

# Recognizing Scenes from Novel Viewpoints

Shengyi Qian<sup>‡</sup> Alexander Kirillov<sup>†</sup> Nikhila Ravi<sup>†</sup> Devendra Singh Chaplot<sup>†</sup>

Justin Johnson<sup>‡,†</sup> David F. Fouhey<sup>‡</sup> Georgia Gkioxari<sup>†</sup>

<sup>†</sup> Facebook AI Research

<sup>‡</sup> University of Michigan, Ann Arbor

## Abstract

Humans can perceive scenes in 3D from a handful of 2D views. For AI agents, the ability to recognize a scene from any viewpoint given only a few images enables them to efficiently interact with the scene and its objects. In this work, we attempt to endow machines with this ability. We propose a model which takes as input a few RGB images of a new scene and recognizes the scene from novel viewpoints by segmenting it into semantic categories. All this without access to the RGB images from those views. We pair 2D scene recognition with an implicit 3D representation and learn from multi-view 2D annotations of hundreds of scenes without any 3D supervision beyond camera poses. We experiment on challenging datasets and demonstrate our model’s ability to jointly capture semantics and geometry of novel scenes with diverse layouts, object types and shapes.<sup>1</sup>

## 1. Introduction

Humans can build a rich understanding of scenes from a handful of images. The pictures in Fig. 1, for instance, let us imagine how the objects would occlude, disocclude, and change shape as we walk around the room. This skill is useful in new environments, for example if one were at a friend’s house, looking for a table to put a cup on. One can reason that a side table may be by the couch without first mapping the room or worrying about what color the side table is. As one walks into a room, one can readily sense floors behind objects and chair seats behind tables, etc. This ability is so intuitive that entire industries like hotels and real estate depend on persuading users with a few photos.

The goal of this paper is to give computers the same ability. In AI, this skill allows autonomous agents to purposefully and efficiently interact with the scene and its objects bypassing the expensive step of mapping. AR/VR and graphics also benefit from 3D scene understanding. To this end, we propose to learn a 3D representation that enables machines to recognize a scene by segmenting it into seman-

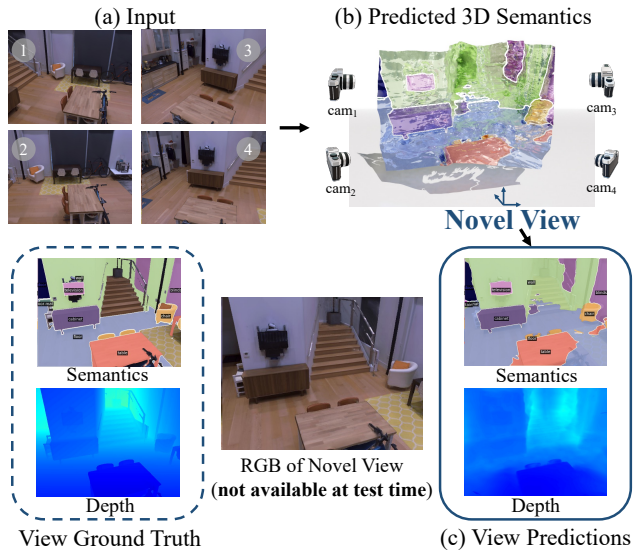



Figure 1. We propose ViewSeg which takes as input (a) a few images of a novel scene, and recognizes the scene from novel viewpoints. The novel viewpoint, in the form of camera coordinates, queries (b) our learnt 3D representation to produce (c) semantic segmentations from the view *without access to the view’s RGB image*. The view query additionally produces (c) depth. ViewSeg trains on hundreds of scenes using multi-view 2D annotations and no 3D supervision. Depth colormap: 0.1m  20m.

tic categories from *novel views*. Each novel view, provided in the form of camera coordinates, queries the learnt 3D representation to produce a semantic segmentation of the scene from that view *without access to the view’s RGB image*, as we show in Fig. 1. We aim to learn this 3D representation without onerous 3D supervision.

Making progress on the problem in complex indoor scenes requires connecting three key areas of computer vision: semantic understanding (i.e., naming objects), 3D understanding, and novel view synthesis (NVS). This puts the problem out of reach of work in any one area. For instance, there have been substantial advances in NVS, including methods like NeRF [45] or SynSin [65]. These approaches perform well on NVS, including results in new scenes [65, 70], but are focused on only appearance. In ad-

<sup>1</sup>Project page: <https://jasonqsy.github.io/viewseg>.

dition to numerous semantic-specific details, recognition in novel viewpoints via direct appearance synthesis is suboptimal: one may be sure of the presence of a rug behind a couch, but unsure of its particular color. Similarly, there have been advances in learning to infer 3D properties of scenes from image cues [20, 46, 63], or with differentiable rendering [10, 29, 38, 50] and other methods for bypassing the need for direct 3D supervision [27, 33, 34, 68]. However, these approaches do not connect to complex scene semantics; they primarily focus on single objects or small, less diverse 3D annotated datasets. More importantly, we empirically show that merely using learned 3D to propagate scene semantics to the new view is insufficient.

We tackle the problem of predicting scene semantics from novel viewpoints, which we refer to as *novel view semantic understanding*, and propose a new end-to-end model, ViewSeg. ViewSeg fuses advances and insights from semantic segmentation, 3D understanding, and novel view synthesis with a number of task-specific modifications. As input, our model takes a few posed RGB images from a previously unseen scene and a target pose (but *not* image). As output, it recognizes and segments objects from the target view by producing a semantic segmentation map. As an additional product, it also produces depth in the target view. During training, ViewSeg depends only on posed 2D images and semantic segmentations, and in particular does not use ground truth 3D annotations beyond camera pose.

Our experiments validate ViewSeg’s contributions on two challenging datasets, Hypersim [52] and Replica [62]. We substantially outperform alternate framings of the problem that build on the state-of-the-art: image-based NVS [70] followed by semantic segmentation [8], which tests whether NVS is sufficient for the task; and lifting semantic segmentations [8] to 3D and differentiable rendering, like [65], which tests the value of using implicit functions to tackle the problem. Our ablations further quantify the value of our problem-specific design. Among others, they reveal that ViewSeg trained for novel view semantic segmentation obtains more accurate depth predictions compared to a variant which trains without a semantic loss and image-based NVS [70], indicating that semantic understanding from novel viewpoints positively impacts geometry understanding. Overall, our results demonstrate ViewSeg’s ability to jointly capture the semantics and geometry of unseen scenes when tested on new, complex scenes with diverse layouts, object types, and shapes.

## 2. Related Work

For the task of novel view semantic understanding, we draw from 2D scene understanding, novel view synthesis and 3D learning to recognize scenes from novel viewpoints.

**Semantic Segmentation.** Segmenting objects and stuff from images is extensively researched. Initial efforts ap-

ply Bayesian classifiers on local features [32] or perform grouping on low-level cues [56]. Others [5, 13] score bottom-up mask proposals [1, 6]. With the advent of deep learning, FCNs [40] perform per-pixel segmentation with a CNN. DeepLab [7] use atrous convolutions and an encoder-decoder architecture [8] to handle scale and resolution.

Regarding multi-view semantic segmentation, [35] improve the temporal consistency of semantic segmentation in videos by linking frames with optical flow and learned feature similarity. [44] map semantic segmentations from RGBD inputs on 3D reconstructions from SLAM. [23] fuse predictions from video frames using super-pixels and optical flow. [42] learn scene dynamics to predict semantic segmentations of future frames given several past frames.

**Novel View Synthesis.** Novel view synthesis is a popular topic of research in computer vision and graphics. [19, 60, 61, 64, 67, 74] show great results synthesizing views from two or more narrow baseline images. Implicit voxel representations have been used to fit a scene from many scene views [39, 57, 59]. Recently, NeRF [45] learn a continuous volumetric scene function which emits density and radiance at spatial locations and show impressive results when fitted on a single scene with hundreds of views. We extend NeRF to emit a distribution over semantic categories at each 3D location. Semantic-NeRF [71] also predicts semantic classes. We differ from [71] as we generalize to novel scenes from sparse input views instead of in-place interpolation within a single scene from hundreds of views. NeRF extensions [24, 51], such as PixelNeRF [70], generalize to novel scenes from few input views with the help of learnt CNNs but show results on single-object benchmarks for RGB synthesis. We differ from [70] by carefully pairing a geometry-aware model with state-of-the-art scene recognition [8] and experiment on realistic multi-object scenes.

**3D Reconstruction from Images.** Scene reconstruction from multiple views is traditionally tackled with classical binocular stereo [21, 54] or with the help of shape priors [2, 3, 14, 25]. Modern techniques learn disparity from image pairs [30], estimate correspondences with contrastive learning [55], perform multi-view stereopsis via differentiable ray projection [28] or learn to reconstruct scenes while optimizing for cameras [26, 48]. Differentiable rendering [10, 29, 36, 38, 41, 47, 50] allows gradients to flow to 3D scenes via 2D re-projections. [10, 29, 38, 50] reconstruct single objects from a single view via rendering from 2 or more views during training. We also use differentiable rendering to learn 3D via 2D re-projections in semantic space.

**Depth Estimation from Images.** Recent methods train networks to predict depth from videos [11, 43, 73] or 3D supervision [9, 17, 37, 49, 69]. We do not use depth supervision but predict depth from novel views via training for semantic and RGB reconstruction from sparse inputs.



