 2, 3, 12

[32] Norman Müller, Andrea Simonelli, Lorenzo Porzi, Samuel Rota Bulò, Matthias Nießner, and Peter Kontschieder. Autorf: Learning 3d object radiance fields from single view observations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3971–3980, 2022. 3

[33] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989*, 2022. 3

[34] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5480–5490, 2022. 2

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 5, 12

[36] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision*, pages 523–540. Springer, 2020. 3

[37] Barbara Roessle, Jonathan T Barron, Ben Mildenhall, Pratul P Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12892–12901, 2022. 2

[38] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[39] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 2

[40] Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. Layered depth images. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 231–242, 1998. 3

[41] Prafull Sharma, Ayush Tewari, Yilun Du, Sergey Zakharov, Rares Ambrus, Adrien Gaidon, William T Freeman, Fredo Durand, Joshua B Tenenbaum, and Vincent Sitzmann. Seeing 3d objects in a single image via self-supervised static-dynamic disentanglement. *arXiv preprint arXiv:2207.11232*, 2022. 2, 3

[42] Chang Shu, Kun Yu, Zhixiang Duan, and Kuiyuan Yang. Feature-metric loss for self-supervised learning of depth and egomotion. In *European Conference on Computer Vision*, pages 572–588. Springer, 2020. 1, 2, 7, 15, 18

[43] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pages 746–760. Springer, 2012. 2

[44] Pratul P Srinivasan, Richard Tucker, Jonathan T Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 175–184, 2019. 3

[45] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11358–11367, 2021. 3

[46] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 551–560, 2020. 3, 8

[47] Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3d scene inference via view synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 302–317, 2018. 3, 7, 8, 12, 13, 15, 18

[48] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 4

[49] Jamie Watson, Michael Firman, Gabriel J Brostow, and Daniyar Turmukhambetov. Self-supervised monocular depth hints. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2162–2171, 2019. 1, 2, 7, 15

[50] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7467–7477, 2020. 3, 8

[51] Nan Yang, Rui Wang, Jorg Stuckler, and Daniel Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 817–833, 2018. 2

[52] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021. 2, 3, 6, 7, 8, 12, 14, 15

[53] Weihao Yuan, Xiaodong Gu, Zuozhuo Dai, Siyu Zhu, and Ping Tan. New crfs: Neural window fully-connected crfs for monocular depth estimation. *arXiv preprint arXiv:2203.01502*, 2022. 1, 2

[54] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 340–349, 2018. 1, 2

[55] Kaichen Zhou, Lanqing Hong, Changhao Chen, Hang Xu, Chaoqiang Ye, Qingyong Hu, and Zhenguo Li. Devnet: Self-supervised monocular depth learning via density volume construction. *arXiv preprint arXiv:2209.06351*, 2022. 2, 7, 15, 18

[56] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1851–1858, 2017. 1, 2

[57] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018. 3, 5, 7, 12, 13

[58] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *European conference on computer vision*, pages 286–301. Springer, 2016. 3

A. Ethics

This research uses datasets (KITTI [11], KITTI-360 [25], and RealEstate10K [57]) to develop and benchmark computer vision models. The datasets are used in a manner compatible with their terms of usage. Some datasets the images can contain visible faces and other personal data collected without consent, however there is no processing of biometric information. Images are CC-BY or used in a manner compatible with the Data Analysis Permission. We do not process biometric information. Please see https://www.robots.ox.ac.uk/~vedaldi/research/union/ethics.html for further information on ethics and data protection rights and mitigation.

B. Limitations

As the model makes predictions from a single frame, it can only rely on priors to predict visible *and invisible* parts of the scene. Naturally, it should thus not be used in safety-critical applications, nor outside of a research setting.

As the model is sampling colors instead of predicting them, it cannot predict plausible colors for unseen objects. Thus, in the setting of novel view synthesis, extreme camera pose changes tend to lead to visible artifacts in the images.

Similar to self-supervised depth prediction methods, our loss formulation relies on photometric consistency. View-dependent effects are not modeled explicitly and could therefore introduce noise in the training process.

Further, our loss formulation relies on a static scene assumption and dynamic objects are not modeled explicitly. While this has the potential to reduce accuracy, there are several reasons why it only has marginal effect in our case. 1. In most cases, we have stereo frames available, which give accurate training signals, even for moving objects. 2. The time difference between the different views is very small (usually in the order of 0.1 seconds. Therefore, even if an object is moving, the introduced noise is rather small. Nonetheless, it would be an interesting direction for future work to investigate explicit modelling of dynamic objects in a loss formulation like ours.

C. Additional Results

In the following, we show additional results for novel view synthesis, capturing true 3D, and depth prediction. Please also see the video for additional qualitative results and explanations. Figures can be found after the text of the supplementary material.

