

Supplementary Material for Cross-Dimensional Refined Learning for Real-Time 3D Visual Perception from Monocular Video

In this appendix, we provide further details on the fragment bounding volume (FBV) construction, the cross-dimensional refinement (CDR) prior mechanism, network design, implementation setup, and further experimental results. We also include a 2-minute interactive video demo using CDRNet model, to showcase its capability of conducting real-time 3D perception.

1. Fragment Bounding Volume Construction

Fig. 1 illustrates the construction process of the batched input data in FBV. A back projection process where the camera-to-world coordinate transform happens is utilized to map the 2D features in the Y-up camera coordinate into \mathcal{F}_i in the Z-up world coordinate for 3D perception.

2. Proof of CDR with Prior Knowledge

The derivation for Eq. (3) in the main body is obtained in this section as follows.

Proof. Considering the 3D feature extraction network described in Sec. 3.1 of the main body **without** CDR priors, the temporal-coherent local feature $L_{s,i}$ at stage s can be inferred by a parametric GRU fusion with the input of a concatenation between the raw geometric feature \mathcal{V}_s and the upsampled 3D feature from previous stage $L_{s+1,i}$ as,

$$L_{s,i} = H_{s,i}(\text{Concat}(\mathcal{V}_s, \text{Up}(L_{s+1,i})), H_{s,i-1}) , \quad (1)$$

$$H_{s,i} = G_{s,i}[\sum_{i'=1}^i \mathcal{F}_{s,i'}.coords()] , \quad (2)$$

where the hidden state of $H_{s,i}$ under current fragment \mathbf{F}_i is extracted from the global feature volume $G_{s,i}$ with the masking coming from the coordinates of \mathcal{V}_s as valid. The target quantity prediction is estimated from the previous hidden state $H_{s,i-1}$ as below,

$$\hat{I}_{s,i} = h(L_{s,i}) = h(H_{s,i}(\mathcal{V}_s, H_{s,i-1} | \mathcal{F}_i)) = g_{\hat{\theta}}(\mathbf{F}_i) . \quad (3)$$

where the hidden layer $h(\cdot)$ is constructed with a single layer perceptron for the target quantity prediction head, θ

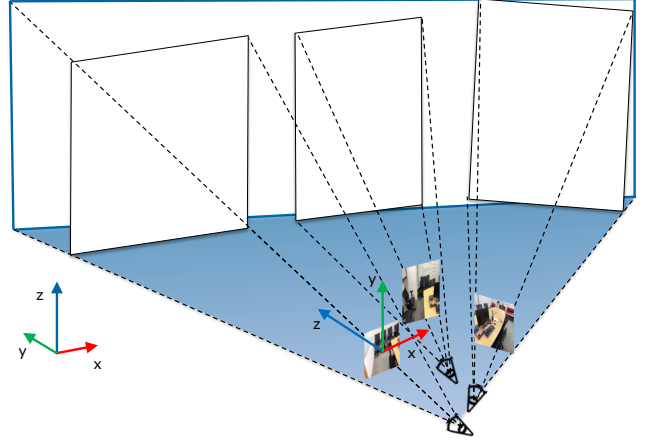


Figure 1. **Illustration of the FBV construction process.** The FBV is constructed with $N_k = 3$ in this example for simplicity. The blue shaded merged frustum is defined as FBV, which is to incorporate the multi-view 3D features for TSDF and semantic volume estimation.

denotes the overall trainable parameters in the network, and naturally $g_{\hat{\theta}}(\cdot)$ denotes the entire neural network mapping from the posed image fragment to the target quantity prediction under the trained $\hat{\theta}$.

