Supplementary Material - Behind the Scenes: Density Fields for Single View Reconstruction

Felix Wimbauer[1,2] Nan Yang[1] Christian Rupprecht[3] Daniel Cremers[1,2,3]
[1]Technical University of Munich [2]MCML [3]University of Oxford
{felix.wimbauer, nan.yang, cremers}@tum.de chrisr@robots.ox.ac.uk

Project page & Code: fwmb.github.io/bts

1. Ethics

This research uses datasets (KITTI [3], KITTI-360 [8], and RealEstate10K [17]) 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.

2. 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.

3. 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.

3.1. Novel View Synthesis

Fig. 4 shows qualitative results from the Tulsiani [13] test split for KITTI [3]. Fig. 5 shows qualitative results for the test set of RealEstate10K [17] proposed by MINE [7].

3.2. Capturing True 3D

Fig. 6 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.

3.3. Depth Prediction

Fig. 7 shows further results comparing our expected ray termination depth with results from other depth prediction methods. Tab. 2 reports additional metrics to the table in the main paper.

4. 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.

4.1. Implementation Details

We base our implementation on the official code repository published by [15]. Further, we are inspired by the

repository of [4] regarding the implementation of the image reconstruction loss functions.

Networks. For encoder, we use a ResNet-50 [6] backbone pretrained on ImageNet. We rely on the official weights provided by PyTorch [11] / Torchvision. As decoder, we follow the architecture of MonoDepth2 [4] 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.

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

We also experimented with coarse and fine sampling as used in many NeRF papers (*e.g.* [10, 15]). 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.

Dataset	Split	#Train	#Val.	#Test
KITTI [3]	Eigen [2]	39.810	4.424	697
	Tulsiani [13]	11.987	1.243	1.079
KITTI-360 [8]	Ours	98.008	11.451	446
RealEstate10K [17]	MINE [7]	8.954.743	245	3270

Table 1. **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.

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 [10, 15]. 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:

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

4.2. Training Configuration

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

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 [4], 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. 1 shows an overview of the different datasets and the used split. Fig. 1 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 [1], 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

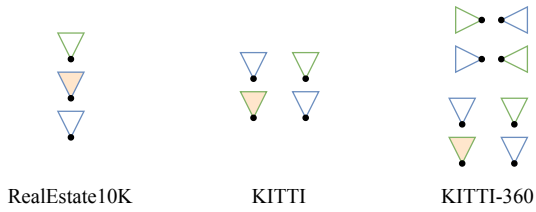

Figure 1. **Frame Arrangement per Sample.** RealEstate10K only has monocular sequences. KITTI and KITTI-360 provide stereo. KITTI-360 also contains fisheye camera frames facing left and right. **Legend**: ref. Fig. 3.

resolution of 640×192 and rely on the Eigen split [2]. 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 [13], 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. 2 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 [7] 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

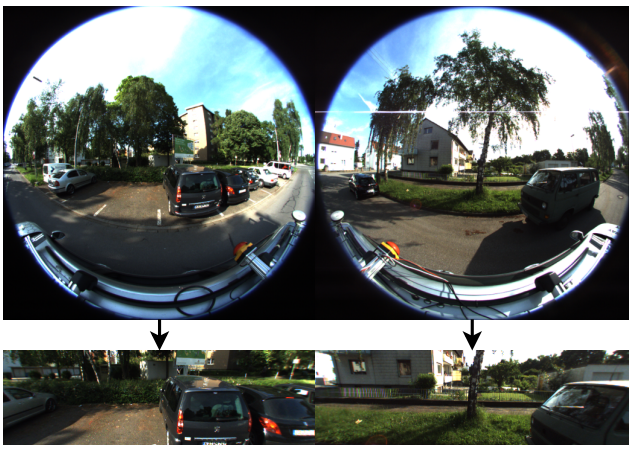

Figure 2. **Fisheye Resampling.** KITTI-360 provides frames from two fisheye cameras, one facing to the left and one facing to the right. We resample them based on a virtual perspective camera that has the same camera intrinsics as the perspective forward-facing cameras. We rotate the virtual camera $15°$ downward to maximize the overlap of the frustums with the forward-facing cameras.

randomly draw an offset from the range $[1, 30]$ between the frames. [17] 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.