C.1. Novel View Synthesis

Fig. 9 shows qualitative results from the Tulsiani [47] test split for KITTI [11]. Fig. 10 shows qualitative results for the test set of RealEstate10K [57] proposed by MINE [22].

C.2. Capturing True 3D

Fig. 11 shows further visualizations of the predicted density field, in which you can clearly make out the different objects in the scene in the top-down view.

C.3. Depth Prediction

Fig. 12 shows further results comparing our expected ray termination depth with results from other depth prediction methods. Tab. 6 reports additional metrics to the table in the main paper.

D. Technical Details

In the following, we discuss the exact implementation details, network configurations, and training setup, so that our results can be reproduced easily. Further, we provide further details regarding the computation of the occupancy metrics.

D.1. Implementation Details

We base our implementation on the official code repository published by [52]. Further, we are inspired by the repository of [13] regarding the implementation of the image reconstruction loss functions.

Networks. For encoder, we use a ResNet-50 [16] backbone pretrained on ImageNet. We rely on the official weights provided by PyTorch [35] / Torchvision. As decoder, we follow the architecture of MonoDepth2 [13] with a minor modification. Since we output feature maps with C channels at the same resolution of the input, we do not reduce the features during upconvolutions below C to prevent information loss. Concretely for every layer, we have an output channel dimension of $\hat{C}_{\text{out}} = \max(C, C_{\text{out}})$, where C_{out} is the output channel dimension of the MonoDepth2 model. We found $C = 64$ to give best results.

For the decoding MLP, we use two fully-connected layers with hidden dimension C (same as the feature dimension) and ReLU activation function. We found that more layers do not improve the quality of the reconstruction. Our hypothesis is that the decoding is a simple task that does not require a network with high capacity.

Rendering. To obtain a color / expected ray termination depth for a given ray, we sample S points between z_{near} and z_{far}. As we deal with potentially unbounded scenes with many different scales, we use inverse depth to obtain the ranges for the different parts. For every range, we uniformly draw one sample. Let d_i be the depth step for the i-th point ($i \in [0, S-1]$) and $r \sim U[0,1]$ a random sample from the uniform distribution between 0 and 1.

$$d_i = 1/\left(\frac{1-s_i}{z_{\text{near}}} + \frac{s_i}{z_{\text{far}}}\right), \quad s_i = \frac{i+r}{S} \tag{9}$$

We also experimented with coarse and fine sampling as used in many NeRF papers (*e.g.* [31,52]). Here, after sampling the entire range of depths as above (coarse sampling), we perform importance sampling based on the returned weights and sampling around the expected ray termination depth (fine sampling). Further, we also duplicate the MLP: one for coarse and one for fine sampling. While the outputs of both networks are used for two seperate reconstructions with separate losses, only the fine reconstruction results are used for evaluation. This technique is particularly helpful in NeRFs to increase the visual quality. While the coarse MLP has to model the density and color distribution for a big range of coordinates, the fine MLP only has to learn the relevant area around surfaces. In our experiments, we found that we do not get any benefit from adding fine sampling, both when using two seperate MLPs or one for coarse and fine sampling. We suspect that our single MLP already has enough capacity to model the density distribution (we do not model color with the MLP) described by the feature at sufficient accuracy.

Positional Encoding. As described in the main paper, we pass d_i and $\mathbf{u}'_{\mathbf{I}}$ (pixel coordinate) values through a positional encoding function, before feeding them to the network along side the sampled feature $f_{\mathbf{u}'_{\mathbf{I}}}$. This positional encoding functions maps the input to sin and cos functions with different frequencies. This is an established practice in methods where networks have to reason about the spatial location of points in 2D or 3D [31,52]. As we deal with real-world scale of scenes, we first normalize the depth to $[-1, 1]$. This ensures that the data-range perfectly matches the used frequencies. $\mathbf{u}'_{\mathbf{I}}$ uses normalized pixel coordinates with $\mathbf{u}'_{\mathbf{I}} \in [-1, 1]^2$ already. For a vector, we compute the positional encoding per element as:

$$\begin{aligned}\gamma(x) = [&x, \sin(x\pi 2^0), \cos(x\pi 2^0), \sin(x\pi 2^1), \cos(x\pi 2^1), \\ &\ldots, \sin(x\pi 2^6), \cos(x\pi 2^6)]\end{aligned} \tag{10}$$

D.2. Training Configuration

Through preliminary experiments, we found that the following training configuration yields the best results:

Dataset	Split	#Train	#Val.	#Test
KITTI [11]	Eigen [9]	39.810	4.424	697
	Tulsiani [47]	11.987	1.243	1.079
KITTI-360 [25]	Ours	98.008	11.451	446
RealEstate10K [57]	MINE [22]	8.954.743	245	3270

Table 5. **Dataset Overview.** Different datasets used in this work with information on data split. Our KITTI-360 split is a modified version of the split for the image segmentation task.

