Heightfields for Efficient Scene Reconstruction for AR

Jamie Watson[1,2] Sara Vicente[1] Oisin Mac Aodha[3]
Clément Godard[4*] Gabriel Brostow[1,2] Michael Firman[1]
[1]Niantic [2]UCL [3]University of Edinburgh [4]Google

https://github.com/nianticlabs/heightfields

Abstract

3D scene reconstruction from a sequence of posed RGB images is a cornerstone task for computer vision and augmented reality (AR). While depth-based fusion is the foundation of most real-time approaches for 3D reconstruction, recent learning based methods that operate directly on RGB images can achieve higher quality reconstructions, but at the cost of increased runtime and memory requirements, making them unsuitable for AR applications. We propose an efficient learning-based method that refines the 3D reconstruction obtained by a traditional fusion approach. By leveraging a top-down heightfield representation, our method remains real-time while approaching the quality of other learning-based methods. Despite being a simplification, our heightfield is perfectly appropriate for robotic path planning or augmented reality character placement. We outline several innovations that push the performance beyond existing top-down prediction baselines, and we present an evaluation framework on the challenging ScanNetV2 dataset, targeting AR tasks.

1. Introduction

Systems for camera tracking are now ubiquitous and widely available in several augmented reality (AR) frameworks, e.g. Apple's ARKit [1] and Google's ARCore [13]. However, complex AR effects require 3D scene understanding that goes beyond camera poses and sparse points. A critical AR application is the placement and navigation of assets such as characters in a real-world scene. To do this well, an estimate of the 3D geometry is typically required. This 3D geometry estimate must be (a) cheap to compute, so it can run on embedded and mobile devices, and (b) accurate, so that AR assets will appear to be standing on objects, and not below or above the surface.

Although accurate, depth sensors are not standard for the majority of mobile devices, meaning that a typical approach is to fuse depth maps estimated from posed RGB images into a global 3D reconstruction [36]. This is fast, but creates noisy and incomplete 3D volumes. Recent works have proposed alternative learning-based approaches to high-quality 3D reconstruction. However, these tend to rely on expensive operations such as 3D convolutions [30, 48, 6]. In this work, we posit that these learning-based 3D reconstructions

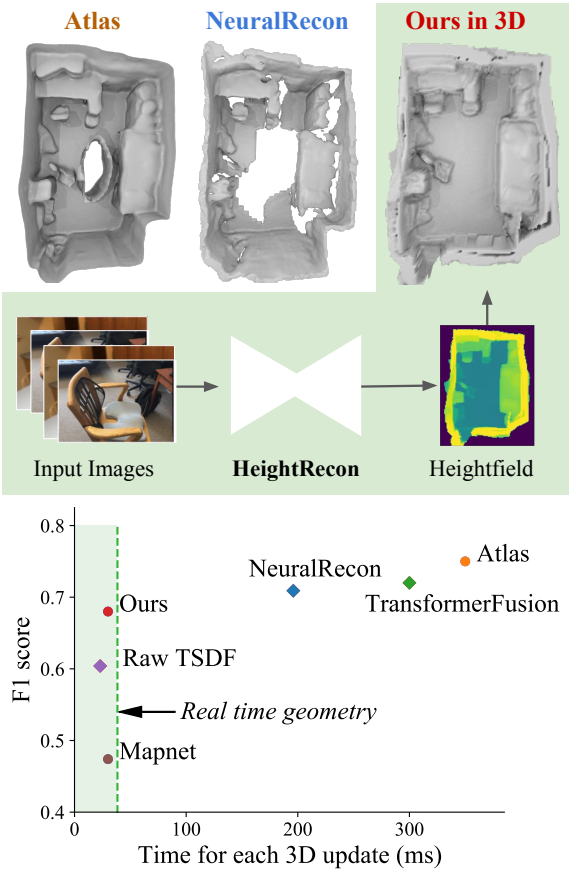


Figure 1. **Heightfields for reconstruction.** By using only 2D convolutions, our top-down heightfield prediction system is significantly faster than full 3D alternatives (bottom), while predicting scene shape with comparable accuracy (top).

*Work done while at Niantic

are unnecessary for many applications such as AR, and propose our 2½D **heightfield** of the scene as a practical alternative in many indoor environments [14, 32].

A heightfield is a 2D grid aligned with the ground plane, where each grid cell represents the maximum height of the scene's objects occupying that site. This manifests the prior that the man-made world is more spatially varying in the ground plane dimensions than it is in the height direction.

We introduce three main contributions: (1) A novel framework for top-down height estimation which approaches the performance of learning based full 3D reconstruction methods, while improving over a raw depth fusion baseline. (2) A novel blending method which learns when to trust raw fused geometry and when to use learned top-down outputs for the final output. (3) A new evaluation protocol that mimics an AR application, by placing 3D objects in the scene and measuring how they reproject into the image.

We show that our method is significantly faster and less memory intensive than other recent learning based full 3D scene reconstruction alternatives, while still remaining competitive in terms of reconstruction fidelity (Fig. 1). We evaluate on the challenging ScanNetV2 dataset [8] and show that we achieve state-of-the-art results on top-down mesh reconstruction and reprojected depth accuracy when compared to other top-down reconstruction baselines. Our method is also the best performing real-time method in our AR-style evaluation.

2. Related Work

Reconstructing 3D scenes. The reconstruction of 3D geometry from RGB images is a long-standing problem in computer vision. Structure from Motion (SfM) pipelines like COLMAP [43, 44] can be used to generate sparse reconstructions, while for dense reconstructions, traditional SfM relies on depth maps as an intermediate representation, *e.g*. from [36, 10, 6, 21]. Per-frame depths are then fused into a global representation using *e.g*. a Truncated Signed Distance Function (TSDF) [7, 31]. A similar depth map fusion strategy is used by other reconstruction methods [31, 36, 42]. Learning-based fusion methods for depth sensor data that improve on the traditional pipeline have also been recently proposed [53, 54, 56].

In contrast, more recent learning-based methods attempt to perform end-to-end reconstruction directly from RGB images, *e.g*. [30, 48]. Here, features extracted from each image are backprojected along the camera rays into a 3D feature volume, followed by expensive 3D convolutions on the 3D volume. The output is an implicit representation of the scene's 3D geometry. Unlike Atlas [30], which maintains a scene-level 3D feature volume, NeuralRecon [48] first performs local surface estimation by chopping the input scene into fragments which are later fused into a single global volume. This significantly speeds up processing speed, but still requires 3D convolutions. TransformerFusion [3] also proposes an incremental transformer-based online approach.