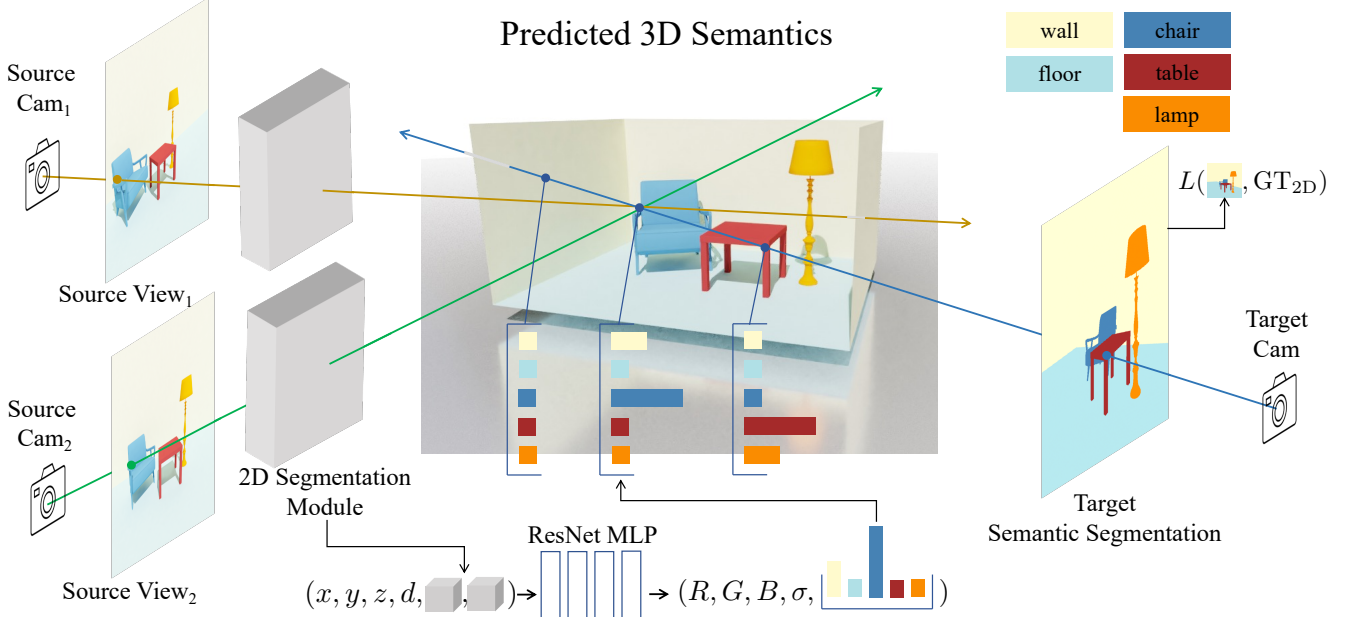


Figure 2. Our model, ViewSeg, uses a few source RGB views of a novel scene (here 2) and predicts the semantic segmentation of that scene from a given unseen target viewpoint. Our approach embeds each source view into a latent semantic space with a 2D segmentation module; this space is used to predict radiance, density and distribution over semantic classes at each spatial 3D location via an MLP. The final semantic segmentation is created by volumetric rendering from the target viewpoint. We train with posed multi-view 2D object annotations and no 3D supervision. We generalize to unseen scenes by training on hundreds of diverse scenes and thousands of source-target pairs.

### 3. Approach

We tackle novel view semantic understanding end-to-end with a new model, ViewSeg. ViewSeg takes as input RGB images from  $N$  *source* views and segments objects and stuff from a novel *target* viewpoint without access to the target image. We pair semantic segmentation with an implicit 3D representation to learn semantics and geometry from hundreds of scenes and thousands of source-target pairs. An overview of our approach is shown in Fig. 2.

Our model consists of a 2D semantic segmentation module which embeds each source view to a higher dimensional semantic space. An implicit 3D representation [45] samples features from the output of the segmentation module and predicts radiance, density and a distribution over semantic categories at each spatial 3D location with an MLP. The 3D predictions are projected to the target view to produce the segmentation from that view via volumetric rendering.

#### 3.1. Semantic Segmentation Module

The role of the semantic segmentation module is to project each source RGB view to a learnt feature space, which will subsequently be used to make 3D semantic predictions. Our 2D segmentation backbone takes as input an image  $I$  and outputs a spatial map  $M$  of the same spatial resolution,  $b^{\text{seg}} : I^{H \times W \times 3} \rightarrow M^{H \times W \times K}$ , using a convolutional neural network (CNN). Here,  $K$  is the dimension of the feature space ( $K = 256$ ).

We build on the state-of-the-art for single-view semantic segmentation and follow the encoder-decoder DeepLabv3+ [7, 8] architecture which processes an image by downsampling and then upsampling it to its original resolution with a sequence of convolutions. We remove the final layer, which predicts class probabilities, and use the output from the penultimate layer.

We initialize our segmentation module by pre-training on ADE20k [72], a dataset of over 20k images with semantic annotations. We empirically show the impact of the network architecture and pre-training in our experiments (Sec. 4).

#### 3.2. Semantic 3D Representation

To recognize a scene from novel viewpoints, we learn a 3D representation which predicts the semantic class for each 3D location. To achieve this we learn a function  $f$  which maps semantic features from our segmentation module to distributions over semantic categories conditioned on the 3D coordinates of each spatial location.

Assume  $N$  source views  $\{I_j\}_{j=1}^N$  and corresponding cameras  $\{\pi_j\}_{j=1}^N$ . For each view we extract the semantic maps  $\{M_j\}_{j=1}^N$  with our 2D segmentation module, or  $M_j = b^{\text{seg}}(I_j)$ . We project every 3D point  $\mathbf{x}$  to the  $j$ -th view with the corresponding camera transformation,  $\pi_j(\mathbf{x})$ , and then sample the  $K$ -dimensional feature vector from  $M_j$  from the projected 2D location. This yields a semantic feature from the  $j$ -th image denoted as  $\phi_j^{\text{seg}}(\mathbf{x}) = M_j(\pi_j(\mathbf{x}))$ .

Following NeRF [45] and PixelNeRF [70],  $f$  takes as input a positional embedding of the 3D coordinates of  $\mathbf{x}$ ,  $\gamma(\mathbf{x})$ , and the viewing direction  $\mathbf{d}$ . We additionally feed the semantic embeddings  $\{\phi_j^{\text{seg}}\}_{j=1}^N$ . As output,  $f$  produces

$$(\mathbf{c}, \sigma, \mathbf{s}) = f(\gamma(\mathbf{x}), \mathbf{d}, \phi_0^{\text{seg}}(\mathbf{x}), \dots, \phi_{N-1}^{\text{seg}}(\mathbf{x})) \quad (1)$$

where  $\mathbf{c} \in \mathbb{R}^3$  is the RGB color,  $\sigma \in \mathbb{R}$  is the occupancy, and  $\mathbf{s} \in \mathbb{R}^{|\mathcal{C}|}$  is the distribution over semantic classes  $\mathcal{C}$ .

We model  $f$  as a fully-connected ResNet [22], similar to PixelNeRF [70]. The positional embedding of the 3D location and the direction are concatenated to each semantic embedding  $\phi_j^{\text{seg}}$ , and each is passed through 3 residual blocks with 512 hidden units. The outputs are subsequently aggregated using average pooling and used to predict the final outputs of  $f$  via two branches: one predicts the semantics  $\mathbf{s}$ , the other the color  $\mathbf{c}$  and density  $\sigma$ . Each branch consists of two residual blocks, each with 512 hidden units. Read about the network architecture in the Supplementary.

**Predicting Semantics.** Rendering the semantic predictions,  $\mathbf{s}$ , from a given viewpoint gives the semantic segmentation of the scene from that view. Following NeRF [45], we accumulate predictions on rays,  $\mathbf{r}(t) = \mathbf{o} + t \cdot \mathbf{d}$ , originating at the camera center  $\mathbf{o}$  with direction  $\mathbf{d}$ ,

$$\hat{S}(\mathbf{r}) = \int_{t_n}^{t_f} T(t) \sigma(t) \mathbf{s}(t) dt \quad (2)$$

where  $T(t) = \exp(-\int_{t_n}^t \sigma(s) ds)$  is the accumulated transmittance along the ray, and  $t_n$  and  $t_f$  are near and far sampling bounds, which are hyperparameters.

The values of the sampling bounds  $(t_n, t_f)$  are crucial for good performance. In the original NeRF [45] method,  $(t_n, t_f)$  are manually set to tightly bound the scene. PixelNeRF [71] uses manually selected parameters for each object category, i.e. different values for chairs, different values for cars, etc. In Semantic-NeRF [71], the values are selected for the Replica rooms, which vary little in size. In the datasets we experiment on in Sec. 4, scene scale varies drastically from human living spaces with regular depth extents (e.g. living rooms) to industrial facilities (e.g. warehouses), lofts or churches with large far fields. With the goal of scene generalization in mind, we set  $(t_n, t_f)$  globally, regardless of the true near/far bounds of each scene we encounter. This more realistic setting makes the problem harder: our model needs to predict the right density values for a large range of depth fields, reasoning about occupancy within each scene but also about the depth extent of the scene as a whole.

Replacing the semantic predictions  $\mathbf{s}$  with the RGB predictions  $\mathbf{c}$  in Eq. 2 produces the RGB view  $\hat{C}(\mathbf{r})$  from the target viewpoint, as in [45]. While photometric reconstruction is not our goal, we use  $\hat{C}$  during learning and show that it helps capture the scene’s geometry more accurately.

**Predicting Depth.** In addition to the semantic segmentation  $\hat{S}$  and the RGB reconstruction  $\hat{C}$  from a target viewpoint, we also predict the pixel-wise depth of the scene from that view, as in [16] by computing

$$\hat{D}(\mathbf{r}) = \int_{t_n}^{t_f} T(t) \sigma(t) t dt. \quad (3)$$

We use the depth output  $\hat{D}$  only during evaluation. By comparing single-view depth ( $\hat{D}$ ) and semantic segmentation ( $\hat{S}$ ) from many novel viewpoints, we measure our model’s ability to capture geometry and semantics.

### 3.3. Learning Objective

Our primary objective is to segment the scene from the target view. We jointly train the segmentation module  $b^{\text{seg}}$  and implicit 3D function  $f$  to directly solve this task. We also find that auxiliary photometric and source-view losses are crucial for performance. Our objectives require RGB images and 2D semantics from various views in the scene as well as poses to perform re-projection.

Since our goal is to predict a semantic segmentation in the target view, our primary objective is a per-pixel cross-entropy loss between the true class labels  $S$  and predicted class distribution  $\hat{S}$ ,

$$L_{\text{target}}^S = - \sum_{\mathbf{r} \in \mathbf{R}} \sum_{j=1}^{|\mathcal{C}|} S^j(\mathbf{r}) \log \hat{S}^j(\mathbf{r}) \quad (4)$$

where  $\mathcal{C}$  is the set of semantic classes and  $\mathbf{R}$  is the set of rays in the target view. Here,  $S^j(\mathbf{r})$  is the  $\{0, 1\}$  true label for the  $j$ -th class at the intersection of ray  $\mathbf{r}$  and the image.