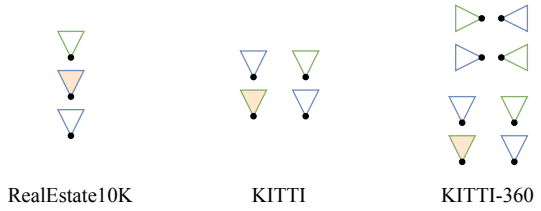

Figure 6. **Frame Arrangement per Sample.** RealEstate10K only has monocular sequences. KITTI and KITTI-360 provide stereo. KITTI-360 also contains fisheye camera frames facing left and right. **Legend**: ref. Fig. 3.

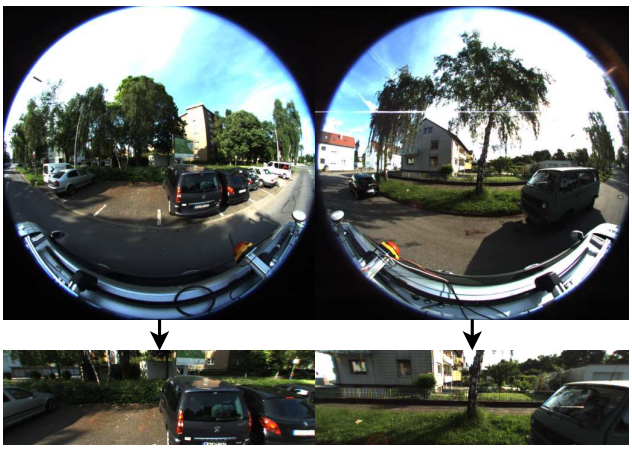

Figure 7. **Fisheye Resampling.** KITTI-360 provides frames from two fisheye cameras, one facing to the left and one facing to the right. We resample them based on a virtual perspective camera that has the same camera intrinsics as the perspective forward-facing cameras. We rotate the virtual camera 15° downward to maximise the overlap of the frustums with the forward-facing cameras.

In all of our experiments, we use a batch size of 16. In total, we sample 2048 rays for each item in a batch. These rays are grouped in 8×8 sized patches randomly sampled from any of the frames in N_{loss} From this, the loss is computed. Following [13], we set $\lambda_{\text{SSIM}} = 0.85$, $\lambda_{\text{L1}} = 0.15$ and $\lambda_{\text{EAS}} = 0.001 * 2$. The default learning rate is $\lambda = 10^{-4}$ and we decrease it to $\lambda = 10^{-5}$ for the last 20% of the training. For all trainings, we use color augmentation (same parameters for all views of an item in the batch) and flip augmentation (randomly horizontally flip the image that gets fed into the encoder-decoder and then flip the resulting feature maps back to avoid changing the geometry of the scene). Tab. 5 shows an overview of the different datasets and the used split. Fig. 6 visualizes the frame arrangement and a possible partitioning into N_{loss} and N_{render} for the different datasets.

KITTI. We rely on poses computed from ORB-SLAM 3 [4], which uses the given stereo cameras, intrinsics, and baseline length. To be consistent with popular depth prediction methods, we perform all trainings and experiments at a resolution of 640×192 and rely on the Eigen split [9]. For evaluation, like in most other works, the cut off distance is set to 80m. The training runs for 50 epochs and we reduce the learning rate after 100.000 iterations. We report depth prediction results for our model which was trained with two timesteps (input + following) and stereo, *i.e.* four frames in total. For occupancy estimation, we also train a model with three timesteps (previous + input + following) and stereo. Depth prediction results for this model are on par, but not better than the model trained with two timesteps. A depth range of $z_{\text{near}} = 3\text{m}$ and $z_{\text{far}} = 80\text{m}$ proved to work best.

On the Tulsiani split [47], we use the same settings, except that we train for 150 epochs, as the split contains around $3\times$ fewer samples.

Training in both cases takes around four days.

KITTI-360. As the setting is very similar to KITTI, we use the same parameters. Because the dataset is significantly larger, we only train for 25 epochs. Training again takes around four days. Additionally to the two stereo frames, we also have access to fisheye camera pointing left and right. In order to be able to use them within our implementation, we resample them based on a virtual perspective camera with the same parameters as the forward-facing perspective cameras. Note that the fisheye cameras seem to be mounted higher up than the perspective cameras. Therefore, we rotate the virtual cameras 15° downwards along the x-axis during the resampling process. Further, fisheye and forward-facing cameras of the same timeframe have barely any overlapping visible areas. Therefore, we offset fisheye cameras by 10 timesteps. Fig. 7 shows examples from the fisheye cameras and the resampled images.