4.3. 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 $L_i = \{\mathbf{x}_j \in \mathrm{R}^3 | j \in [M]\}$ 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.

$$S_i = \left\{ \text{pol}_{xz}(\mathbf{x}_j) | \mathbf{x}_j \in L_i \wedge y_{\text{min}} \leq x_j^y \leq y_{\text{max}} \right\} \quad (3)$$

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.

$$\begin{aligned} \alpha, d &= \text{pol}_{xz}(P_i^{-1}\mathbf{x}) \\ \alpha_l, \alpha_r &= \lfloor \alpha \rfloor, \lceil \alpha \rceil \\ \delta &= \frac{\alpha - \alpha_l}{\alpha_r - \alpha_l} \\ \text{is\_free}_i(\mathbf{x}) &= d < ((1-\delta) S_i[\alpha_l] + \delta S_i[\alpha_r]) \end{aligned} \quad (4)$$

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

$$\text{occ}(\mathbf{x}) = \neg \left(\bigvee_{i \in [0,N]} \text{is\_free}_i(\mathbf{x}) \right) \quad (5)$$

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

$$\text{vis}(\mathbf{x}) = \neg \text{is\_free}_0(\mathbf{x}) \quad (6)$$

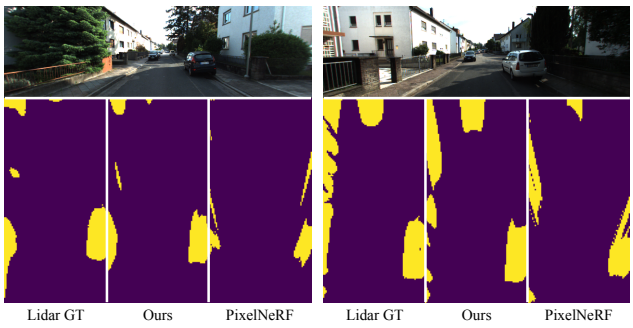

Figure 3. **Occupancy Metric on KITTI-360.** Visualization of the 1. occupancy ground-truth accumulated from 20 Lidar scans, 2. predicted occupancy map by our model, and 3. predicted occupancy map by PixelNeRF [15].

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.

$$O_{\text{acc}} = \frac{1}{N_{\text{pts}}} \sum_{i=1}^{N_{\text{pts}}} (\text{occ}(\mathbf{x}) == (\sigma_{\mathbf{x}} > 0.5)) \quad (7)$$

$$\text{IE}_{\text{acc}} = \frac{1}{|X_{\neg\text{vis}}|} \sum_{i \in X_{\neg\text{vis}}} (\text{occ}(\mathbf{x}) == (\sigma_{\mathbf{x}} > 0.5)) \quad (8)$$

$$\text{IE}_{\text{rec}} = \frac{1}{|X_{\neg\text{vis}\wedge\neg\text{occ}}|} \sum_{i \in X_{\neg\text{vis}\wedge\neg\text{occ}}} (\sigma_{\mathbf{x}} < 0.5) \quad (9)$$

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. 3 shows examples of the evaluation for two samples. Evaluation code will be included in the code release.

5. 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.

6. Visualizations

Assets for Fig. 2 were taken from Blendswap[1][2] under the CC-BY license.

References

[1] 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

[2] 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, 3, 6, 9

[3] 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. 1, 2

[4] 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. 2, 6, 9

[5] 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. 6

[6] 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. 2

[7] 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. 1, 2, 3, 6, 9

[8] 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. 1, 2

[9] 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. 6

[10] 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. 2

[11] 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. 2

[12] 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. 6, 9

[13] 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. 1, 2, 3, 6, 9

[14] 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. 6

[15] 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. 1, 2, 4, 6

[16] 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. 6, 9

[17] 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. 1, 2, 3

[1] https://blendswap.com/blend/18686
[2] https://blendswap.com/blend/13698

Input Image | Predicted Novel View | Target View

Figure 4. **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 [15]	✓	Eigen [2]	0.130	1.241	5.134	0.220	0.845	0.943	0.974
EPC++ [9]	✗		0.128	1.132	5.585	0.209	0.831	0.945	0.979
MonoDepth 2 [4]	✗		0.106	0.818	4.750	0.196	0.874	0.957	0.975
PackNet [5]	✗		0.111	0.785	4.601	0.189	0.878	0.960	0.982
DepthHint [14]	✗		0.105	0.769	4.627	0.189	0.875	0.959	0.982
FeatDepth [12]	✗		0.099	0.697	4.427	0.184	0.889	0.963	0.982
DevNet [16]	(✓)		**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 [7]	✓	Tulsiani [13]	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 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 [15] and MINE [7]. DevNet [16] performs better, but does not show any results of their volume.