Our approach combines the best of both perspectives. We first predict a depth map for each image and fuse them using a TSDF. This raw TSDF is used to aggregate depths and features into a top-down grid. A lightweight network (*i.e*. no 3D convolutions) processes this grid and outputs a heightfield, addressing the speed and memory limitations of end-to-end methods like [30].

Top-down semantic reasoning. Top-down (or 'birds-eye-view') representations exploit the fact that objects and structures in the man-made world can often be approximated by their 2D locations on a planar surface. They can be effective in applications such as semantic segmentation of road scenes [40, 39, 35] and overhead view synthesis [58]. Different strategies have been proposed to convert from 2D image observations (*e.g*. captured by a camera approximately parallel to the ground) to the top-down space, both where depth is [5, 16] and isn't [27, 35, 40, 39] available. These methods are predominately concerned with top-down semantics. Instead, we show that similar 2½D representations can be used for lightweight 3D reconstruction.

Top-down geometric reasoning. Top-down representations have also been used for geometry-based reasoning. For example, height estimation from aerial monocular images is a common problem in remote sensing [47, 28]. Before deep learning, [11] fused estimated depth maps into a 2D grid from which a heightfield can be extracted. In our experiments, we show that a similar depth-only fusion baseline is inferior to our full method. In the context of SLAM, [59] perform dense reconstruction via a mesh-based heightfield representation. Their approach does not include a learning component, so the mesh is fitted at inference time via nonlinear optimization. [9] use a heightmap for efficiency, but do not attempt to improve upon the 3D reconstruction given as input. Stixels [2] model free space using a height-based representation, but represent the scene in camera-centric coordinates, unlike our fused coherent world-based representation.

In the context of learning-based approaches, [29] model the relationship between indoor walking trajectories and free space in indoor scenes, estimating a small number of discrete occupancy bins for each 2D location on the floor grid. This is related to the binary occupancy reasoning used in the neural SLAM method of [37]. [51] estimate free space and object 'footprints' in scenes, but do not estimate geometry. [45] perform *single-view* 3D scene reconstruction using an overhead height estimation network. However, the training supervision needed is non-trivial as it needs the occluded depth behind and beneath objects, thus synthetic data [46] is used. Finally, the convolutional single-plane decoder from [33] is related to our heightfield decoder, but we predict height directly rather than evalu-

ating multiple different 3D locations for each 2D position, resulting in less computation for our method.

Finally, there is also a large body of work that estimates floorplans for indoor environments from single images [22, 26], sequences [34], or 3D data [25, 17, 24]. While related, these methods are mostly concerned with estimating the shape of rooms, ignoring 'things,' *e.g.* furniture.

3. Problem Setup

Our goal is to convert successive posed color images into a height-based representation of the 3D world, while also minimizing the computation and memory footprint. The heightfield $\mathcal{H}$ is defined on a 2D grid $\mathcal{C}$ and assigns to each cell $c \in \mathcal{C}$ a height $\mathcal{H}_c \in [0, h_{\text{max}}]$, which corresponds to the maximum height of objects in the scene over the cell.

In practice, for navigation and augmented reality, we only care about heights up to a certain maximum height; we set this as a user-defined parameter h_{max} (*e.g.* in all our experiments we set $h_{\text{max}} = 1.5m$). This ensures we ignore the ceiling and top portions of walls in both training and evaluation for indoor scenes, but include furniture and other 'stuff'. Our 2D grid $\mathcal{C}$ is divided into cells of size $4cm \times 4cm$ in all experiments.

At both **training** and **test** time we assume access to a sequence of RGB images $\{I_i\}_{i=1}^N$, each of size (H, W) and with known camera poses $\{\Omega_i\}_{i=1}^N$ and intrinsics, all in a common 3D coordinate system roughly aligned relative to gravity. Poses can be estimated *e.g.* with [55, 57]. For our ScanNetV2 experiments we follow [30, 48, 3] in using ScanNet's pre-aligned poses. In the supplementary video, we show qualitative results of using HeightRecon for scenes with poses obtained using a visual-inertial odometry technique (Apple's ARKit [1]). At **training** time we also assume access to per-frame ground truth depth maps and to a ground truth heightfield of the scene.

4. Method

We estimate the heightfield $\mathcal{H}$ in four steps: (1) Estimate depth and deep features for each RGB input image; (2) Integrate estimated depths into a noisy, incomplete *3D* voxel grid $\mathcal{V}$; (3) Integrate the deep features into a top-down *2D* feature grid $\mathcal{F}$; (4) Create the final output $\mathcal{H}$ via a trained 2D CNN. A visual overview of these steps is shown in Fig. 2.

4.1. Per-image depth and feature estimation

We bootstrap our heightfield method with an unrefined estimate of the scene geometry. We do this via a pipeline of depth estimation followed by fusion of these depth maps into a common voxel volume. For each frame in the input sequence, we estimate a per-frame depth image D, of size (H, W). We estimate this using a supervised multi-view stereo (MVS) system, similar in spirit to [18, 52].

Low-level features are extracted from image I, as well as from nearby keyframe images. The geometric compatibility between the features of the reference and target images is computed in a cost volume [19] for depths between d_{min} and d_{max}. The cost volume is then convolved by a network to estimate a per-image depth D.

Depth estimation architecture. Our architecture is based on [52] but is trained with *supervised* data rather than self-supervision. We train with an $l1$-based loss and a gradient loss from [38]. More details of the architecture, hyperparameters, and how we select keyframes are given in the supplementary material.

Feature maps. Additionally, a feature map F is computed for each I. Similar to [30], we compute F as the sum of four residual blocks, each upsampled to (H, W). F has dimensionality K, where $K = 32$ in all our experiments.

4.2. Creating a noisy and incomplete raw TSDF $\mathcal{V}$

As a starting point for our final heightfield, we create an unrefined 'raw' fused volume, integrating the estimated depth maps from each image into a common reference frame. Following traditional depth fusion-based methods (*e.g.* [7, 31]), we integrate all depth maps into a voxel TSDF $\mathcal{V}$, using the known pose of each image. We set the bounds of this TSDF to a region around the camera center of the input image. At both training and test time, the center of this TSDF is the camera center of Ω_1. This TSDF would typically be voxel-hashed, making it extremely memory efficient. For ease of implementation, though, our implementation uses a dense tensor; this is still far more lightweight than learning-based baselines (see Sec. 5.6).