Then at the time of training, given N_b batches of \mathbf{F}_i as the input and the voxel data pair values in the sparsified FBV, $I_i \in \mathcal{F}_i = \{T_i^{N_o}, S_i^{N_o}\}$ as the groundtruth of the target quantity estimation for each stage of the coarse-to-fine hierarchy while the number of occupied voxels is defined as,

$$N_o = \text{len}(\mathcal{V}_s^{x \times y \times z}[\hat{o}_i].\text{flatten}()) , \quad (4)$$

θ can be estimated following *maximum likelihood estima-*

tion (MLE) as below:

$$\begin{aligned}
\hat{\theta}_{\text{MLE}} &= \arg \max_{\theta} \prod_{i=1}^{N_b} \prod_{k=1}^{N_o} p(I_{i,k} | \mathbf{F}_i, \theta) \\
&= \arg \max_{\theta} \sum_{i=1}^{N_b} \sum_{k=1}^{N_o} \log p(I_{i,k} | \mathbf{F}_i, \theta) \quad (5) \\
&= \arg \min_{\theta} \sum_{l \in \{T, S\}} \sum_{i=1}^{N_b} \sum_{k=1}^{N_o} \mathcal{L}_l(\hat{I}_{i,k}, I_{i,k}) ,
\end{aligned}$$

where \mathcal{L}_l is the respective loss that is responsible for each type of target quantity. It can be categorized as either a regression or a classification problem and hence derived from the likelihood $p(I_{i,k} | \mathbf{F}_i, \theta)$. Readers are suggested to refer to Sec. 5.5 in [3] for further details about the loss. Without loss of generality, we plotted the $\mathcal{L}_l(\theta)$ - θ characteristic in Fig. 2 by adopting the same visualization methods described in [4, 8]. Fig. 2(a) shows the mean absolute error loss for Truncated Signed Distance Function (TSDF) prediction, where θ is optimized with MLE and finalized as $\hat{\theta}_{\text{MLE}} = \theta_{\text{MLE}}$. Thus, at inference with the trained parameters substituted in Eq. (3),

$$\hat{I}_i = g_{\hat{\theta}_{\text{MLE}}}(\mathbf{F}_i) = g_{\theta_{\text{MLE}}}(\mathbf{F}_i) . \quad (6)$$

Via Bayes' rule, we hope to optimize the training process in a *maximum a posteriori* (MAP) perspective with the CDR priors, namely,

$$\begin{aligned}
\hat{\theta}_{\text{MAP}} &= \arg \max_{\theta} \prod_{i=1}^{N_b} \prod_{k=1}^{N_o} p(I_{i,k} | \mathbf{F}_i, \theta) \times p(\theta) \\
&= \arg \max_{\theta} \sum_{i=1}^{N_b} \sum_{k=1}^{N_o} (\log p(I_{i,k} | \mathbf{F}_i, \theta) + \log p(\theta)) \\
&= \arg \min_{\theta} - \sum_{i=1}^{N_b} \sum_{k=1}^{N_o} (\log p(I_{i,k} | \mathbf{F}_i, \theta) + \log p(\theta)) , \quad (7)
\end{aligned}$$

where $p(\theta)$ denotes the CDR prior probability on network parameters which incorporates 2D information to construct the 3D counterpart. The intuitive mechanism is explained in the main body. To generate the CDR priors, a dedicated hidden-layers module $h_{\text{prior}}(\cdot)$ with 2D prior distribution on the network $p(\theta)$ is included. The target quantity estimation with CDR priors involved can be retrieved as,

$$\begin{aligned}
\hat{I}_{i_{\text{MAP}}} &= g_{\theta_{\text{MAP}}}(\mathbf{F}_i) \\
&= h(h_{\text{prior}}(H_{s,i})) \\
&= \epsilon g_{\theta_{\text{MLE}}}(\mathbf{F}_i) + (1 - \epsilon) g_{\theta_{\text{prior}}}(\mathbf{F}_i) , \quad (8)
\end{aligned}$$

where $g_{\theta_{\text{prior}}}(\cdot)$ is the functional mapping from \mathbf{F}_i to $h_{\text{prior}}(H_{s,i})$ similar to Eq. (3); ϵ denotes the weighting