In addition to this, we minimize auxiliary losses that improve performance on our primary task. The first is a photometric loss on RGB images, namely the squared  $L_2$  distance between the prediction  $\hat{C}$  and actual image  $C$ , or

$$L_{\text{target}}^P = \sum_{\mathbf{r} \in \mathbf{R}} \left\| \hat{C}(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \quad (5)$$

where  $C(\mathbf{r})$  is the true RGB color at the intersection of ray  $\mathbf{r}$  and the image.

Finally, in addition to standard losses on the target view [45, 70, 71], we find it is important to apply losses on the source views. Specifically, we create  $L_{\text{source}}^S$  and  $L_{\text{source}}^P$  that are the semantic and photometric losses, respectively, applied to rays from the source views. These losses help enforce consistency with the input views. Our final objective is given by

$$L = L_{\text{target}}^P + L_{\text{source}}^P + \lambda \cdot (L_{\text{target}}^S + L_{\text{source}}^S) \quad (6)$$

where  $\lambda$  scales the semantic and photometric losses.

## 4. Experiments

We experiment on Hypersim [52] and Replica [62]. Both provide posed views of complex scenes with over 30 object types and under varying conditions of occlusion and lighting. At test time, we evaluate on novel scenes not seen during training. Due to its large size, we treat Hypersim as our main dataset where we run an extensive quantitative analysis. We then show generalization to the smaller Replica.

**Metrics.** We report novel view metrics for semantics and geometry. For a novel view of a test scene, we project the semantic predictions (Eq. 2) and depth (Eq. 3) and compare to the ground truth semantic and depth maps, respectively. Ideally, we would also evaluate directly in 3D, which requires access to full 3D ground truth. However, 3D ground truth is not publicly available for Hypersim and is generally hard to collect. Thus, we treat novel view metrics as proxy metrics for 3D semantic segmentation and depth estimation.

For semantic comparisons, we report semantic segmentation metrics [4] implemented in Detectron2 [66]: **mIoU** is the intersection-over-union (IoU) averaged across classes, **IoU<sup>T</sup>** and **IoU<sup>S</sup>** report IoU by merging all things (object) and stuff classes (wall, floor, ceiling), respectively. **fwIoU** is the per class IoU weighted by the pixel-level frequency of each class, **pACC** is the percentage of correctly labelled pixels and **mACC** is the pixel accuracy averaged across classes. For all, performance is in % and higher is better.

For depth comparisons, we report depth metrics following [18]: **L<sub>1</sub>** is the per-pixel average L<sub>1</sub> distance between ground truth and predicted depth, **Rel** is the L<sub>1</sub> distance normalized by the true depth value, **Rel<sup>T</sup>** and **Rel<sup>S</sup>** is the Rel metric for all things and stuff, respectively.  $\delta < \tau$  is the percentage of pixels with predicted depth within  $[\frac{1}{\tau}, \tau] \times$  the true depth. L<sub>1</sub> is in meters and  $\delta < \tau$  is in %. For  $\delta < \tau$  metrics, higher is better. For all other, lower is better ( $\downarrow$ ).

### 4.1. Experiments on Hypersim

Hypersim [52] is a dataset of 461 complex scenes. Camera trajectories across scenes result in 77,400 images with camera poses, masks for 40 semantic classes [58], along with true depth maps. Hypersim contains on average 50 objects per image, making it a very challenging dataset.

**Dataset.** For each scene, we create source-target pairs from the available views. Each image is labelled as target and is paired with an image from a different viewpoint if: (1) the view frustums intersect by no less than 10%; (2) the camera translation is greater than 0.5m; and (3) the camera rotation is at least 30°. This ensures that source and target views are from different camera viewpoints and broadly depict the same parts of the scene but without large overlap. We follow the original Hypersim split, which splits train/val/test to 365/46/50 disjoint scenes, respectively. Overall, there are 120k/14k/14k pairs in train/val/test, respectively.

**Training details.** We implement ViewSeg in PyTorch with Detectron2 [66] and PyTorch3D [50]. We train on the Hypersim training set for 13 epochs with a batch size of 32 across 32 Tesla V100 GPUs. The input and render resolution are set to 1024×768, maintaining the size of the original dataset. We optimize with Adam [31] and a learning rate of 5e-4. We follow the PixelNeRF [70] strategy for ray sampling: We sample 64 points per ray in the coarse pass and 128 points per ray in the fine pass. In addition to the target view, we randomly sample rays on each source view and additionally minimize the source view loss, as we describe in Sec. 3.3. We set  $t_n = 0.1\text{m}$ ,  $t_f = 20\text{m}$  in Eq. 2 & 3 and  $\lambda = 0.04$  in Eq. 6. More details in the Supplementary.

**Baselines.** In addition to extensive ablations that reveal which components of our method are most important, we compare with multiple baselines and oracle methods to provide context for our method. Our baselines aim to test alternate strategies, including inferring the true RGB image and then predicting the pixel classes as well as lifting a predicted semantic segmentation map to 3D and re-projecting.

To provide context, we report a **Target View Oracle** that has access to the true target view’s image. The target RGB image fundamentally resolves many ambiguities in 3D about what is where, and is not available to our method. Instead, our method is tasked with predicting segmentations and depth from new viewpoints *without the target RGB images*. Our oracle applies appropriate supervised models directly on the true target RGB. For semantic segmentation, we use a model [8] that is identical to ours pre-trained on ADE20k [72] and finetuned on all images of the Hypersim training set. For depth, we use the model from [69], which predicts normalized depth. We obtain metric depth by aligning with the optimal shift and scale following [49, 69].

Our first baseline, denoted **PixelNeRF++**, tests the importance of the end-to-end nature of our approach by performing a two stage process: we use the novel view synthesis (NVS) method of PixelNeRF [70] to infer the RGB of the target view and then apply an image-based model for semantic segmentation. To ensure a fair comparison, PixelNeRF is trained on the Hypersim training set. We use the segmentation model we trained for the oracle to predict semantic segmentations. Depth is predicted with Eq. 3.

Our second baseline, named **CloudSeg**, tests the importance of an implicit representation by comparing with an explicit 3D point cloud representation. Inspired by SynSin [65], we train a semantic segmentation backbone similar to our ViewSeg, along with a depth model, from [69], to lift each source view to a 3D point cloud with per point class probabilities. A differentiable point cloud renderer [50] projects the point clouds from the source images to the target view to produce a semantic and a depth map. CloudSeg is trained on the Hypersim training set and uses the same 2D supervision as our ViewSeg.



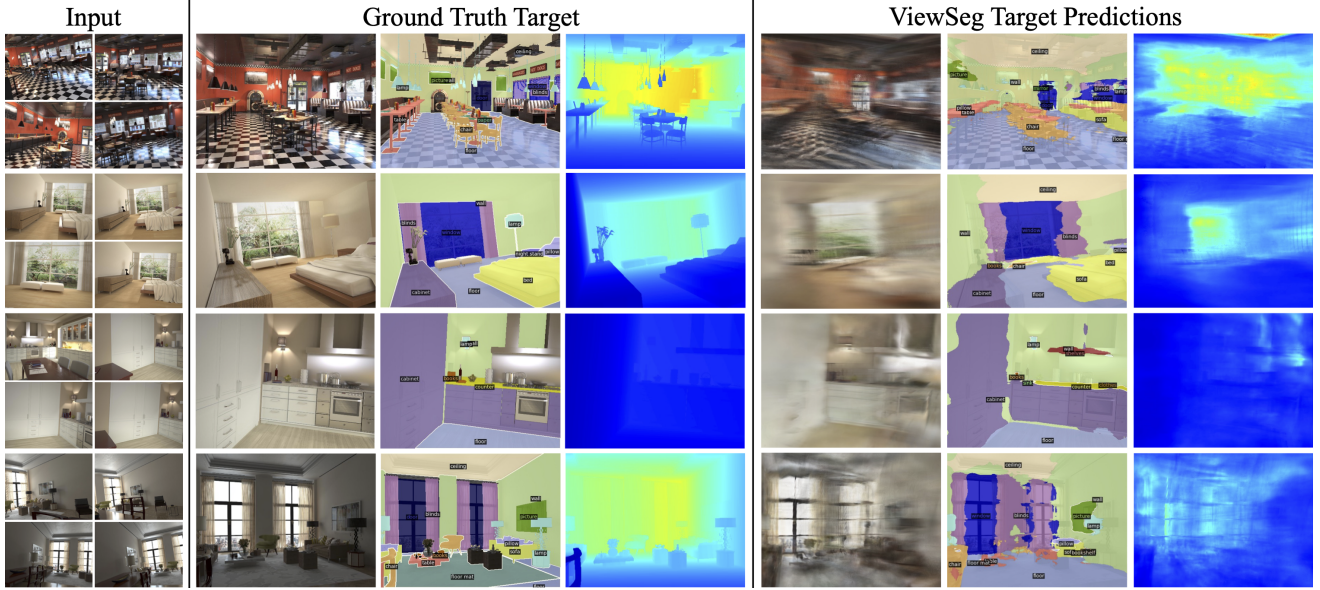


Figure 3. Predictions on Hypersim. For each example, we show the 4 input RGB views (left), the ground truth RGB, semantic and depth maps for the novel target view (middle) and ViewSeg’s predictions (right). RGB synthesis is not our goal, but we show the predicted RGB for completeness. Our model does not have access to the true observations from the target view at test time. Depth: 0.1m 20m.

Model	mIoU	IoU <sup>T</sup>	IoU <sup>S</sup>	fwIoU	pACC	mACC	L <sub>1</sub> (↓)	Rel (↓)	Rel <sup>T</sup> (↓)	Rel <sup>S</sup> (↓)	$\delta < 1.25$	$\delta < 1.25^2$
PixelNeRF++	1.58	14.9	47.9	17.9	3.63	35.9	2.80	0.746	0.856	0.653	0.300	0.531
CloudSeg	0.46	29.6	4.42	1.80	3.31	3.25	3.81	0.856	0.997	0.737	0.145	0.277
ViewSeg	<b>17.1</b>	<b>33.2</b>	<b>58.9</b>	<b>44.8</b>	<b>23.9</b>	<b>62.2</b>	<b>2.29</b>	<b>0.646</b>	<b>0.721</b>	<b>0.584</b>	<b>0.409</b>	<b>0.656</b>
Oracle	40.0	58.1	71.3	66.6	52.1	79.1	0.96	0.235	0.317	0.163	0.731	0.898

Table 1. Comparisons on Hypersim val. We report the performance of both semantic segmentation (blue) and depth estimation (green) for our method, ViewSeg, an oracle which applies supervised single-image semantic segmentation and depth estimation models on the true target RGB views, and two competing approaches, PixelNeRF++ and an explicit 3D point cloud model, CloudSeg, inspired by SynSin [65].