RealEstate10K As RealEstate10K contains magnitudes more images than KITTI or KITTI-360, approximately 8 mio. for training. Therefore, we define the training length by the number of iterations, in this case 360k. We follow [22] and perform all experiments at a resolution of 384×256. We train with three frames (previous + input + following) per item in the batch. As the framerate is very high, we randomly draw an offset from the range $[1, 30]$ between the frames. [57] states that all sequences are normalized to fit a depth range of $z_{\text{near}} = 1m$ and $z_{\text{far}} = 100m$ with inverse depth spacing. We use the poses provided by the dataset.

D.3. Occupancy Metric

We rely on Lidar scans provided by KITTI-360 to build ground-truth occupancy and visibility maps, which we then use to evaluate our prediction quality. Our evaluation protocol relies on 2D slices parallel to the ground between the street and the height of the car. This allows to focus on the interesting regions of the scene, that also contain other objects like cars and pedestrians, and ignore areas that are not interesting, like the area below the street or the sky.

Consider now a single input frame, for which we would like to build our ground-truth occupancy and visibility. As KITTI-360 is an autonomous driving dataset, the vehicle is generally moving forward at a steady pace. Thus, consecutive Lidar scans captured a short time later measure different areas within the camera frustum. Note that these Lidar measurements can reach areas that are occluded in the input image. To determine whether a point is occupied, we check whether it is *in front* of the measured surface for any of the Lidar scans. Intuitively, every Lidar measurement "carves out" unoccupied areas in 3D space.

Let $L_i = \left\{\mathbf{x}_j \in \mathrm{R}^3 | j \in [M]\right\}$ be the Lidar scan i timesteps after the input frame with M measurement points in the world coordinate system. L_0 denotes the Lidar scan captured synchronously to the input frame. Let P_i denote the vehicle-to-world transformation (for ease of notation we assume that the Lidar scanner is centered at the vehicle pose and that the

input frame has the same pose as the the 0-th Lidar scan).

3D interpolation with sparse Lidar point clouds is difficult. Therefore, we extract slices from the scan pointclouds and project them onto a 2D plane. Additionally, we convert every point from Cartesian coordinates to polar coordinates, centered around the origin of the respective Lidar scan. This makes the measurements much more dense and evaluation of whether a point is in front or behind a surface easier.

Let $y_{\text{min}}, y_{\text{max}}$ describe the min and max y-coordinate for our slice. $\text{pol}_{xz}(\mathbf{x}) \rightarrow (\alpha, d)$ denotes a function to convert a Cartesian coordinate to a polar coordinate after projecting it onto the xz-plane.

$$S_i = \left\{ \text{pol}_{xz}(\mathbf{x}_j) | \mathbf{x}_j \in L_i \wedge y_{\text{min}} \leq x_j^y \leq y_{\text{max}} \right\} \tag{11}$$

For a given Lidar scan, we can now check whether a point $\mathbf{x}$ is in front or behind the measured surface by transforming it into the scan's coordinate system, converting it to polar coordinates and then comparing the distance. However, experiments showed that the Lidar scans in KITTI-360 can be fairly noisy, especially for objects like cars, that have translucent or reflective materials. Oftentimes, single outlier points are measured to be at a much bigger distance. As we rely on the "carving out" idea, such points would carve out a lot of free space and lead to inaccurate occupancy maps. To filter out these outliers, we split the 360 degree range of the polar coordinates into $b = 360$ equally sized bins and assign every measured point to the corresponding bin. For every bin $S_i[\alpha]$, we then choose the minimal measured distance.

Using the S_i, we now define a function that allows to check whether a point is occupied or not. Note that we can obtain the two closest bins for a given angle α through the floor and ceil functions.

$$\begin{aligned} \alpha, d &= \text{pol}_{xz}(P_i^{-1}\mathbf{x}) \\ \alpha_l, \alpha_r &= \lfloor \alpha \rfloor, \lceil \alpha \rceil \\ \delta &= \frac{\alpha - \alpha_l}{\alpha_r - \alpha_l} \\ \text{is\_free}_i(\mathbf{x}) &= d < ((1-\delta) S_i[\alpha_l] + \delta S_i[\alpha_r]) \end{aligned} \tag{12}$$

We then accumulate several timesteps $i \in [0, N-1]$ to build a ground-truth occupancy map. In practice, we consider $N = 20$ timesteps.

$$\text{occ}(\mathbf{x}) = \neg \left(\bigvee_{i \in [0,N]} \text{is\_free}_i(\mathbf{x}) \right) \tag{13}$$