4.3. Creating the top-down feature map $\mathcal{F}$

As mentioned, prior works in deep 3D reconstruction (*e.g.* [30, 48]) project deep image features into a *3D* feature volume. However, building and processing such a volume is very expensive. Instead, we create a *top-down* feature map $\mathcal{F}$, where each cell contains image-level features corresponding to that grid location. This raises the questions of (i) how to best map features from images captured from arbitrary camera poses onto a 2D grid of the ground, and (ii) how to accumulate features from multiple such images? Previous works in semantic reasoning have proposed different approaches to this, *e.g.* [40, 35]. These approaches are better suited to automotive applications where the camera pose relative to the ground plane is more well behaved, *i.e.* at a constant height and angle to the ground plane. In our experiments, we show that these methods are inferior to our approach in the context of indoor scene reconstruction.

Naively, we could use our depths to project features from image space into 3D world space, before collapsing down to 2D (similar to [4], which we compare to as a baseline). However, this has the key limitation that it needs ground

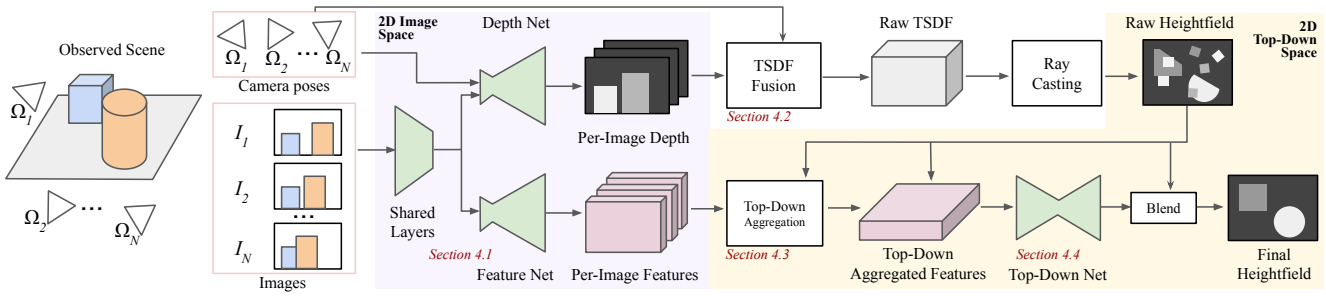


Figure 2. **Overview of our HeightRecon model**. Our method predicts a heightfield from a sequence of posed images. It starts by extracting depth maps and features maps for all the images in the sequence so far. The depth maps are then fused into a TSDF volume that is converted to a raw heightfield by raycasting from a virtual orthographic camera. This raw heightfield is used in our top-down feature aggregation step to convert the per-frame feature maps into a top-down aggregated feature map $\mathcal{F}$. The aggregated feature map and the raw heightfield are further processed by our top-down network to produce the final heightfield.

truth depths, without which noisy depth estimates will place features in the wrong grid cells. To better cope with the noise of depth estimates, we exploit the already-computed raw TSDF volume $\mathcal{V}$, which enables us to efficiently and accurately grab appropriate image-level features for each grid cell. First we convert the raw TSDF into a heightfield by raycasting the raw TSDF $\mathcal{V}$ from above, using a virtual orthographic camera. This gives a raw heightfield $\mathcal{H}^{\text{raw}}$. The height $\mathcal{H}^{\text{raw}}_c$ at each cell c is converted to a 3D point $\mathcal{P}_c$.

Gathering features from a single image. For each cell c in the 2D grid, we want to populate the top-down feature map $\mathcal{F}_i$ with features from F_i, the feature map for image I_i. To do this, we project each $\mathcal{P}_c$ into camera Ω_i using the known camera extrinsics and intrinsics. We determine if the point is occluded by comparing its coordinates with the estimated depth map. For the unoccluded points, we then sample the input image feature map F_i at the reprojected image coordinates, to obtain $\mathcal{F}_i$. We also maintain a visibility mask $\mathcal{T}_i$, where $\mathcal{T}_{ic}$ is 1 if $\mathcal{P}_c$ is directly visible and unoccluded in camera Ω_i, and 0 otherwise. This depth-based sampling is in contrast to methods like [30] which project image features to *all* voxels along the ray for a given pixel.

Aggregating a sequence of images into the grid. Having obtained a $\mathcal{F}_i$ and visibility mask $\mathcal{T}_i$ from each image i, the final feature map $\mathcal{F}$ is the average of the feature maps from each image, taking visibility into account, $\mathcal{F}_c = \sum_i \mathcal{T}_{ic}\mathcal{F}_{ic} / \sum_i \mathcal{T}_{ic}$. For online applications we compute this as a rolling average [31].

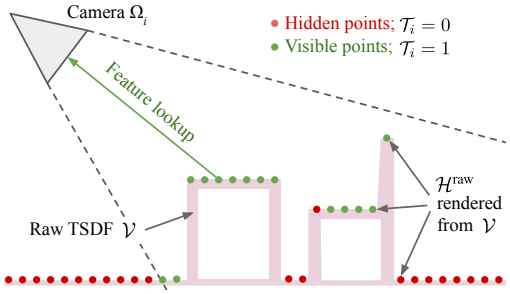


Figure 3. **Computation of $\mathcal{F}$ and $\mathcal{T}$.** Each 3D point in $\mathcal{H}^{\text{raw}}$ is projected into camera i, so image-level features can be sampled. $\mathcal{T}$ records if each point is visible (1) or hidden (0) in the camera.

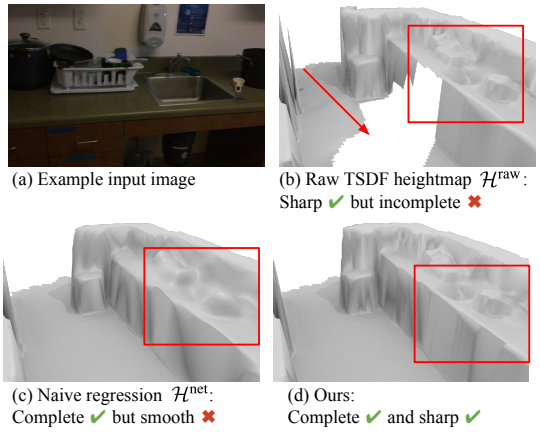


Figure 4. Naive TSDF volumes give sharp but incomplete geometry, while naive top-down network outputs are complete but blurry. Our novel blending approach (Sec. 4.4) borrows from both.

4.4. Top-down heightfield regression

The next step is to regress the heightfield of the surrounding environment from the top-down feature map, derived thus far from N cameras and averaged into per-cell features $\mathcal{F}_c$. Our top-down network takes as input the 2D feature map $\mathcal{F}$, and the top-down render of the raw TSDF, $\mathcal{H}^{\text{raw}}$. These are stacked together to make a $(K+1)$-channel tensor, where K is the number of feature channels in F. Our top-down network is a *2D* convolutional network which outputs a single-channel tensor representing the heightfield $\mathcal{H}$. $\mathcal{H}$ is predicted in an absolute scale in meters.