**Results.** Table 1 compares our ViewSeg, PixelNeRF++ and CloudSeg with 4 source views and the oracle on Hypersim val. We observe that PixelNeRF++, which predicts the target RGB view and then applies an image-based model for semantic prediction, performs worse than ViewSeg. This is explained by the low quality RGB predictions, shown in Fig. 4. Predicting high fidelity RGB of novel complex scenes from only 4 input views is still difficult for NVS models, suggesting that a two-stage solution for semantic segmentation will not perform competitively. Indeed, ViewSeg significantly outperforms PixelNeRF++ showing the importance of learning semantics end-to-end. In addition to semantics, ViewSeg outperforms PixelNeRF++ for depth. This suggests that learning semantics jointly has a positive impact on geometry as well. Finally, CloudSeg has a hard time predicting semantics and geometry. This is likely attributed to the wide baseline source and target views in our task which cause explicit 3D representations to produce holes and erroneous predictions in the rendered target output. In SynSin [65], the camera transform between source and target in the datasets the authors explore is significantly narrower, unlike our task where novel viewpoints correspond to wider camera transforms, as shown in Fig. 3.

Fig. 3 shows ViewSeg’s predictions on Hypersim val. We show the 4 source views (left), the ground truth target RGB, semantic and depth map (middle) and ViewSeg’s predictions from the target viewpoint (right). Note that ViewSeg does not have access to ground truth target observations and only receives the 4 images along with camera coordinates for the source and the target viewpoints. Examples in Fig. 3 are of diverse scenes (restaurant, bedroom, kitchen, living room) with many objects (chair, table, counter, cabinet, window, blinds, lamp, picture, floor mat, etc.). We observe that the predicted RGB is of poor quality proving that NVS has a hard time in complex scenes and with few views. RGB synthesis is not our goal. We aim to predict the scene semantics and Fig. 3 shows that our model achieves this. ViewSeg detects stuff (floor, wall, ceiling) well and predicts object segments for the right object types, even for diverse target views. Our depth predictions show that ViewSeg captures the scene’s geometry, even though it was not trained with any 3D supervision.

Fig. 4 compares ViewSeg to PixelNeRF++. Semantic segmentation from predicted RGB results in bad predictions, as shown in the PixelNeRF++ column. Fig. 5 shows examples of semantic 3D reconstructions.

ViewSeg loss	mIoU	IoU <sup>T</sup>	IoU <sup>S</sup>	fwIoU	pACC	mACC	L <sub>1</sub> (↓)	Rel (↓)	Rel <sup>T</sup> (↓)	Rel <sup>S</sup> (↓)	$\delta < 1.25$	$\delta < 1.25^2$
w/o photometric loss	16.9	30.8	58.7	44.8	22.7	62.5	2.49	0.677	0.750	0.615	0.359	0.611
w/o semantic loss	-	-	-	-	-	-	2.58	0.787	0.919	0.678	0.345	0.587
w/o source view loss	14.3	28.2	57.9	28.2	19.3	61.1	2.37	0.683	0.764	0.615	0.397	0.649
w/o viewing dir	16.0	33.1	<b>59.2</b>	<b>44.9</b>	21.5	62.1	2.53	0.708	0.783	0.646	0.354	0.602
final	<b>17.1</b>	<b>33.2</b>	58.9	44.8	<b>23.9</b>	<b>62.2</b>	<b>2.29</b>	<b>0.646</b>	<b>0.721</b>	<b>0.584</b>	<b>0.409</b>	<b>0.656</b>

Table 2. Ablating loss components. We report semantic (blue) and depth (green) performance on Hypersim val without the photometric, the semantic and the source view loss and when excluding the viewing direction from the input. Our model is reported in the last row.

ViewSeg backbone	mIoU	IoU <sup>T</sup>	IoU <sup>S</sup>	fwIoU	pACC	mACC	L <sub>1</sub> (↓)	Rel (↓)	Rel <sup>T</sup> (↓)	Rel <sup>S</sup> (↓)	$\delta < 1.25$
DLv3+ [8] + ADE20k [72]	<b>17.1</b>	<b>33.2</b>	58.9	44.8	<b>23.9</b>	62.2	2.29	0.645	0.721	0.584	0.409
DLv3+ [8] + IN [15]	16.3	<b>33.2</b>	<b>59.2</b>	<b>45.2</b>	22.0	<b>62.5</b>	<b>2.28</b>	<b>0.614</b>	<b>0.682</b>	<b>0.559</b>	<b>0.415</b>
ResNet34 [22] + IN [15]	7.45	21.7	55.9	37.1	11.2	56.1	2.67	0.712	0.815	0.626	0.320

Table 3. Performance on Hypersim val with different semantic segmentation backbones. We show the performance of ViewSeg with DeepLabv3+ (DLv3+) [8] pretrained on ADE20k [72] and on ImageNet [15] and a ResNet34 [22] backbone pretrained on ImageNet [15]. The latter is used in PixelNeRF [70]. DLv3+ improves performance significantly, while ADE20k helps ever so slightly.

ViewSeg	mIoU	pACC	L <sub>1</sub> (↓)	Rel (↓)
w/ 4 views	<b>17.1</b>	<b>23.9</b>	<b>2.29</b>	<b>0.584</b>
w/ 3 views	15.5	20.8	2.39	0.652
w/ 2 views	13.6	18.2	2.57	0.765
w/ 1 view	11.6	15.8	2.62	0.734

Table 4. Input study on Hypersim val for varying number of source views. More views improve semantic and depth performance.

**Ablations and Input Study.** Table 2 ablates various terms in our objective. For reference, the performance of our ViewSeg trained with 4 source views and with the final objective (Eq. 6) is shown in the last row. When we remove the photometric loss,  $L^P$ , semantic performance remains roughly the same but depth performance drops ( $-20\text{cm}$  in  $L_1$ ), which proves that appearance helps capture scene geometry. When we remove the semantic loss,  $L^S$ , and train solely with a photometric loss, we observe a drop in depth ( $-29\text{cm}$  in  $L_1$ ). This suggests that semantics helps geometry; we made a similar observation when comparing to PixelNeRF++ in Table 1. When training without source view losses both semantic and depth performance drop, with semantic performance deteriorating the most ( $-2.8\%$  in mIoU). This confirms our insight that enforcing consistency with the source views improves learning. Finally, when we remove the viewing direction from the model’s input, depth performance suffers the most ( $-24\text{cm}$  in  $L_1$ ).

Table 3 compares different backbones for the 2D segmentation module. We compare DeepLabv3+ (DLv3+) [8] pre-trained on ImageNet [15] and ADE20k [72] and ResNet34 [22] pre-trained on ImageNet [15]. The latter is used in PixelNeRF [70]. DLv3+ significantly boosts performance for both semantics and depth while pre-training on ADE20k slightly adds to the final performance.

Table 4 compares ViewSeg with varying number of source views. We observe that more views improve both semantic segmentation and depth. More than 4 views could lead to further improvements but substantially increase memory and time requirements during training.

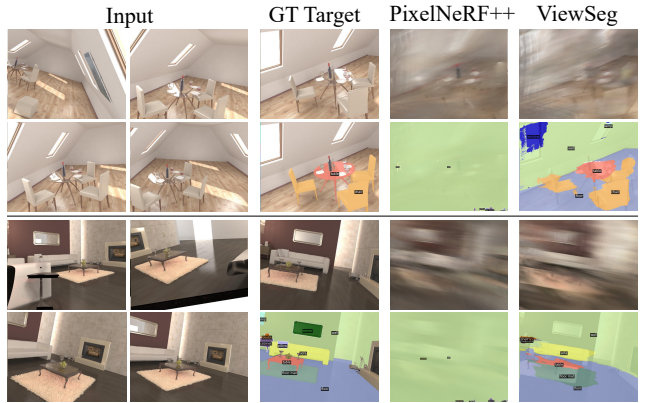


Figure 4. Comparison of ViewSeg and PixelNeRF++ . For each example we show the 4 RGB inputs (1<sup>st</sup>-2<sup>nd</sup> col.), the true RGB and semantic map from the target view (3<sup>rd</sup> col.), the RGB and semantic prediction by PixelNeRF++ (4<sup>th</sup> col.) and the RGB and semantic prediction by our ViewSeg (5<sup>th</sup> col.).

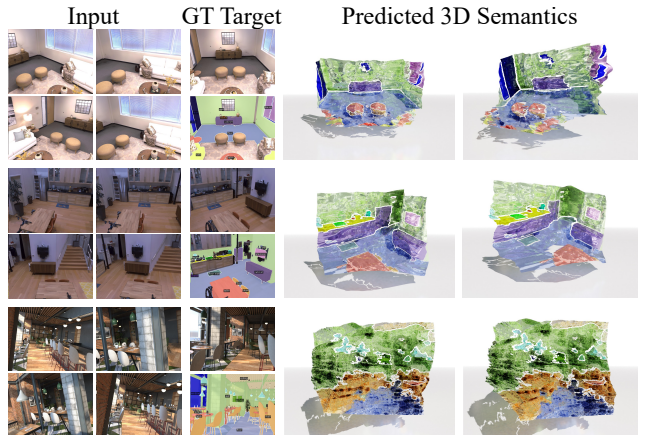


Figure 5. 3D semantic reconstructions on Replica (top two) and Hypersim (bottom). We show the 4 RGB inputs (1<sup>st</sup>-2<sup>nd</sup> col.), the true RGB and semantic map from the novel view (3<sup>rd</sup> col.), and two views of the 3D semantic reconstructions (4<sup>th</sup>-5<sup>th</sup> col.).