Similarly, we determine visibility by only considering the Lidar scan corresponding to the input frame:

$$\text{vis}(\mathbf{x}) = \neg \text{is\_free}_0(\mathbf{x}) \tag{14}$$

Based on these functions, we can compute the final metric results. We consider a point $\mathbf{x}$ to be occupied, if the predicted density is over a threshold: $\sigma_{\mathbf{x}} > 0.5$. Let $\mathbf{x}_i, i \in [1, N_{\text{pts}}]$ be points we sample from the camera frustum. Let $X_{\neg\text{vis}} = \{i \in [1, N_{\text{pts}}] | \neg\text{vis}(\mathbf{x}_i)\}$ be the subset of points that are invisible, and $X_{\neg\text{vis}\wedge\neg\text{occ}} = \{i \in [1, N_{\text{pts}}] | \neg\text{vis}(\mathbf{x}_i) \wedge \neg\text{occ}(\mathbf{x}_i)\}$ be the subset of points that are invisible *and* empty.

$$O_{\text{acc}} = \frac{1}{N_{\text{pts}}} \sum_{i=1}^{N_{\text{pts}}} (\text{occ}(\mathbf{x}) == (\sigma_{\mathbf{x}} > 0.5)) \tag{15}$$

$$\text{IE}_{\text{acc}} = \frac{1}{|X_{\neg\text{vis}}|} \sum_{i \in X_{\neg\text{vis}}} (\text{occ}(\mathbf{x}) == (\sigma_{\mathbf{x}} > 0.5)) \tag{16}$$

$$\text{IE}_{\text{rec}} = \frac{1}{|X_{\neg\text{vis}\wedge\neg\text{occ}}|} \sum_{i \in X_{\neg\text{vis}\wedge\neg\text{occ}}} (\sigma_{\mathbf{x}} < 0.5) \tag{17}$$

We sample 2720 points in total, uniformly spaced from a cuboid with dimensions $x = [-4m, 4m], y = [0m, 1m], z = [3m, 20m]$ (y-axis facing downward). This means that all points are just above the surface of the street. Fig. 8 shows examples of the evaluation for two samples. Evaluation code will be included in the code release.

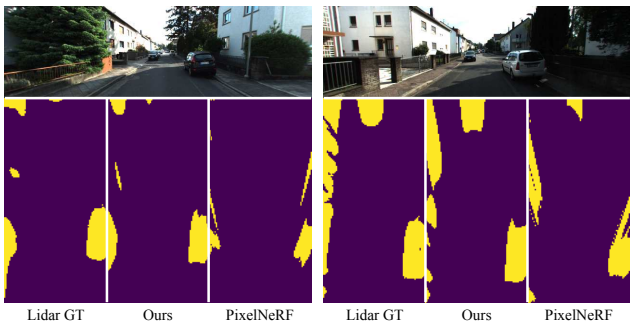

Figure 8. **Occupancy Metric on KITTI-360.** Visualization of the 1. occupancy ground-truth accumulated from 20 Lidar scans, 2. predicted occupancy map by our model, and 3. predicted occupancy map by PixelNeRF [52].

E. Additional Considerations

In this section, we discuss hypotheses on the working principles of our proposed approach, for which we do not have immediate experimental results.

Training Stability. Contrary to many NeRF-based methods, we find that our proposed approach offers stable training and that it is not overly sensitive to changes in hyperparameters. We hypothesize, that this is due to the nature of color sampling, which shares similarities with classical stereo matching. When casting a ray and sampling the color from a frame, the sampling positions will lie on the epipolar line. The best match on this epipolar line, which would be the desired correspondence point in stereo matching, will give the smallest loss and a clear training signal. This is even the case when sampling color from a single or very few frames. In contrast, with a NeRF formulation, the color gets learned when multiple rays with the same color go through the same area in space. Therefore, here we require many views to give a meaningful signal.

Reconstruction Quality. One of the key advantages of NeRF-based methods is that they offer a great way to aggregate the information from many frames that see the same areas of a scene. In our formulation, color is only aggregated through the $\min$ operation in the loss term. In a setting with many views, NeRF would clearly provide better reconstruction quality than a density field with color sampling.

However, in settings, where there are only few view scenes per scene available, most areas in the scene have very limited view coverage. This means, that the aggregation aspect of NeRFs becomes much less relevant and the "visual expressiveness" of NeRFs and density fields with color sampling converge.

F. Visualizations

Assets for Fig. 2 were taken from Blendswap[3][4] under the CC-BY license.

[3] https://blendswap.com/blend/18686
[4] https://blendswap.com/blend/13698

Input Image | Predicted Novel View | Target View

Figure 9. **Novel View Synthesis on KITTI.** Rendering the right stereo frame based on the density field predicted from the left stereo frame. Colors are also sampled from the same frame we make the prediction from. Areas of the image that are not occluded in both the input and target frame are reconstructed very accurately.