Learned blending for improved heightfields. A naive version of heightfield regression, given $\mathcal{F}$ and $\mathcal{H}^{\text{raw}}$ as input, would be to use a model similar to our depth regression network, *i.e.* a standard regression CNN, comprising an encoder, decoder, and skip connections. We found that com-

pared to the raw heightfield $\mathcal{H}^{\text{raw}}$, the final prediction from such a naive model would be more *complete*, but also far less *sharp*. Such a network learns to make sensible predictions, but at the expense of smoothing over discontinuities. We therefore form our final prediction $\mathcal{H}$ from a *blend* of the rendered $\mathcal{H}^{\text{raw}}$ and the network prediction $\mathcal{H}^{\text{net}}$. This blend is adjusted on a per-cell basis, modulated by a blend map Φ which is predicted as a second output channel from our top-down network. During training, we supervise Φ by asking the network to predict where $\mathcal{H}^{\text{net}}$ is closer to the ground truth than $\mathcal{H}^{\text{raw}}$. At inference time, we threshold Φ to make $\bar{\Phi}$ and take the final height value for a cell c to be

$$\mathcal{H}_c = \bar{\Phi}_c \, \mathcal{H}_c^{\text{net}} + (1 - \bar{\Phi}_c) \, \mathcal{H}_c^{\text{raw}}. \tag{1}$$

Our experiments show that blending results in qualitative and quantitative improvements over using just $\mathcal{H}^{\text{raw}}$ or $\mathcal{H}^{\text{net}}$ on their own; see Table 2 and Fig. 4.

Top-down losses. We use standard losses from depth estimation to predict accurate and sharp heightfields. We only apply the top-down loss on cells inside the convex hull of valid ground truth regions, to avoid supervising in regions where the training data is unreliable. Our loss is then

$$\mathcal{L}_{\text{heightfield}} = \frac{\sum_c \mathcal{M}_c |\, \mathcal{H}_c^{\text{net}} - \mathcal{H}_c^{\text{gt}} \,|}{\sum_c \mathcal{M}_c}, \tag{2}$$

where $\mathcal{M}$ is a binary mask, which is 1 for cells inside the convex hull of the ground truth mask and 0 otherwise. We also use a gradient matching loss [23, 38], to improve the sharpness of the top-down prediction, so

$$\mathcal{L}_{\text{grad}} = \sum_{c \in \mathcal{C}} |\nabla_x (\mathcal{H}_c^{\text{net}} - \mathcal{H}_c^{\text{gt}})| + |\nabla_y (\mathcal{H}_c^{\text{net}} - \mathcal{H}_c^{\text{gt}})|, \tag{3}$$

where ∇_x and ∇_y are 2D gradients. Our blending map Φ is supervised with a binary cross entropy loss,

$$\mathcal{L}_{\text{blend}} = \sum_{c \in \mathcal{C}} \text{BCE}(\Phi_c, |\mathcal{H}_c^{\text{net}} - \mathcal{H}_c^{\text{gt}}| < |\mathcal{H}_c^{\text{raw}} - \mathcal{H}_c^{\text{gt}}|). \tag{4}$$

We train with 256×256 crops, but at test time we use the network's fully convolutional capability to predict for arbitrary size scenes. Our final loss is $\mathcal{L} = \mathcal{L}_{\text{heightfield}} + \mathcal{L}_{\text{grad}} + \mathcal{L}_{\text{blend}}$, which we average over four output scales [12, 38]. We refer to our combined depth and heightfield regression model as 'HeightRecon'.

4.5. Online operation

For real-time operation, HeightRecon can be used to give always-on 'live' updates of the current estimate of the 3D scene. We keep three structures in memory during online processing: a set of keyframe feature maps for depth estimation, the raw TSDF volume, and the top-down aggregated feature map. When a new image frame is captured, we run the depth and feature extraction networks. The depth of this new frame is fused into the raw TSDF volume. The raw TSDF volume is converted into a raw heightfield, which is used to gather features for this new frame. The aggregated feature map $\mathcal{F}$ is updated with the features from the new frame and processed together with the raw heightfield by the top-down network, returning the final heightfield. We use fixed size grids in our experiments, but for very large scenes, $\mathcal{V}$ and $\mathcal{F}$ can easily be extended dynamically.

5. Experiments

We train and evaluate on the challenging ScanNetv2 dataset [8], which comprises 1,201 training, 312 validation, and 100 testing scans of indoor scenes, captured with a handheld RGBD sensor. At test time, HeightRecon does not have access to the depth channel, *i.e*. we use RGBD data for training and RGB for evaluation.

5.1. Implementation details

Our depth network is based on [52], which uses a ResNet-18 [15] backbone and decoder similar to [12, 50], and is trained using depth supervision. Depths in the cost volume are spaced linearly between $d_{\text{min}} = 0.1$m and $d_{\text{max}} = 10$m. See supplementary material for network specifics. At training time we use a TSDF $\mathcal{V}$ of dimensions $256 \times 256 \times 50$. For all our experiments we use voxels of size $(4\text{cm})^3$, and for our TSDFs we use a truncation parameter of $\tau = 20$cm. We train with a learning rate of 10^{-4} for 150k steps, using a batch size of 4, with the Adam optimizer [20]. RGB training images have the same augmentations as [12]. The residual blocks of the feature extractor are shared with the depth network. For faster training, we pretrain the depth network. When optimizing the heightfield loss $\mathcal{L}$ we keep the depth network fixed (and thus also the residual layers shared by the feature network), but the remaining layers of the feature network are trained end-to-end. Unless otherwise stated, our networks use ImageNet [41] pretrained weights for faster convergence. We create ground truth training and testing heightfields by raycasting the ground truth meshes from above, using an orthographic camera. At inference time we threshold the blend map Φ at 0.6.