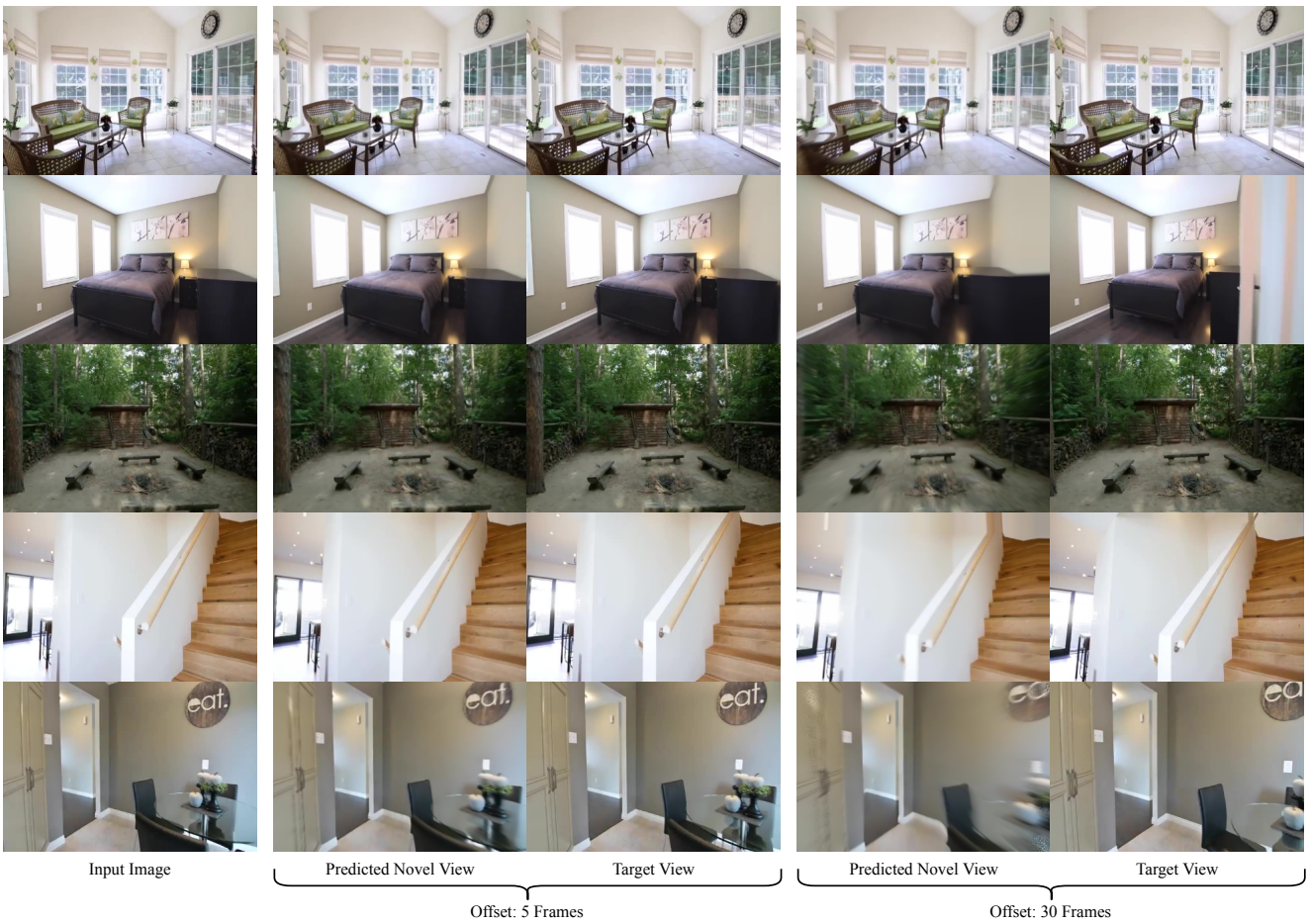

Figure 5. **Novel View Synthesis on RealEstate10K.** Rendering a later frame based on the density field predicted from the input frame. Colors are also sampled from the same frame we make the prediction from.