Model	Volumetric	Split	Abs Rel	Sq Rel	RMSE	$\text{RMSE}_{\log}$	$\alpha < 1.25$	$\alpha < 1.25^2$	$\alpha < 1.25^3$
PixelNeRF [52]	✓	Eigen [9]	0.130	1.241	5.134	0.220	0.845	0.943	0.974
EPC++ [28]	✗		0.128	1.132	5.585	0.209	0.831	0.945	0.979
MonoDepth 2 [13]	✗		0.106	0.818	4.750	0.196	0.874	0.957	0.975
PackNet [15]	✗		0.111	0.785	4.601	0.189	0.878	0.960	0.982
DepthHint [49]	✗		0.105	0.769	4.627	0.189	0.875	0.959	0.982
FeatDepth [42]	✗		0.099	0.697	4.427	0.184	0.889	0.963	0.982
DevNet [55]	(✓)		**0.095**	**0.671**	**4.365**	**0.174**	**0.895**	**0.970**	**0.988**
Ours	✓		0.102	0.751	4.407	0.188	0.882	0.961	0.982
MINE [22]	✓	Tulsiani [47]	0.137	1.993	6.592	0.250	0.839	0.940	0.971
Ours	✓		**0.132**	**1.936**	**6.104**	**0.235**	**0.873**	**0.951**	**0.974**

Table 6. **Depth Prediction on KITTI.** While our goal is full volumetric scene understanding, we compare to state-of-the-art self-supervised depth estimation method. Our approach achieves competitive performance while clearly improving over other volumetric approaches like PixelNeRF [52] and MINE [22]. DevNet [55] performs better, but does not show any results of their volume.