5.2. Evaluation

We evaluate HeightRecon in three separate scenarios, reflecting our desire for a fast, accurate reconstruction which is suitable for AR character placement and navigation:

Heightfield mesh quality. We evaluate the heightfield-derived mesh quality of HeightRecon and compare it to 2D and 3D baselines using the 3D metrics from [30]. For the full 3D methods (*e.g*. [30, 48]), we first convert their 3D output to a heightfield by raycasting from above the meshes

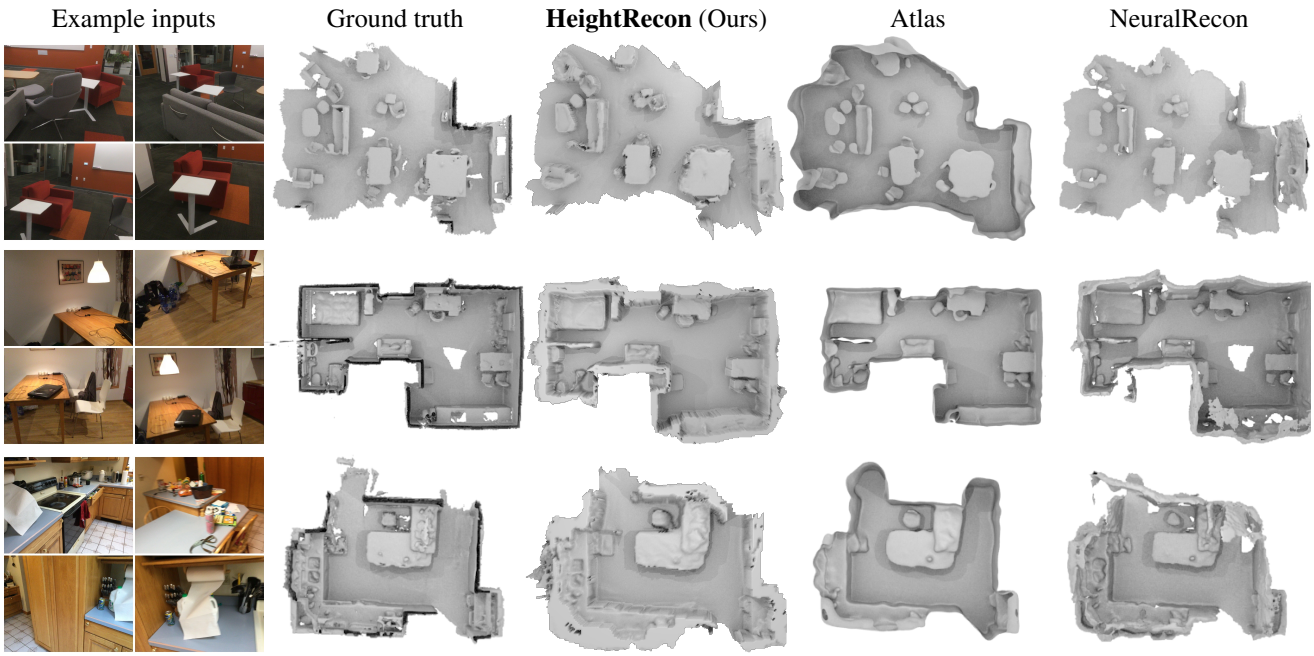


Figure 5. **Qualitative comparisons with full 3D reconstructions.** Our reconstructions are more complete than NeuralRecon [48] and are qualitatively comparable to Atlas [30], despite the fact that we only estimate 2D heightfields. Example input images are shown on the left.

obtained from their predicted TSDFs, or raycasting from the provided meshes directly for [3]. For all methods, at inference time we remove any geometry more than 1.5m above the minimum predicted height.

Augmented reality evaluation. To evaluate HeightRecon's effectiveness in an AR setting, we construct an evaluation protocol that measures two important aspects of AR: (1) reprojection error of 3D assets placed on predicted heightfields, and (2) IoU between estimated and ground truth heightfield derived navigation maps. This is described and evaluated in Sec. 5.5.

Full 3D evaluation and depth reprojection metrics. In the supplementary material, we evaluate our HeightRecon method using the full 3D mesh evaluation of [3], as well as their depth reprojection metrics.

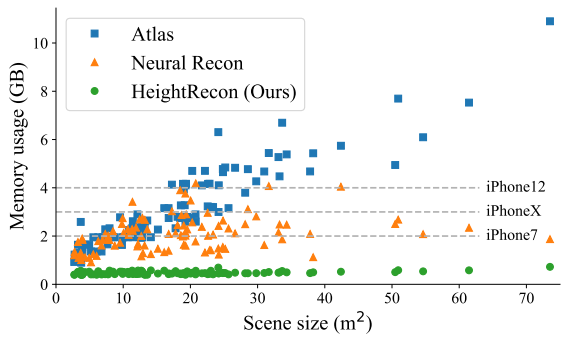
	Comp↓	Acc↓	Precision↑	Recall↑	F1↑
Non real-time methods					
Atlas [30]	.056	**.060**	.767	**.735**	**.750**
NeuralRecon [48]	**.039**	.075	**.768**	.657	.706
TransformerFusion [3]	.041	.062	.748	.695	.720
Real-time methods					
Lift splat shoot [35]	.137	.193	.243	.232	.237
OFT single [40]	.111	.183	.357	.340	.348
OFT multi [40]	.085	.118	.453	.446	.450
Mapnet [5]	.091	.105	.478	.471	.474
Mapnet [5] + $\mathcal{H}^{\text{raw}}$	.093	.106	.481	.475	.478
Raw TSDF ($\mathcal{H}^{\text{raw}}$)	.065	.066	.604	.592	.598
HeightRecon (Ours)	**.052**	**.057**	**.684**	**.677**	**.680**

Table 1. **Reconstruction evaluation of heightfield meshes.** We outperform other real-time baselines, and are competitive with non real-time full-3D methods, for the task of top-down mesh evaluation, where the ground truth heightfield is computed from the full 3D volume. See Sec. 5.3 for details.

5.3. Comparison to 2D and 3D baselines

We compare to three recent state-of-the-art methods that reason in 3D: Atlas [30], NeuralRecon [48] and TransformerFusion [3]. All use 3D convolutions and are not real-time. We also compare to methods which, like us, use a 2D top-down representation [35, 40, 5]. These are designed for semantic segmentation rather than heightfields, so we use their approaches for projecting from camera space to top-down space, but predict heights using our top-down network, trained using $\mathcal{L}_{\text{heightfield}} + \mathcal{L}_{\text{grad}}$. We provide a detailed description of how we implemented these baselines in the supplementary material.

Heightfield mesh quality prediction results in Table 1 show that our approach outperforms alternative 2D top-down methods, and is competitive with more expensive methods that do full 3D reasoning. We also compare with the raw TSDF, which for this evaluation is converted into a heightfield and thus corresponds to $\mathcal{H}^{\text{raw}}$, one of the inputs to our top-down network. HeightRecon improves over this baseline demonstrating the value of our top-down network in refining the noisy and incomplete raw TSDF. Qualitative results in Fig. 5 show that HeightRecon produces similar, and sometimes even superior results relative to expensive 3D baselines.