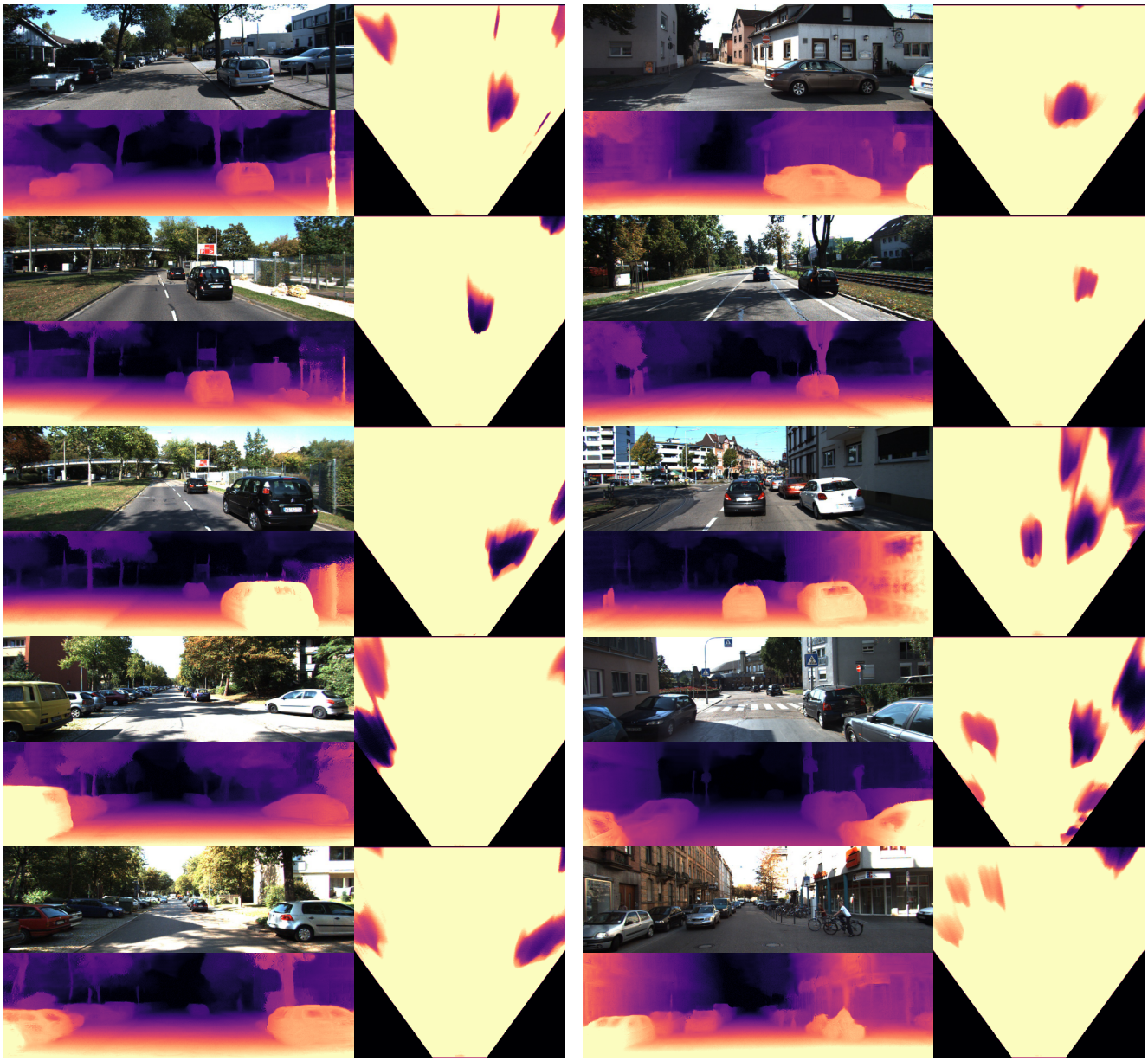

Figure 6. **Occupancy Estimation.** More qualitative top-down visualization of the occupancy map predicted by different methods. We show an area of $x = [-15m, 15m], z = [5m, 30m]$ and aggregate density from the y-coordinate of the camera $1m$ downward.