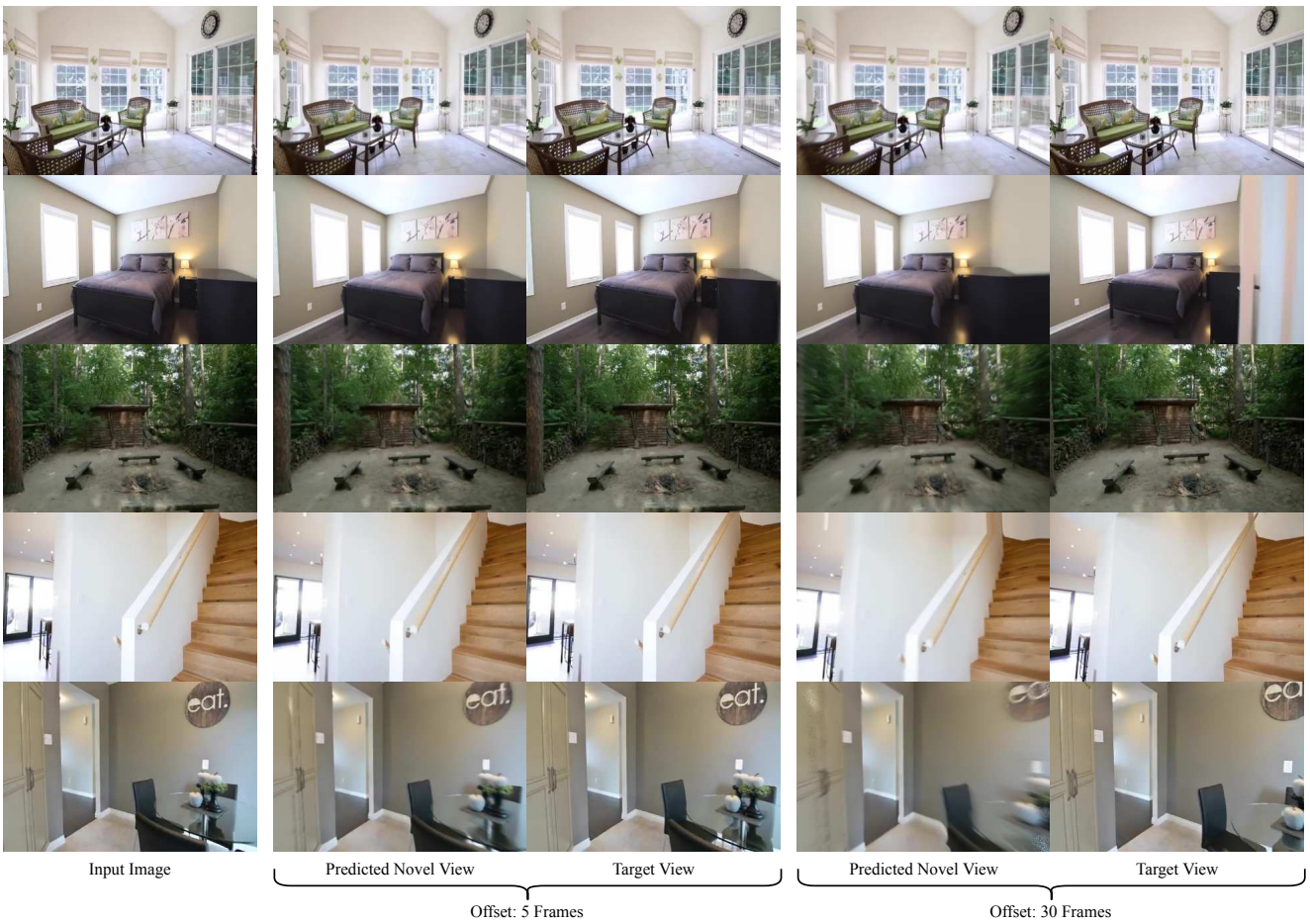

Figure 10. **Novel View Synthesis on RealEstate10K.** Rendering a later frame based on the density field predicted from the input frame. Colors are also sampled from the same frame we make the prediction from.

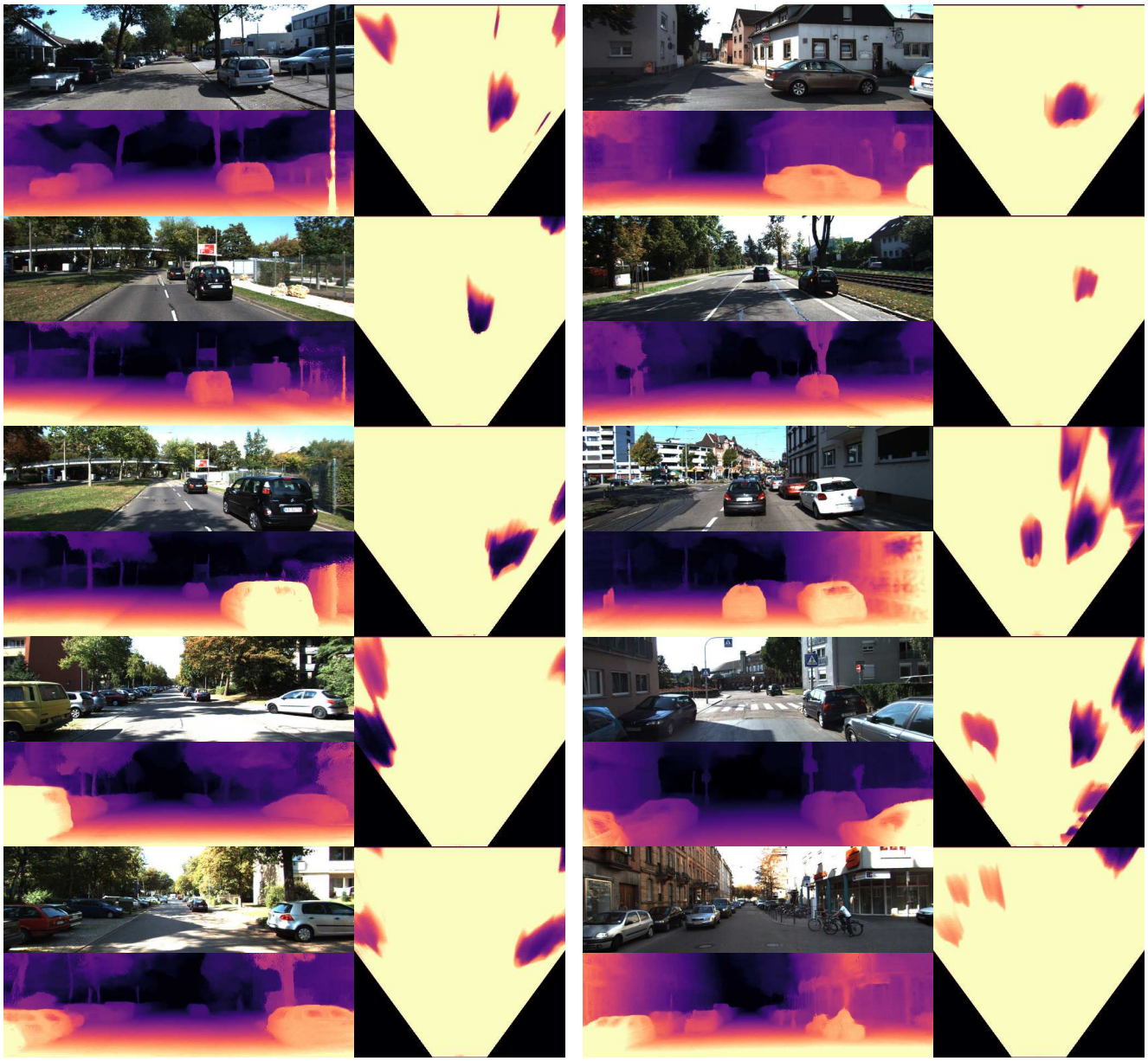

Figure 11. **Occupancy Estimation.** More qualitative top-down visualization of the occupancy map predicted by different methods. We show an area of $x = [-15m, 15m], z = [5m, 30m]$ and aggregate density from the y-coordinate of the camera $1m$ downward.