	Mean (MB)	Max (MB)	Min (MB)
Atlas [30]	3221	10900	896
Neural Recon [48]	2077	4184	910
HeightRecon (Ours)	**475**	**724**	**386**

Figure 6. **Our memory usage is significantly lower than full 3D methods.** (Top) Each dot represents a single ScanNetV2 test scene. The x-axis is the size of the scene in m^2, as measured by occupied 2D cells in the ground truth volume. Horizontal lines indicate the maximum available memory for some popular mobile devices. However, memory allocation is at the discretion of the OS and some will also be reserved for application logic. (Below) The memory usage in megabytes on the ScanNetv2 test set.

5.4. Ablation

We validate our contributions by turning them on and off in Table 2. We note that our full HeightRecon method performs better than all the following variants:

$\mathcal{H}^{\text{net}}$: The heightfield directly regressed by our network, without our learned blending.

Ours w/o $\mathcal{H}^{\text{raw}}$: HeightRecon, but the top-down network only gets access to $\mathcal{F}$ and not $\mathcal{H}^{\text{raw}}$.

Ours w/o $\mathcal{F}$: HeightRecon, but the top-down network only gets access to $\mathcal{H}^{\text{raw}}$ and not $\mathcal{F}$.

Ours w/o $\mathcal{V}$: HeightRecon, without the raw TSDF step, where the raw heightfield is computed directly from depths projected to the top-down space.

The effect of depth estimation. We also report in Table 2 the performance of HeightRecon when using depth maps obtained with the computationally more expensive DVMVS [10]. For this experiment, we do not retrain the top-down model, and instead we just evaluate our model using different depth maps as input. Our performance is improved when using these higher quality depth maps, at the cost of increased computational time (Fig. 8). More importantly, our top-down network still leads to improvements over the TSDF mesh computed using [10]. For further evaluation using depth from a monocular network and ground truth sensor depths, please see the supplementary material.

5.5. Augmented reality-style evaluation

HeightRecon enables real-time prediction of 3D reconstructions in AR-style applications. We evaluate this in two ways. First, we simulate the placement of 3D objects in our reconstructed 3D scenes and quantify how closely the reprojected visible object masks match the ones resulting from placing the same object in the ground truth 3D reconstructions. The metric takes into account the quality of placement, as well as occlusion. For every tenth frame, we randomly select a valid top-down position on the ground truth mesh, place a cuboid there, and place one at the same location in the predicted mesh. Performance is measured in image space, using the 2D IoU between the reprojected visibility mask of the cuboid from the estimated reconstruction versus the ground truth reprojection, taking scene occluders into account. We refer to this as **Rendering IoU**.

Second, we convert the predicted heightfield into an AR navigation map by comparing the heights of nearby cells in a sliding window. We then compute the 2D IoU between the ground truth navigation map and the predicted navigation map; which we refer to as **Placement IoU**.

Full details for this protocol are provided in the supplementary material. In Fig. 8 we observe that HeightRecon greatly improves over the raw TSDF, and is close to 3D reasoning methods, while only requiring a fraction of the computation resources. Additionally, Fig. 7 shows that the navigation maps for HeightRecon are far more complete and noise free than the raw TSDF, coming very close to the quality of the far more computationally expensive Atlas [30].

	Comp↓	Acc↓	Prec.↑	Recall↑	F1↑
$\mathcal{H}^{\text{net}}$	.058	.072	.637	.622	.629
Ours w/o $\mathcal{H}^{\text{raw}}$	.065	.071	.611	.606	.609
Ours w/o $\mathcal{F}$	.070	.084	.531	.520	.526
Ours w/o $\mathcal{V}$	.069	.079	.544	.539	.541
HeightRecon	**.052**	**.057**	**.684**	**.677**	**.680**
DVMVS [10]	.051	.062	.680	.657	.668
HeightRecon (DVMVS depth)	**.049**	**.053**	**.720**	**.718**	**.719**

Table 2. **Our contributions lead to better results**, as evaluated on our 'heightfield mesh quality' task. Turning off each of our contributions in turn degrades performance; see Sec. 5.4 for details. In the bottom section we show that better depths (*e.g.* here from DVMVS) can increase scores, even without retraining our model.

5.6. Run-time efficiency

Fig. 6 shows how much less memory our 2½D HeightRecon uses when compared to full 3D deep reconstruction methods. Atlas [30] has the highest peak memory usage, as it needs a full 3D feature grid spanning the whole scene size. NeuralRecon [48] computes voxel reconstructions in fragments, which saves memory. However our 2D feature map and lightweight 2D convolutions mean that ours is the most memory efficient of the three. See the supplementary material for how we computed the memory.

For real-time applications, we need quick updating of the scene shape as new frames come in. While some 3D baselines take several hundred milliseconds to integrate a new RGB frame and produce the updated 3D reconstruction, in