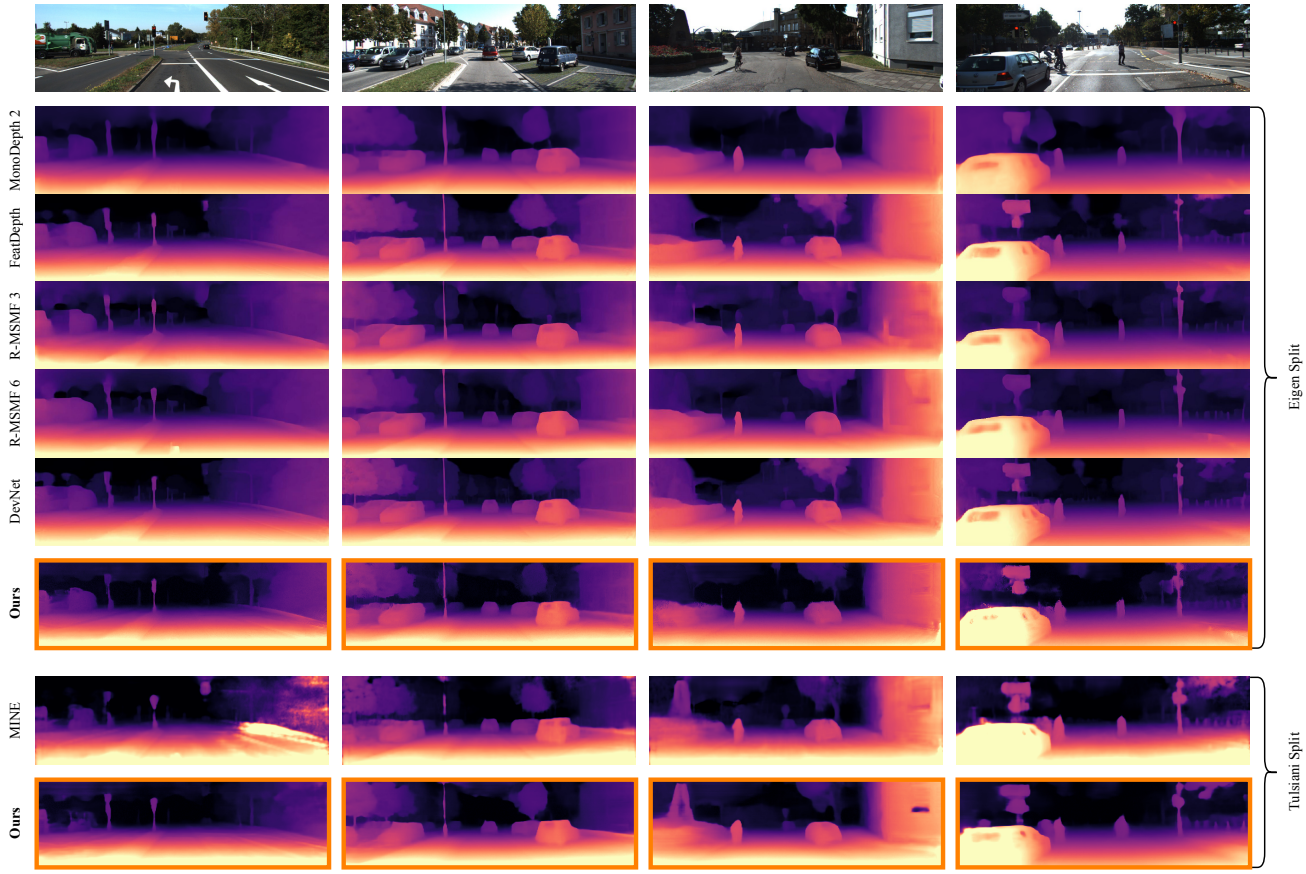

Figure 7. **Depth Prediction.** Additional visualizations of the expected ray termination depth compared with depth prediction results of other state-of-the-art methods [4, 7, 12, 16] on both the Eigen [2] and [13] split. Visualizations for DevNet and FeatDepth are taken from [16].