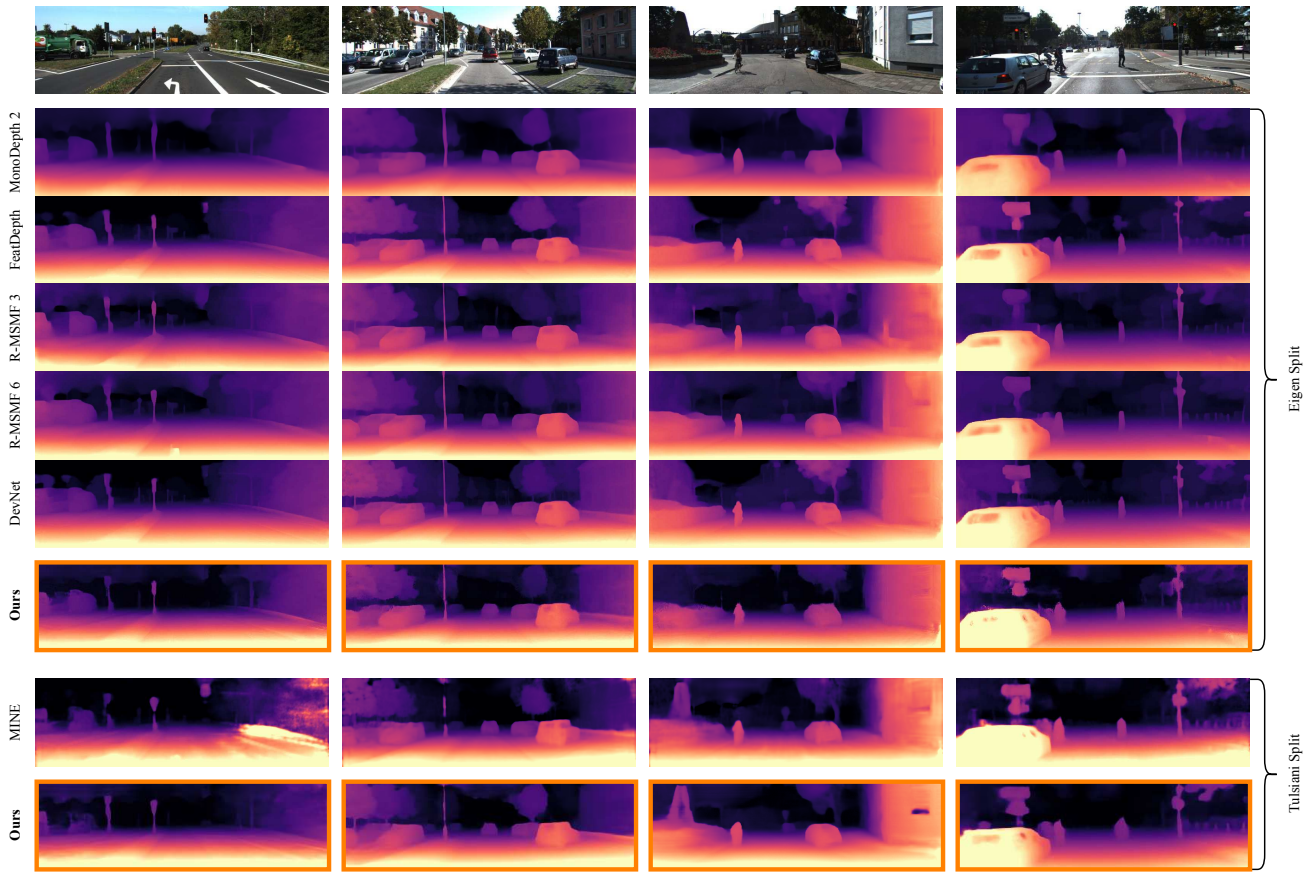

Figure 12. **Depth Prediction.** Additional visualizations of the expected ray termination depth compared with depth prediction results of other state-of-the-art methods [13, 22, 42, 55] on both the Eigen [9] and [47] split. Visualizations for DevNet and FeatDepth are taken from [55].