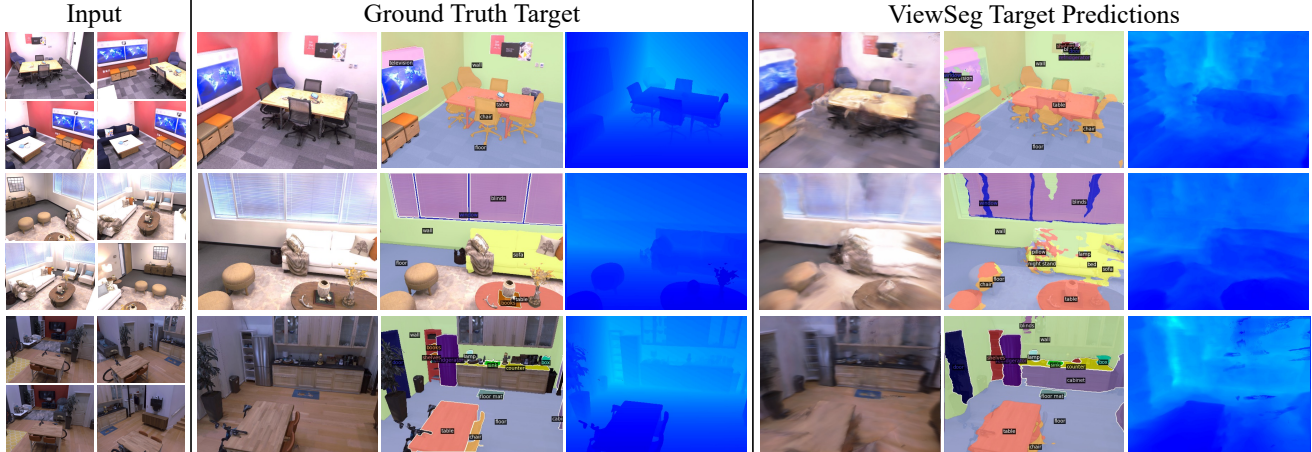



Figure 6. Results on Replica test. We show the 4 input views (left), the ground truth RGB, semantic and depth maps from the novel target viewpoint (middle) and ViewSeg’s predictions (right). Depth colormap: 0.1m  20m.

Model	mIoU	IoU <sup>T</sup>	IoU <sup>S</sup>	fwIoU	pACC	mACC	L <sub>1</sub> (↓)	Rel (↓)	Rel <sup>T</sup> (↓)	Rel <sup>S</sup> (↓)	$\delta < 1.25$	$\delta < 1.25^2$
ViewSeg noft	13.2	44.8	56.0	51.4	27.1	66.8	0.982	0.222	0.194	0.254	0.623	0.880
ViewSeg	30.2	56.2	62.8	62.3	48.4	75.6	0.550	0.130	0.130	0.130	0.851	0.961
Oracle	56.2	76.8	78.0	90.1	79.4	93.8	0.226	0.058	0.065	0.050	0.976	0.998

Table 5. Performance for semantic segmentation (blue) and depth (green) on the Replica [62] test set before finetuning (noft) and after finetuning ViewSeg on Replica’s training set. We additionally report the target view oracle.

## 4.2. Generalization to Replica

We experiment on the Replica dataset [62] which contains real-world scenes such as living rooms and offices. Scenes are complex with many object types and instances in various layouts. We show generalization by applying ViewSeg pre-trained on Hypersim and then further fine-tune it on Replica to better fit to Replica’s statistics.

**Dataset.** We use AI-Habitat [53] to extract multiple views per scene. For each view we collect the RGB image, semantic labels and depth. For each Replica scene, we simulate an agent circling around the center of the 3D scene and render the observations. Note that this is unlike Hypersim [52], where camera trajectories are extracted by the authors a-priori. We use the same camera intrinsics and resolution as Hypersim: the horizontal field of view is 60° and the image resolution is 1024×768. Finally, we map the 88 semantic classes from Replica to NYUv3-13, following [12, 71]. Our dataset consists of 12/3 scenes for train/test, respectively, resulting in 360/90 source-target pairs. Note that this is 330× smaller than Hypersim. Yet, we show compelling results on Replica by pre-training ViewSeg on Hypersim.

**Results.** Table 5 reports the performance of our ViewSeg, trained on Hypersim, before fine-tuning (denoted as noft) and after fine-tuning, as well as a *Target View Oracle*. The oracle fine-tunes the supervised semantic segmentation model on images from our Replica dataset and finds the optimal depth scale and shift for the test set. We observe that ViewSeg’s performance improves significantly when fine-

tuning on Replica for both semantic segmentation and depth across all metrics. It is not surprising that fine-tuning on Replica improves performance as the scenes across the two datasets vary in both object appearance and geometry. We also observe that performance is significantly higher than Hypersim (Table 1). Again, this is not a surprise, as Hypersim contains far more diverse and challenging scenes. However, the trends across the both datasets remain.

Fig. 6 shows predictions on Replica by ViewSeg. We show the 4 RGB inputs (left), the ground truth RGB, semantic and depth map for the novel target viewpoint (middle) and ViewSeg’s predictions (right). Fig. 5 shows 3D semantic reconstructions on two Replica test scenes.

## 5. Conclusion

We present ViewSeg, an end-to-end approach which learns a 3D representation of a scene from merely 4 input views. ViewSeg enables models to query its learnt 3D representation with a previously unseen target viewpoint of a novel scene to predict semantics and depth from that view, without access to any visual information (e.g. RGB) from the view. We discuss our work’s limitations with an extensive quantitative and qualitative analysis. We believe that we present a very promising direction to learn 3D from 2D data with lots of potential for exciting future work.

Regarding ethical risks, we do not foresee any immediate dangers. The datasets used in this work do not contain any humans or any other sensitive information.



**Acknowledgments** We thank Shuaifeng Zhi for his help of Semantic-NeRF, Oleksandr Maksymets and Yili Zhao for their help of AI-Habitat, and Ang Cao, Chris Rockwell, Linyi Jin, Nilesh Kulkarni for helpful discussions.

## References

- [1] Pablo Arbeláez, Jordi Pont-Tuset, Jonathan T Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping. In *CVPR*, 2014. 2
- [2] Sid Yingze Bao, Manmohan Chandraker, Yuanqing Lin, and Silvio Savarese. Dense object reconstruction with semantic priors. In *CVPR*, 2013. 2
- [3] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *SIGGRAPH*, 1999. 2
- [4] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Coco-stuff: Thing and stuff classes in context. In *CVPR*, 2018. 5
- [5] Joao Carreira, Rui Caseiro, Jorge Batista, and Cristian Sminchisescu. Semantic segmentation with second-order pooling. In *ECCV*, 2012. 2
- [6] Joao Carreira and Cristian Sminchisescu. Cpmc: Automatic object segmentation using constrained parametric min-cuts. *TPAMI*, 2011. 2
- [7] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 2, 3
- [8] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 2, 3, 5, 7, 12, 15
- [9] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-image depth perception in the wild. In *NeurIPS*, 2016. 2
- [10] Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. In *NeurIPS*, 2019. 2
- [11] Weifeng Chen, Shengyi Qian, and Jia Deng. Learning single-image depth from videos using quality assessment networks. In *CVPR*, 2019. 2
- [12] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 8
- [13] Jifeng Dai, Kaiming He, and Jian Sun. Convolutional feature masking for joint object and stuff segmentation. In *CVPR*, 2015. 2
- [14] Amaury Dame, Victor A. Prisacariu, Carl Y. Ren, and Ian Reid. Dense reconstruction using 3d object shape priors. In *CVPR*, 2013. 2
- [15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 7, 12, 15
- [16] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. *arXiv preprint arXiv:2107.02791*, 2021. 4
- [17] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015. 2
- [18] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2014. 5
- [19] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *CVPR*, 2019. 2
- [20] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. In *ICCV*, 2019. 2
- [21] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004. 2
- [22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 4, 7, 15
- [23] Yang He, Wei-Chen Chiu, Margret Keuper, and Mario Fritz. Std2p: Rgb-d semantic segmentation using spatio-temporal data-driven pooling. In *CVPR*, 2017. 2
- [24] Philipp Henzler, Jeremy Reizenstein, Patrick Labatut, Roman Shapovalov, Tobias Ritschel, Andrea Vedaldi, and David Novotny. Unsupervised learning of 3d object categories from videos in the wild. In *CVPR*, 2021. 2
- [25] Christian Häne, Nikolay Savinov, and Marc Pollefeys. Class specific 3d object shape priors using surface normals. In *CVPR*, 2014. 2
- [26] Linyi Jin, Shengyi Qian, Andrew Owens, and David F Fouhey. Planar surface reconstruction from sparse views. In *ICCV*, 2021. 2
- [27] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *ECCV*, 2018. 2
- [28] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *Advances in neural information processing systems*, pages 365–376, 2017. 2
- [29] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3D mesh renderer. In *CVPR*, 2018. 2
- [30] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *ICCV*, 2017. 2
- [31] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2014. 5
- [32] Scott Konishi and Alan L Yuille. Statistical cues for domain specific image segmentation with performance analysis. In *CVPR*, 2000. 2
- [33] Nilesh Kulkarni, Abhinav Gupta, David Fouhey, and Shubham Tulsiani. Articulation-aware canonical surface mapping. In *CVPR*, 2020. 2
- [34] Abhijit Kundu, Yin Li, and James M. Rehg. 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In *CVPR*, 2018. 2
- [35] Abhijit Kundu, Vibhav Vineet, and Vladlen Koltun. Feature space optimization for semantic video segmentation. In *CVPR*, 2016. 2