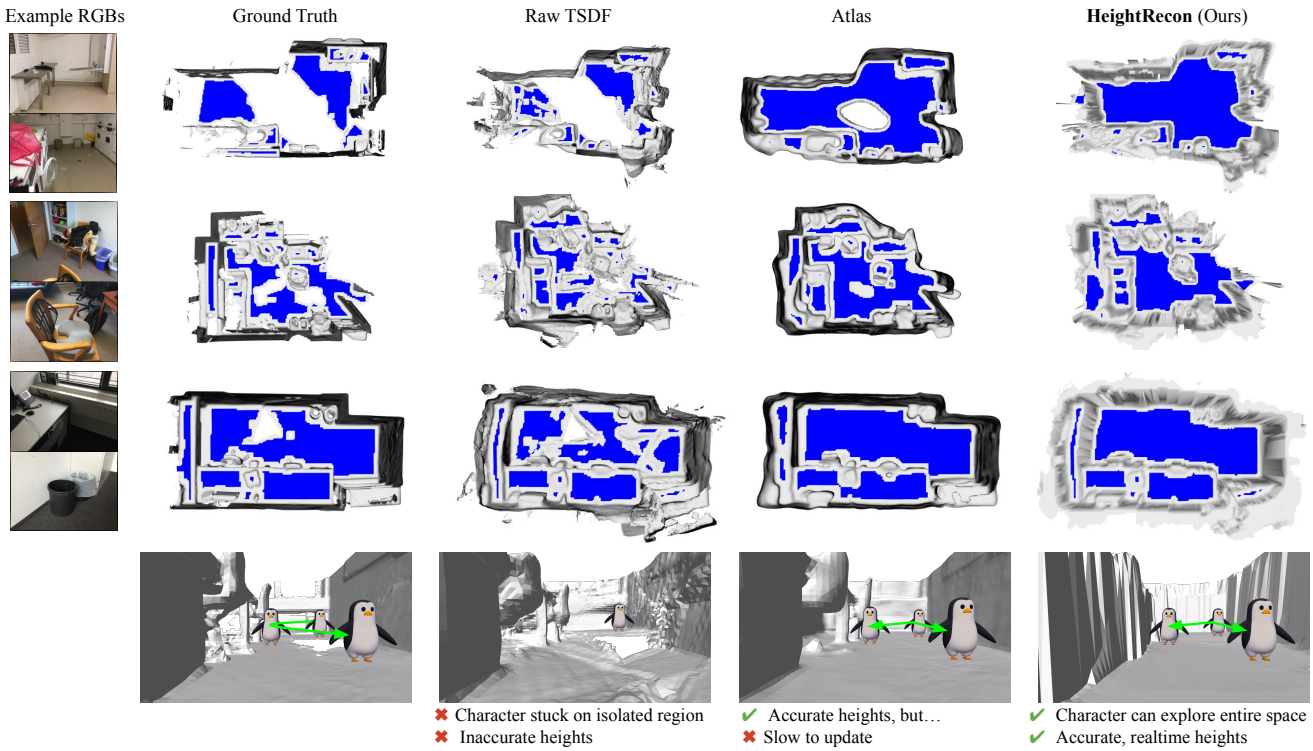


Figure 7. **Heightfields allows for usable character placement and navigation.** (Top) Navigation meshes, in blue, are computed from the 3D reconstructions. The raw TSDF is noisy and incomplete, so the character cannot explore the whole room. Atlas [30] is high quality, but not suitable for real-time use. Our Heightfields can be predicted in real time, and they allow for good navigation and placement, including recovery of walkable regions which are missing in the ground truth. (Bottom) Our predictions enable AR character navigation.

	Rendering	Placement	
	IoU↑	IoU↑	Time↓
Non real-time methods			
Atlas [30]	**0.889**	**0.768**	382
NeuralRecon [48]	0.887	0.673	196*
TransformerFusion [3]	0.872	0.678	304*
Real-time methods			
Raw TSDF	0.799	0.628	20
DVMVS [10]	0.817	0.621	40*
HeightRecon (Ours)	**0.838**	**0.727**	33
HeightRecon (DVMVS depth)	**0.866**	**0.740**	53

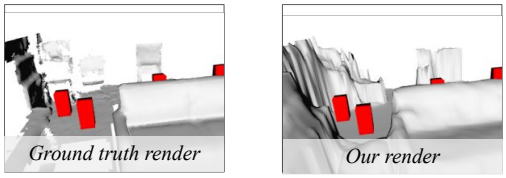


Figure 8. **HeightRecon allows for accurate AR object placement and rendering in realtime.** We evaluate the quality of augmented object rendering and placement, in terms of 2D IoU, on ScanNetv2. An example evaluation render is on the bottom. See Sec. 5.5 for details. We also show the time taken per update in milliseconds running on a NVIDIA GTX 1080Ti; time estimates marked with * are taken from the original papers. See Sec. 5.6.

Fig. 8 we see that HeightRecon is fast (∼33ms to make an updated 3D reconstruction or ∼30 fps, on an NVIDIA GTX 1080Ti), and can thus be run in real-time with low memory overhead. See also Fig. 1 for a graph of update time vs. accuracy, showing we are more accurate than competing real-time methods. In the supplementary material, we include a video showing online reconstructions using our method.

6. Conclusion

We presented HeightRecon, a novel method for predicting heightfields from a series of posed color images. We compared to multiple baselines and showed that we outperform all baselines which predict in top-down space, including our target applications of AR character placement and rendering. We produce comparable results to full 3D methods, but run at a fraction of both memory and compute, enabling real-time interactive applications.

Limitations. By definition, heightfields are unable to reconstruct the undersides of objects such as tables, or overhanging structures such as kitchen cupboards or wall lights. This could potentially be addressed using a layer representation (*e.g.* [49, 45]), but we leave this for future work. We are also limited by the diversity of the training data. ScanNet [8] mostly comprises western-style homes, and thus HeightRecon may perform poorly in other environments.

References

[1] Apple: ARKit, https://developer.apple.com/documentation/arkit, Accessed: 12 July 2022

[2] Badino, H., Franke, U., Pfeiffer, D.: The stixel world — a compact medium level representation of the 3D-world. In: Joint Pattern Recognition Symposium (2009)

[3] Božič, A., Palafox, P., Thies, J., Dai, A., Nießner, M.: Transformerfusion: Monocular RGB scene reconstruction using transformers. In: NeurIPS (2021)

[4] Brahmbhatt, S., Gu, J., Kim, K., Hays, J., Kautz, J.: Geometry-aware learning of maps for camera localization. In: CVPR (2018)

[5] Cartillier, V., Ren, Z., Jain, N., Lee, S., Essa, I., Batra, D.: Semantic mapnet: Building allocentric semanticmaps and representations from egocentric views. In: AAAI (2021)

[6] Choe, J., Im, S., Rameau, F., Kang, M., Kweon, I.S.: VolumeFusion: Deep depth fusion for 3D scene reconstruction. In: ICCV (2021)

[7] Curless, B., Levoy, M.: A volumetric method for building complex models from range images. In: Computer graphics and interactive techniques (1996)

[8] Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In: CVPR (2017)

[9] De Gregorio, D., Di Stefano, L.: SkiMap: An efficient mapping framework for robot navigation. In: ICRA (2017)

[10] Duzceker, A., Galliani, S., Vogel, C., Speciale, P., Dusmanu, M., Pollefeys, M.: DeepVideoMVS: Multi-view stereo on video with recurrent spatio-temporal fusion. In: CVPR (2021)

[11] Gallup, D., Frahm, J.M., Pollefeys, M., Zuerich, E.: A heightmap model for efficient 3D reconstruction from street-level video. In: 3DPVT (2010)

[12] Godard, C., Mac Aodha, O., Firman, M., Brostow, G.J.: Digging into self-supervised monocular depth estimation. In: ICCV (2019)

[13] Google: ARCore, https://developers.google.com/ar, Accessed: 12 July 2022

[14] Häne, C., Zach, C., Lim, J., Ranganathan, A., Pollefeys, M.: Stereo depth map fusion for robot navigation. In: IROS (2011)

[15] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)

[16] Henriques, J.F., Vedaldi, A.: MapNet: An allocentric spatial memory for mapping environments. In: CVPR (2018)

[17] Ikehata, S., Yang, H., Furukawa, Y.: Structured indoor modeling. In: ICCV (2015)

[18] Im, S., Jeon, H.G., Lin, S., Kweon, I.S.: DPSNet: End-to-end deep plane sweep stereo. In: ICLR (2019)