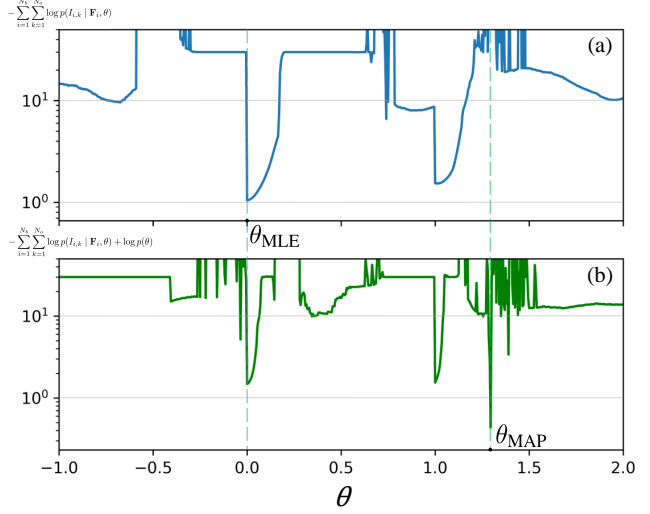


Figure 2. **Loss function against trainable θ on the ScanNet validation dataset.** (a) Negative log-likelihood against θ ; (b) Negative log-posterior against θ . θ_{suffix} denotes the optimal θ in the “suffix” optimization situation. Optimizing with MAP in (b) can generalize more expressive θ as justified in the experiments.

factor that is learnt by the network to generalize the offset relationship between θ_{prior} and θ_{MLE} . Merging the distributions of θ_{MLE} and θ_{prior} , we have the posterior corresponding to $g_{\theta_{\text{MAP}}}(\mathbf{F}_i)$ and the optimization $\hat{\theta}_{\text{MAP}} = \theta_{\text{MAP}}$, which are shown in Fig. 2(b).

At the time of inference when training is done, with the estimated parameter $\hat{\theta}$ in the trained network, the hidden state output from GRU fusion is equivalent to the MLE optimized parameter’s result as in Eq. (3). While the prior-conditioned 3D feature is equivalent to prior-optimized result, namely,

$$g_{\theta_{\text{MLE}}}(\mathbf{F}_i) = h(H_{s,i}(\mathcal{V}_s, H_{s,i-1} | \mathcal{F}_i)) , \quad (9)$$

$$g_{\theta_{\text{prior}}}(\mathbf{F}_i) = X_{\text{prior}} . \quad (10)$$

Thus, by substituting Eq. (9) and Eq. (10) into Eq. (8), Eq. (3) in the main body holds. Similarly, $\hat{\theta}$ that is responsible for semantics predictions can be retrieved by enforcing $\mathcal{L}_l = \mathcal{L}_{\text{CE}}$. \square

Discussion on the Loss Landscape Visualization. It is noteworthy that, for the visualization of the loss landscape for high-dimensional deep neural networks (beyond 12 million parameters in CDRNet as shown in Table 1) in Fig. 2, it is impossible to compare the loss sweeping through all variable θ entries even with dimensionality reduction such as principal component analysis. Therefore, one needs to find a workaround to bypass the θ sweeping to investigate the loss landscape of the neural network. To this end, we adopt the same loss visualization methods as in [4, 8] by

computing:

$$\begin{aligned}\mathcal{L}_l(g_\theta(\mathbf{F}_i), I_i) &= \mathcal{L}_l(\theta) \\ &= \mathcal{L}_l((1 - \alpha) \cdot \theta_{\text{MLE}} + \alpha \cdot \theta_{\text{prior}}),\end{aligned}\quad (11)$$

where $\mathcal{L}_l(\theta)$ is achieved by substituting Eq. (3) into Eq. (5) for varying α who serves as a linear interpolation between two different parameter vectors in parameter space. Such visualization enables one to comprehend how is CDR priors benefit to the learning of the backbone network by optimizing into θ_{MAP} with a lower testing loss in parameter space. Hence, we plot the $\mathcal{L}_l(\theta)$ - θ characteristics under MLE and MAP optimizations, in Fig. 2 respectively. In Fig. 2(a), we remove feature refinement modules in CDRNet to retain the network in an MLE optimization fashion, the network parameters are learned as $\hat{\theta} = \theta_{\text{MLE}}$; whereas in Fig. 2(b), the parameters in CDRNet are learned as $\hat{\theta} = \theta_{\text{MAP}}$ which is lower than θ_{MLE} with the help of the 2D-to-3D prior knowledge $p(\theta)$ during optimization, proving the efficacy in generalizability of the proposed 2D-to-3D refinement method.