- [36] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *TOG*, 2018. 2
- [37] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *CVPR*, 2018. 2
- [38] Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction. In *ICCV*, 2019. 2
- [39] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751*, 2019. 2
- [40] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 2
- [41] Matthew M Loper and Michael J Black. OpenDR: An approximate differentiable renderer. In *ECCV*, 2014. 2
- [42] Pauline Luc, Natalia Neverova, Camille Couprie, Jakob Verbeek, and Yann LeCun. Predicting deeper into the future of semantic segmentation. In *ICCV*, 2017. 2
- [43] Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. Consistent video depth estimation. *ACM Transactions on Graphics (TOG)*, 39(4):71–1, 2020. 2
- [44] John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In *ICRA*, 2017. 2
- [45] 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. In *ECCV*, 2020. 1, 2, 3, 4, 12
- [46] Yinyu Nie, Xiaoguang Han, Shihui Guo, Yujian Zheng, Jian Chang, and Jian Jun Zhang. Total3dunderstanding: Joint layout, object pose and mesh reconstruction for indoor scenes from a single image. In *CVPR*, 2020. 2
- [47] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: a retargetable forward and inverse renderer. *TOG*, 2019. 2
- [48] Shengyi Qian, Linyi Jin, and David F Fouhey. Associative3d: Volumetric reconstruction from sparse views. In *ECCV*, 2020. 2
- [49] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020. 2, 5
- [50] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 2, 5, 12
- [51] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *International Conference on Computer Vision*, 2021. 2
- [52] Mike Roberts and Nathan Paczan. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. *arXiv preprint arXiv:2011.02523*, 2020. 2, 5, 8, 12
- [53] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *ICCV*, 2019. 8
- [54] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 2002. 2
- [55] Tanner Schmidt, Richard Newcombe, and Dieter Fox. Self-supervised visual descriptor learning for dense correspondence. In *IEEE Robotics and Automation Letters*, 2017. 2
- [56] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *TPAMI*, 2000. 2
- [57] Daeyun Shin, Zhile Ren, Erik B Sudderth, and Charles C Fowlkes. 3d scene reconstruction with multi-layer depth and epipolar transformers. In *ICCV*, 2019. 2
- [58] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from RGBD images. In *ECCV*, 2012. 5
- [59] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings. In *CVPR*, 2019. 2
- [60] 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 *CVPR*, 2019. 2
- [61] Pratul P Srinivasan, Tongzhou Wang, Ashwin Sreelal, Ravi Ramamoorthi, and Ren Ng. Learning to synthesize a 4d rgbd light field from a single image. In *ICCV*, 2017. 2
- [62] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 2, 5, 8, 12, 15
- [63] Shubham Tulsiani, Saurabh Gupta, David Fouhey, Alexei A. Efros, and Jitendra Malik. Factoring shape, pose, and layout from the 2D image of a 3D scene. *CVPR*, 2018. 2
- [64] Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3d scene inference via view synthesis. In *ECCV*, 2018. 2
- [65] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. SynSin: End-to-end view synthesis from a single image. In *CVPR*, 2020. 1, 2, 5, 6
- [66] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. <https://github.com/facebookresearch/detectron2>, 2019. 5, 12
- [67] Zexiang Xu, Sai Bi, Kalyan Sunkavalli, Sunil Hadap, Hao Su, and Ravi Ramamoorthi. Deep view synthesis from sparse photometric images. *TOG*, 2019. 2
- [68] Yufei Ye, Shubham Tulsiani, and Abhinav Gupta. Shelf-supervised mesh prediction in the wild. In *CVPR*, 2021. 2
- [69] Wei Yin, Jianming Zhang, Oliver Wang, Simon Niklaus, Long Mai, Simon Chen, and Chunhua Shen. Learning to recover 3d scene shape from a single image. In *CVPR*, 2021. 2, 5
- [70] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *CVPR*, 2021. 1, 2, 4, 5, 7, 12

- [71] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew Davison. In-place scene labelling and understanding with implicit scene representation. In *ICCV*, 2021. [2](#), [4](#), [8](#)
- [72] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *IJCV*, 2019. [3](#), [5](#), [7](#), [12](#), [15](#)
- [73] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017. [2](#)
- [74] 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. [2](#)



## A. Qualitative Results

Figure 7 shows more qualitative results on Hypersim. We draw a few interesting observations. Overall our model is able to segment objects and stuff from novel viewpoints and capture the underlying 3D geometry as seen in the predicted depth maps. In addition, our model is able to generalize to completely unseen parts of a scene. For example, consider the 4<sup>th</sup> example in Figure 7 of a museum hall. Here, the four input images show the left side of the hall while the novel view queries the right side, which is not captured by the inputs. As seen from our RGB prediction, our model struggles to reconstruct that side of the scene in appearance space. It does much better in semantic space, by correctly placing the wall and the ceiling and by extending the floor. The predicted depth also shows that the model can reason about the geometry of the scene as the right wall starts close to the camera and extends backward consistent with the overall structure of the room. Though both depth and semantic predictions are not perfect, they are evidence that our model has learnt to reason about 3D semantics. We believe this is attributed to the scene priors our model has captured after being trained on hundreds of diverse scenes. The 8<sup>th</sup> (last) example in Figure 7 leads to a similar conclusion. Our model correctly predicts the *pillows* in the target viewpoint but additionally predicts a sofa which is sensibly placed relative to the location and extent of the predicted pillows. The sofa prediction, though not in the ground truth, is a reasonable one and is likely driven by the scene priors our model has captured during training; a line of pillows usually exists on a sofa. Finally, the examples in Figure 7 and Figure 3 in the main paper also reveal our work’s limitations. Our model does not make pixel-perfect predictions and often misses parts of objects or misplaces them by a few pixels in the target view. It is likely that more training data would lead to significantly better predictions. Figure 8 shows more qualitative results on Replica.

Last but not least, we provide video animations of our predictions in the supplementary folder. These complement the static visualizations in the pdf submissions and better demonstrate our model’s ability to make predictions from novel viewpoints on novel scenes.

## B. Network Architecture

The detailed architecture of  $f$  is illustrated in Figure 9. It takes as input a positional embedding of the 3D coordinates of  $\mathbf{x}$ ,  $\gamma(\mathbf{x})$ , the viewing direction  $\mathbf{d}$  and the semantic embeddings  $\{\phi_j^{\text{seg}}\}_{j=1}^N$ . As output,  $f$  produces

$$(\mathbf{c}, \sigma, \mathbf{s}) = f(\gamma(\mathbf{x}), \mathbf{d}, \phi_0^{\text{seg}}(\mathbf{x}), \dots, \phi_{N-1}^{\text{seg}}(\mathbf{x})) \quad (7)$$

where  $\mathbf{c} \in \mathbb{R}^3$  is the RGB color,  $\sigma \in \mathbb{R}$  is the occupancy, and  $\mathbf{s} \in \mathbb{R}^{|\mathcal{C}|}$  is the distribution over semantic classes  $\mathcal{C}$ . Hypersim [52] provides annotations for 40 semantic classes.

We discard  $\{\text{otherstructure}, \text{otherfurniture}, \text{otherprop}\}$  and thus  $|\mathcal{C}| = 37$  for both Hypersim [52] and Replica [62].

We largely follow PixelNeRF [70] for the design of our network. We deviate from PixelNeRF and use 128 instead of 512 for the dimension of hidden layers (as in NeRF [45]) for a more compact network. The dimension of the linear layer which inputs  $\phi_j^{\text{seg}}(\mathbf{x})$  is set to 256 to match the dimension of semantic features from DeepLabv3+ [8].

## C. Training Details

**Pretraining the Semantic Segmentation Module.** We first pretrain DeepLabv3+ [8] on ADE20k [72], which has 20,210 images for training and 2,000 images for validation. We implement DeepLabv3+ in PyTorch with Detectron2 [66]. We train on the ADE20k training set for 160k iterations with a batch size of 16 across 8 Tesla V100 GPUs. The model is initialized using ImageNet weights [15]. We optimize with SGD and a learning rate of 1e-2. During training, we crop each input image to 512x512. We remove the final layer, which predicts class probabilities, and use the output from the penultimate layer as our semantic encoder. We do not freeze the model when training ViewSeg, allowing finetuning on Hypersim semantic categories.

**Training Details on Hypersim.** We implement ViewSeg in PyTorch3D [50] and Detectron2 [66]. We initialize the semantic segmentation module with ADE20k pretrained weights. We train on the training set for 13 epochs with a batch size of 32 across 32 Tesla V100 GPUs. The input and render resolution are set to 1024x768. We optimize with Adam and a learning rate of 5e-4. We follow the PixelNeRF [70] strategy for ray sampling: We sample 64 points per ray in the coarse pass and 128 points per ray in the fine pass; we sample 1024 rays per image.

**Training Details on Replica.** We finetune our model on the Replica training set [62]. Replica has 18 scenes. In practice, we find Replica does not have room-level annotations and our sampled source and target views can be at different rooms within the Replica apartments. Hence, we exclude them from our data. We split the rest 15 scenes into a train/val split of 12/3 scenes. We use the same hyperparameters as Hypersim to finetune our model on Replica.

## D. Noisy Camera Experiment

In our experiments, we assume camera poses both during training and evaluation. We perform an additional ablation assuming *noisy* cameras for both during training and testing. During evaluation, source view cameras are noisy but not the target camera, as we wish to compare to the target view ground truth. We insert noise to the cameras by perturbing the rotation matrix with random angles sampled from  $[-10^\circ, 10^\circ]$  in all three axis ( $X$ ,  $Y$  &  $Z$ ). This results