[19] Kendall, A., Martirosyan, H., Dasgupta, S., Henry, P., Kennedy, R., Bachrach, A., Bry, A.: End-to-end learning of geometry and context for deep stereo regression. In: ICCV (2017)

[20] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv:1412.6980 (2014)

[21] Koestler, L., Yang, N., Zeller, N., Cremers, D.: Tandem: Tracking and dense mapping in real-time using deep multi-view stereo. In: CoRL (2021)

[22] Kruzhilov, I., Romanov, M., Babichev, D., Konushin, A.: Double refinement network for room layout estimation. In: ACCV (2019)

[23] Li, Z., Snavely, N.: Megadepth: Learning single-view depth prediction from internet photos. In: CVPR (2018)

[24] Lin, C., Li, C., Wang, W.: Floorplan-jigsaw: Jointly estimating scene layout and aligning partial scans. In: ICCV (2019)

[25] Liu, C., Wu, J., Furukawa, Y.: FloorNet: A unified framework for floorplan reconstruction from 3D scans. In: ECCV (2018)

[26] Liu, C., Schwing, A.G., Kundu, K., Urtasun, R., Fidler, S.: Rent3D: Floor-plan priors for monocular layout estimation. In: CVPR (2015)

[27] Mani, K., Daga, S., Garg, S., Narasimhan, S.S., Krishna, M., Jatavallabhula, K.M.: MonoLayout: Amodal scene layout from a single image. In: WACV (2020)

[28] Mou, L., Zhu, X.X.: Im2height: Height estimation from single monocular imagery via fully residual convolutional-deconvolutional network. arXiv:1802.10249 (2018)

[29] Mura, C., Pajarola, R., Schindler, K., Mitra, N.: Walk2Map: Extracting floor plans from indoor walk trajectories. In: Computer Graphics Forum (2021)

[30] Murez, Z., van As, T., Bartolozzi, J., Sinha, A., Badrinarayanan, V., Rabinovich, A.: Atlas: End-to-end 3D scene reconstruction from posed images. In: ECCV (2020)

[31] Newcombe, R.A., Izadi, S., Hilliges, O.: Kinectfusion: Real-time dense surface mapping and tracking. In: UIST (2011)

[32] Oda, O., Lister, L.J., White, S., Feiner, S.: Developing an augmented reality racing game. In: INtelligent TEchnologies for interactive enterTAINment (2008)

[33] Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., Geiger, A.: Convolutional occupancy networks. In: ECCV (2020)

[34] Phalak, A., Chen, Z., Yi, D., Gupta, K., Badrinarayanan, V., Rabinovich, A.: DeepPerimeter: Indoor boundary estimation from posed monocular sequences. arXiv:1904.11595 (2019)

[35] Philion, J., Fidler, S.: Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3D. In: ECCV (2020)

[36] Pradeep, V., Rhemann, C., Izadi, S., Zach, C., Bleyer, M., Bathiche, S.: MonoFusion: Real-time 3D reconstruction of small scenes with a single web camera. In: ISMAR (2013)

[37] Ramakrishnan, S.K., Al-Halah, Z., Grauman, K.: Occupancy anticipation for efficient exploration and navigation. In: ECCV (2020)

[38] Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., Koltun, V.: Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. TPAMI (2020)

[39] Roddick, T., Cipolla, R.: Predicting semantic map representations from images using pyramid occupancy networks. In: CVPR (2020)

[40] Roddick, T., Kendall, A., Cipolla, R.: Orthographic feature transform for monocular 3D object detection. In: BMVC (2019)

[41] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. IJCV (2015)

[42] Sayed, M., Gibson, J., Watson, J., Prisacariu, V., Firman, M., Godard, C.: SimpleRecon: 3D reconstruction without 3D convolutions. In: CVPR (2022)

[43] Schonberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: CVPR (2016)

[44] Schönberger, J.L., Zheng, E., Frahm, J.M., Pollefeys, M.: Pixelwise view selection for unstructured multi-view stereo. In: ECCV (2016)

[45] Shin, D., Ren, Z., Sudderth, E.B., Fowlkes, C.C.: 3D scene reconstruction with multi-layer depth and epipolar transformers. In: ICCV (2019)

[46] Song, S., Yu, F., Zeng, A., Chang, A.X., Savva, M., Funkhouser, T.: Semantic scene completion from a single depth image. In: CVPR (2017)

[47] Srivastava, S., Volpi, M., Tuia, D.: Joint height estimation and semantic labeling of monocular aerial images with CNNs. In: International Geoscience and Remote Sensing Symposium (2017)

[48] Sun, J., Xie, Y., Chen, L., Zhou, X., Bao, H.: NeuralRecon: Real-time coherent 3d reconstruction from monocular video. In: CVPR (2021)

[49] Tulsiani, S., Tucker, R., Snavely, N.: Layer-structured 3D scene inference via view synthesis. In: ECCV (2018)

[50] Watson, J., Firman, M., Brostow, G.J., Turmukhambetov, D.: Self-supervised monocular depth hints. In: ICCV (2019)

[51] Watson, J., Firman, M., Monszpart, A., Brostow, G.J.: Footprints and free space from a single color image. In: CVPR (2020)

[52] Watson, J., Mac Aodha, O., Prisacariu, V., Brostow, G.J., Firman, M.: The temporal opportunist: Self-supervised multi-frame monocular depth. In: CVPR (2021)

[53] Weder, S., Schonberger, J., Pollefeys, M., Oswald, M.R.: Routedfusion: Learning real-time depth map fusion. In: CVPR (2020)

[54] Weder, S., Schonberger, J.L., Pollefeys, M., Oswald, M.R.: Neuralfusion: Online depth fusion in latent space. In: CVPR (2021)

[55] Xian, W., Li, Z., Fisher, M., Eisenmann, J., Shechtman, E., Snavely, N.: UprightNet: Geometry-aware camera orientation estimation from single images. In: ICCV (2019)

[56] Yan, Z., Tian, Y., Shi, X., Guo, P., Wang, P., Zha, H.: Continual neural mapping: Learning an implicit scene representation from sequential observations. In: ICCV (2021)

[57] Zhao, Y., Kong, S., Fowlkes, C.: Camera pose matters: Improving depth prediction by mitigating pose distribution bias. In: CVPR (2021)

[58] Zhu, X., Yin, Z., Shi, J., Li, H., Lin, D.: Generative adversarial frontal view to bird view synthesis. In: 3DV (2018)

[59] Zienkiewicz, J., Davison, A., Leutenegger, S.: Real-time height map fusion using differentiable rendering. In: IROS (2016)