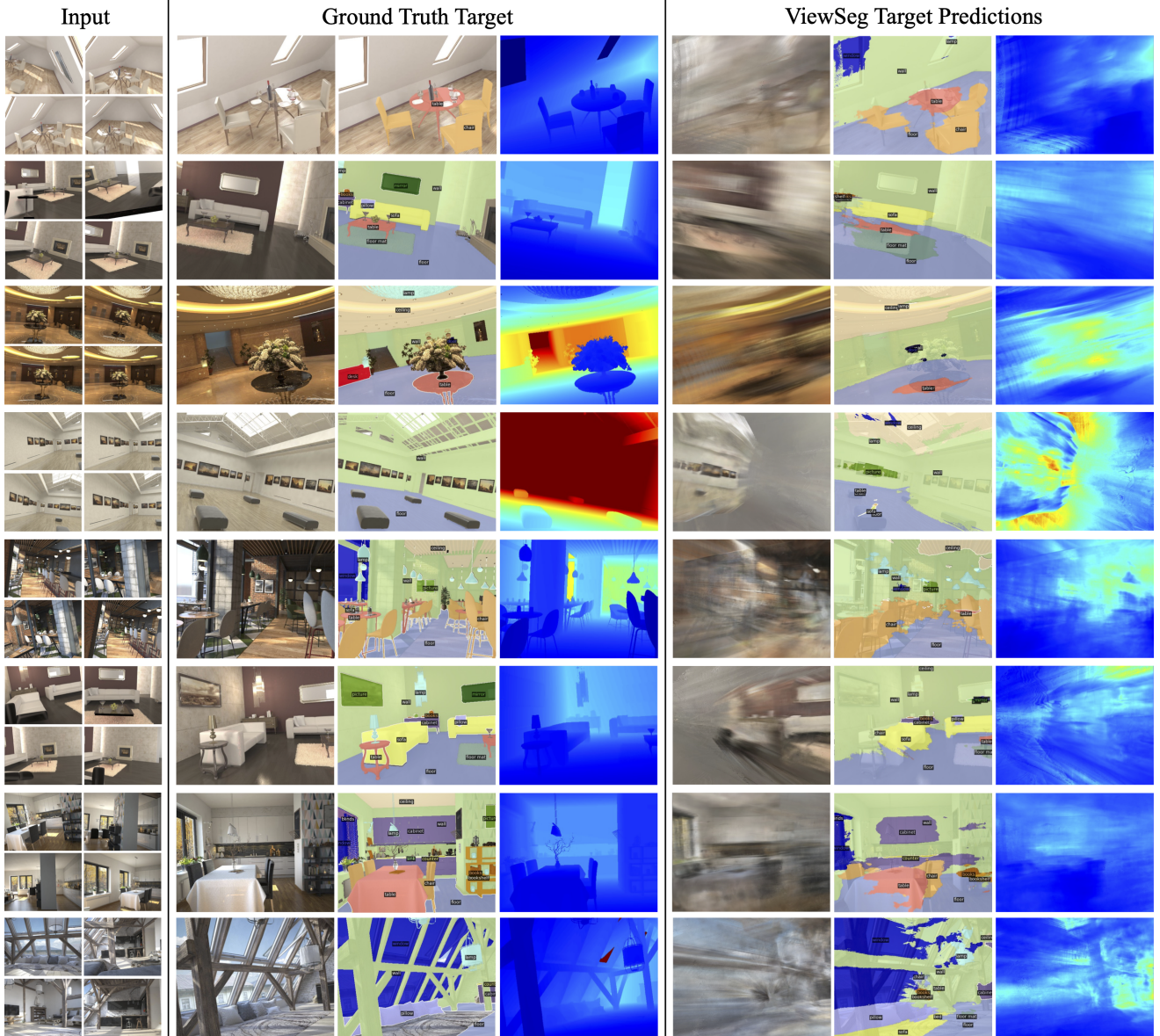


Figure 7. More predictions on Hypersim. For each example, we show the 4 input RGB views (left), the ground truth RGB, semantic and depth maps for the novel target view (middle) and ViewSeg’s predictions (right). Our model does not have access to the true observations from the target view at test time. See our video animations. Depth: 0.1m 20m.

in a significant camera noise and stretch tests our method under such conditions.

Table 6 shows results on the noisy Replica test set. The 1<sup>st</sup> row shows the performance of our ViewSeg pre-trained on Hypersim and without any finetuning. The 2<sup>nd</sup> row shows the performance of our model finetuned on the noisy Replica training set. We observe that performance for both model variants is worse compared to having perfect cameras, as is expected. Performance improves after training with camera noise, suggesting that our ViewSeg is able to generalize better when trained with noise in cameras. Figure 10 shows qualitative results on the noisy Replica test

set. The predicted RGB targets are significantly worse compared to the perfect camera scenario, which suggests that RGB prediction with imperfect cameras is very challenging. However, the semantic predictions are much better computer to their RGB counterparts. This shows that our approach is able to capture scene priors and generalize to new scenes even under imperfect conditions.

## E. Evaluation

We report the complete set of depth metrics for all the tables in the main submission. The comparison with PixelNeRF and CloudSeg is in Table 7. The ablation of different



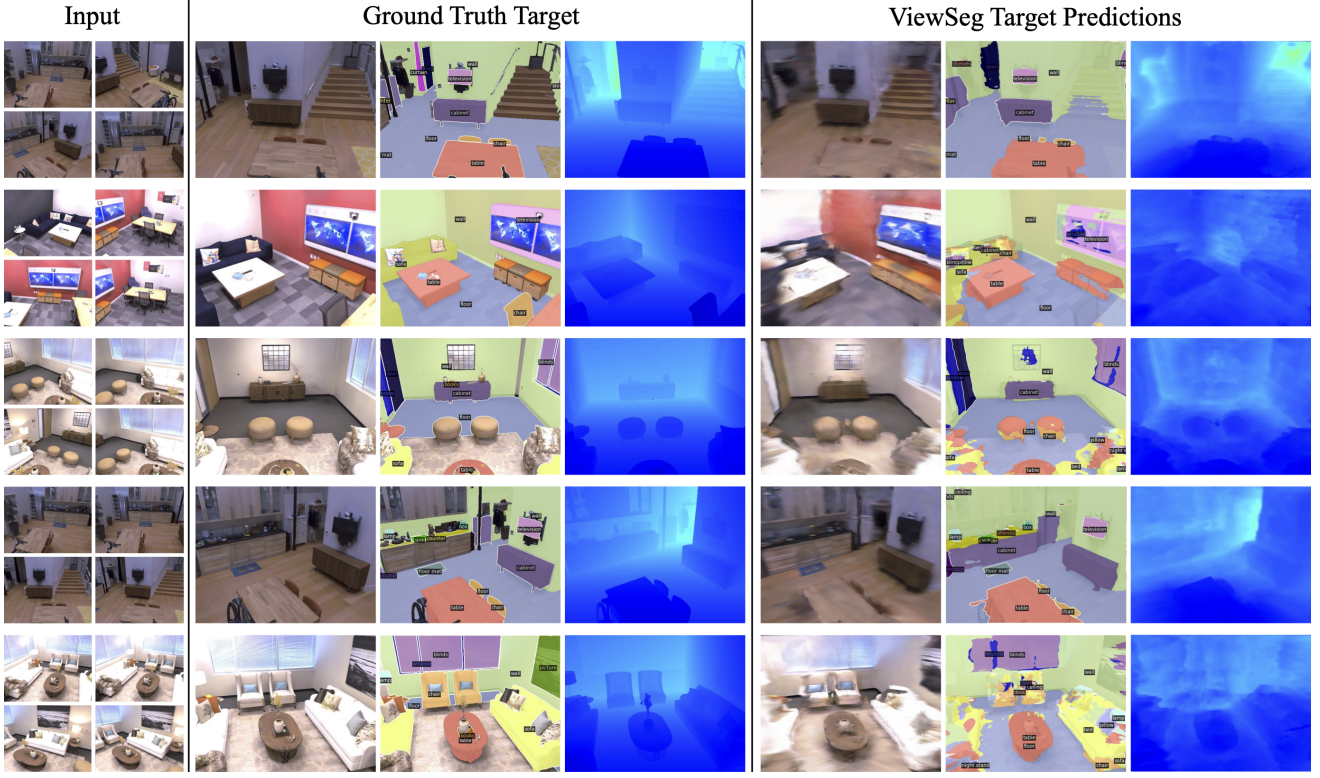



Figure 8. More predictions on Replica. For each example, we show the 4 input RGB views (left), the ground truth RGB, semantic and depth maps for the novel target view (middle) and ViewSeg’s predictions (right). Our model does not have access to the true observations from the target view at test time. See our video animations. Depth: 0.1m  20m.

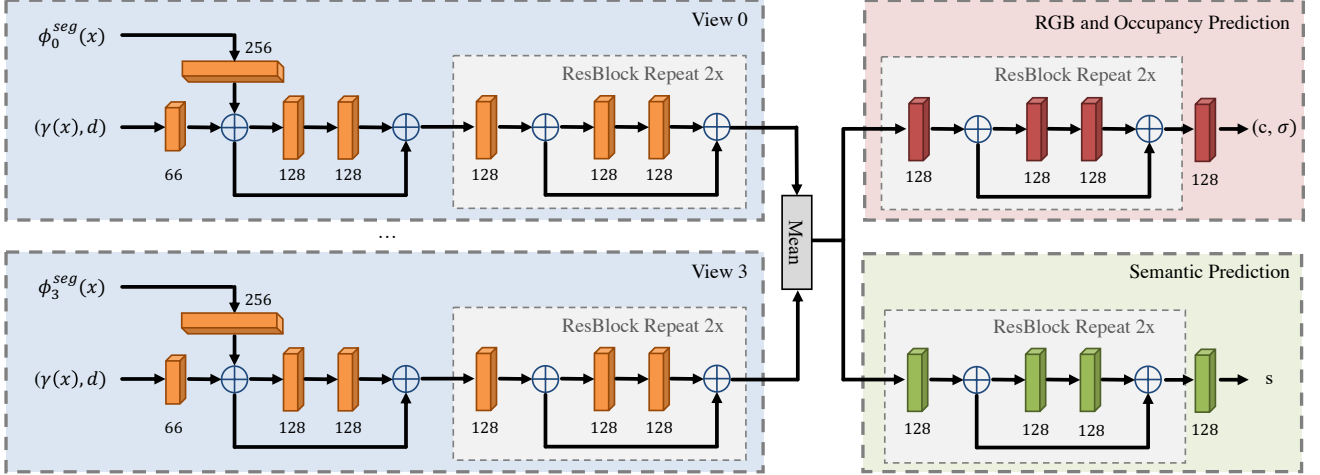


Figure 9. Detailed architecture of our semantic 3D representation  $f$ . Each cuboid is a linear layer where the number around it represents the input dimension. The output dimension is always 128 except the last layer of RGB and semantic prediction. Before the mean aggregation, the network takes inputs from each source view but weights are shared.

loss terms is in Table 8. The comparison of different backbones is in Table 9. The input study is in Table 10. Our experiments on the Replica dataset are in Table 11.



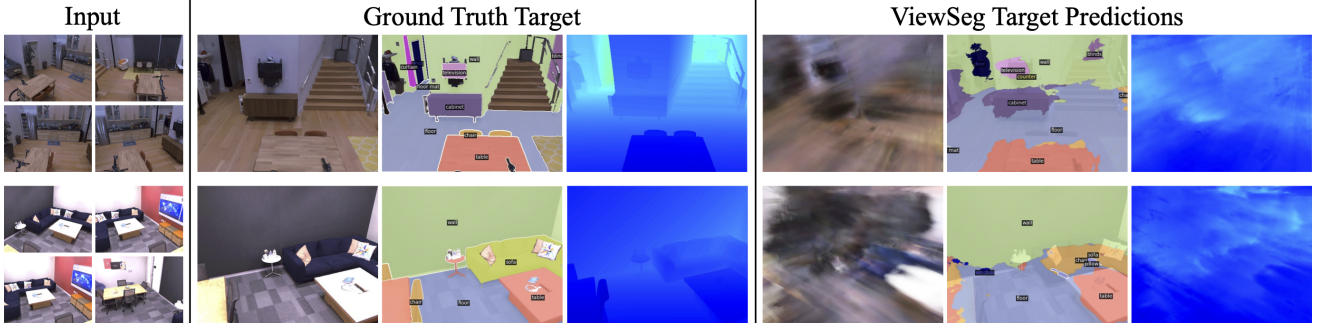



Figure 10. Predictions on Replica with noisy cameras. For each example, we show the 4 input RGB views (left), the ground truth RGB, semantic and depth maps for the novel target view (middle) and ViewSeg’s predictions (right). Our model does not have access to the true observations from the target view at test time. Depth: 0.1m  20m.

Model	mIoU	IoU <sup>T</sup>	IoU <sup>S</sup>	fwIoU	pACC	mACC	L <sub>1</sub> (↓)	Rel (↓)	Rel <sup>T</sup> (↓)	Rel <sup>S</sup> (↓)	$\delta < 1.25$	$\delta < 1.25^2$
ViewSeg noft	8.57	35.2	61.3	40.3	18.3	57.9	1.10	0.253	0.240	0.268	0.579	0.840
ViewSeg	14.1	40.1	62.8	42.8	24.6	60.6	0.887	0.208	0.232	0.180	0.631	0.888

Table 6. Performance for semantic segmentation (blue) and depth (green) on the Replica [62] test set with noisy cameras.

Model	mIoU	IoU <sup>T</sup>	IoU <sup>S</sup>	fwIoU	pACC	mACC	L <sub>1</sub> (↓)	L <sub>1</sub> <sup>T</sup> (↓)	L <sub>1</sub> <sup>S</sup> (↓)	Rel (↓)	Rel <sup>T</sup> (↓)	Rel <sup>S</sup> (↓)	$\delta < 1.25$	$\delta^T < 1.25$	$\delta^S < 1.25$	$\delta < 1.25^2$	$\delta^T < 1.25^2$	$\delta^S < 1.25^2$	$\delta < 1.25^3$	$\delta^T < 1.25^3$	$\delta^S < 1.25^3$
PixelNeRF++	1.58	14.9	47.9	17.9	3.63	35.9	2.80	2.69	2.90	0.746	0.856	0.653	0.300	0.276	0.319	0.531	0.500	0.557	0.689	0.663	0.712
CloudSeg	0.46	29.6	4.42	1.80	3.31	3.25	3.81	3.77	3.83	0.856	0.997	0.737	0.145	0.105	0.178	0.277	0.211	0.332	0.389	0.314	0.451
ViewSeg	<b>17.1</b>	<b>33.2</b>	<b>58.9</b>	<b>44.8</b>	<b>23.9</b>	<b>62.2</b>	<b>2.29</b>	<b>2.18</b>	<b>2.38</b>	<b>0.646</b>	<b>0.721</b>	<b>0.584</b>	<b>0.409</b>	<b>0.393</b>	<b>0.421</b>	<b>0.656</b>	<b>0.633</b>	<b>0.676</b>	<b>0.794</b>	<b>0.772</b>	<b>0.812</b>
Oracle	40.0	58.1	71.3	66.6	52.1	79.1	0.96	1.10	0.83	0.235	0.317	0.163	0.731	0.651	0.800	0.898	0.848	0.942	0.954	0.925	0.978

Table 7. Extended version of Table 1 in the main paper.

ViewSeg loss	mIoU	IoU <sup>T</sup>	IoU <sup>S</sup>	fwIoU	pACC	mACC	L <sub>1</sub> (↓)	L <sub>1</sub> <sup>T</sup> (↓)	L <sub>1</sub> <sup>S</sup> (↓)	Rel (↓)	Rel <sup>T</sup> (↓)	Rel <sup>S</sup> (↓)	$\delta < 1.25$	$\delta^T < 1.25$	$\delta^S < 1.25$	$\delta < 1.25^2$	$\delta^T < 1.25^2$	$\delta^S < 1.25^2$	$\delta < 1.25^3$	$\delta^T < 1.25^3$	$\delta^S < 1.25^3$
w/o photometric loss	16.9	30.8	58.7	44.8	22.7	62.5	2.49	2.35	2.61	0.677	0.750	0.615	0.359	0.347	0.363	0.611	0.594	0.625	0.764	0.744	0.780
w/o semantic loss	-	-	-	-	-	-	2.58	2.52	2.63	0.787	0.919	0.678	0.345	0.317	0.369	0.587	0.548	0.621	0.740	0.704	0.770
w/o source view loss	14.3	28.2	57.9	28.2	19.3	61.1	2.37	2.27	2.45	0.683	0.764	0.615	0.397	0.378	0.413	0.649	0.623	0.670	0.785	0.761	0.806
w/o viewing dir	16.0	33.1	<b>59.2</b>	<b>44.9</b>	21.5	62.1	2.53	2.38	2.65	0.708	0.783	0.646	0.354	0.351	0.356	0.602	0.593	0.610	0.759	0.744	0.772
final	<b>17.1</b>	<b>33.2</b>	58.9	44.8	<b>23.9</b>	<b>62.2</b>	<b>2.29</b>	<b>2.18</b>	<b>2.38</b>	<b>0.646</b>	<b>0.721</b>	<b>0.584</b>	<b>0.409</b>	<b>0.393</b>	<b>0.421</b>	<b>0.656</b>	<b>0.633</b>	<b>0.676</b>	<b>0.794</b>	<b>0.772</b>	<b>0.812</b>

Table 8. Extended version of Table 2 in the main paper.

ViewSeg backbone	mIoU	IoU <sup>T</sup>	IoU <sup>S</sup>	fwIoU	pACC	mACC	L <sub>1</sub> (↓)	L <sub>1</sub> <sup>T</sup> (↓)	L <sub>1</sub> <sup>S</sup> (↓)	Rel (↓)	Rel <sup>T</sup> (↓)	Rel <sup>S</sup> (↓)	$\delta < 1.25$	$\delta^T < 1.25$	$\delta^S < 1.25$	$\delta < 1.25^2$	$\delta^T < 1.25^2$	$\delta^S < 1.25^2$	$\delta < 1.25^3$	$\delta^T < 1.25^3$	$\delta^S < 1.25^3$
DLv3+ [8] + ADE30k [72]	<b>17.1</b>	<b>33.2</b>	58.9	44.8	<b>23.9</b>	62.2	2.29	2.18	2.38	0.646	0.721	0.584	0.409	0.393	0.421	0.656	0.633	0.676	0.794	0.772	0.812
DLv3+ [8] + IN [15]	16.3	<b>33.2</b>	<b>59.2</b>	<b>45.2</b>	22.0	<b>62.5</b>	<b>2.28</b>	<b>2.17</b>	<b>2.36</b>	<b>0.614</b>	<b>0.682</b>	<b>0.559</b>	<b>0.415</b>	<b>0.400</b>	<b>0.427</b>	<b>0.663</b>	<b>0.640</b>	<b>0.682</b>	<b>0.799</b>	<b>0.776</b>	<b>0.818</b>
ResNet34 [22] + IN [15]	7.45	21.7	55.9	37.1	11.2	56.1	2.67	2.51	2.81	0.712	0.815	0.626	0.320	0.304	0.333	0.562	0.541	0.580	0.720	0.702	0.736

Table 9. Extended version of Table 3 in the main paper.

ViewSeg	mIoU	IoU <sup>T</sup>	IoU <sup>S</sup>	fwIoU	pACC	mACC	L <sub>1</sub> (↓)	L <sub>1</sub> <sup>T</sup> (↓)	L <sub>1</sub> <sup>S</sup> (↓)	Rel (↓)	Rel <sup>T</sup> (↓)	Rel <sup>S</sup> (↓)	$\delta < 1.25$	$\delta^T < 1.25$	$\delta^S < 1.25$	$\delta < 1.25^2$	$\delta^T < 1.25^2$	$\delta^S < 1.25^2$	$\delta < 1.25^3$	$\delta^T < 1.25^3$	$\delta^S < 1.25^3$
w/ 4 views	<b>17.1</b>	<b>33.2</b>	<b>58.9</b>	<b>44.8</b>	<b>23.9</b>	<b>62.2</b>	<b>2.29</b>	<b>2.18</b>	<b>2.38</b>	<b>0.646</b>	<b>0.721</b>	<b>0.584</b>	<b>0.408</b>	<b>0.393</b>	<b>0.421</b>	<b>0.656</b>	<b>0.633</b>	<b>0.676</b>	<b>0.794</b>	<b>0.772</b>	<b>0.812</b>
w/ 3 views	15.5	31.3	58.7	43.9	20.8	61.5	2.39	2.25	2.49	0.652	0.730	0.587	0.387	0.376	0.395	0.634	0.617	0.648	0.777	0.759	0.793
w/ 2 views	13.6	27.4	57.7	41.9	18.2	60.2	2.57	2.49	2.64	0.765	0.878	0.672	0.363	0.339	0.383	0.605	0.574	0.633	0.751	0.721	0.776
w/ 1 view	11.6	24.9	56.5	39.7	15.8	57.9	2.62	2.52	2.70	0.734	0.828	0.657	0.332	0.322	0.339	0.562	0.541	0.580	0.710	0.686	0.730

Table 10. Extended version of Table 4 in the main paper.

ViewSeg	mIoU	IoU <sup>T</sup>	IoU <sup>S</sup>	fwIoU	pACC	mACC	L <sub>1</sub> (↓)	L <sub>1</sub> <sup>T</sup> (↓)	L <sub>1</sub> <sup>S</sup> (↓)	Rel (↓)	Rel <sup>T</sup> (↓)	Rel <sup>S</sup> (↓)	$\delta < 1.25$	$\delta^T < 1.25$	$\delta^S < 1.25$	$\delta < 1.25^2$	$\delta^T < 1.25^2$	$\delta^S < 1.25^2$	$\delta < 1.25^3$	$\delta^T < 1.25^3$	$\delta^S < 1.25^3$
ViewSeg noft	13.2	44.8	56.0	51.4	27.1	66.8	0.982	0.851	1.138	0.222	0.194	0.254	0.623	0.687	0.546	0.880	0.905	0.850	0.968	0.974	0.961
ViewSeg	30.2	56.2	62.8	62.3	48.4	75.6	0.550	0.510	0.597	0.130	0.130	0.130	0.851	0.857	0.844	0.961	0.953	0.972	0.986	0.980	0.992
Oracle	56.2	76.8	78.0	90.1	79.4	93.8	0.226	0.230	0.220	0.058	0.065	0.050	0.976	0.965	0.991	0.998	0.996	0.999	1.000	1.000	1.000

Table 11. Extended version of Table 5 in the main paper.