3. Network Structure

To give a detailed dissection on our proposed network, following the existing sophisticated network layer block definitions in [5, 9], we first define a layer block as the combination of either a parameterized operation (e.g., a convolutional layer or a fully-connected layer) with a batch normalization layer, or a parameterized operation with a batch normalization layer plus an activation, such as ReLU. In Table 1, we present the number of layer blocks for each learnable single module of our network, and also other module information to showcase the network details in a layer level. It is noteworthy that, the value of K and S here represents the majority value of the layers in the particular module, by omitting the 1×1 convolutions that appears in most of the modules.

From Table 1, it can be seen that the coarse stage dominates in terms of the parameter amount. This one of the reasons why we set the weighting factor of the losses on the coarse stage with the highest weighting factor, as discussed in Sec. 4.

4. Implementation Setup

We train our network in an end-to-end manner with randomly initialized weights except that we load the pretrained model the 2D feature extractor, MNasNet from PyTorch official checkpoints. We adopt trilinear interpolation for refining both displaced depth predictions and anchored features, and nearest-neighbor interpolation for upsampling features to the next stage in the coarse-to-fine hierarchy. At each stage, we use Sigmoid activation to fuse the GRU hidden state and current state to create output coherent feature from

Module	#Layers	K	S	C_{in}	C_{out}	#Parameters
MNasNet-FPN.FineConv2d	12	3	2	3	24	13.3k
MNasNet-FPN.MediumConv2d	9	5	2	24	40	33.1k
MNasNet-FPN.CoarseConv2d	9	5	2	40	80	217.7k
MNasNet-FPN.OutsConv2d	5	3	1	80	C_{set}	57.8k
Semseg2d-Preds	1	FC	-	24	C_{sem}	0.48k
CostVolume-RegConv3d	11	3	2	24	1	296.3k
CoarseStage.SparseConv3d	25	3	1	81	96	4.5M
CoarseStage.GRUConv3d	3	3	1	192	96	1.5M
CoarseStage.SemsegLinker	5	3	1	720	96	2.8M
CoarseStage.DepthPtfldwDcdr	4	1	1	1	96	46.0k
CoarseStage.OccRefmnt	12	3	1	176	1	167.0k
CoarseStage.Preds	3	FC	-	96	C_{sem}	4.7k
MediumStage.SparseConv3d	25	3	1	139	48	1.2M
MediumStage.GRUConv3d	3	3	1	96	48	387.2k
MediumStage.SemsegLinker	5	3	1	360	48	689.9k
MediumStage.OccRefmnt	12	3	1	88	1	133.2k
MediumStage.Preds	3	FC	-	48	C_{sem}	2.6k
FineStage.SparseConv3d	25	3	1	75	24	295.1k
FineStage.GRUConv3d	3	3	1	48	24	96.8k
FineStage.SemsegLinker	5	3	1	216	24	218.0k
FineStage.OccRefmnt	12	3	1	48	1	117.9k
FineStage.Preds	3	FC	-	24	C_{sem}	1.6k
Total	195	-	-	-	-	12.73M

Table 1. **CDRNet network architecture details.** K denotes kernel length, S denotes stride, C_{in} denotes the input channel and C_{out} denotes the output channel. FC is short for fully-connected, meaning that the module contains no convolutional kernel. $C_{set} = \{80, 40, 24\}$ to create three stages of features respectively while C_{sem} denotes the number of semantic categories of the training dataset, which equals to 20 for ScanNet [2]’s valid annotation. These modules are ranked chronologically from top to down according to the computational graph.

Method	Abs. Diff.↓	Abs. Rel.↓	Sq. Rel.↓	RMSE↓	$\delta < 1.25 \uparrow$
MVDepthNet [16]	0.191	0.098	0.061	0.293	89.6
MaGNet [1]	0.141	0.081	0.030	0.196	93.1
3DVNet [12]	0.079	0.040	0.015	0.154	97.5
Atlas [10]	0.124	0.065	0.043	0.251	93.6
NeuralRecon [15]	0.106	0.065	0.031	0.195	94.8
SimpleRecon [14]	0.089	0.043	0.013	0.147	97.8
Ours	0.084	0.052	0.026	0.156	96.1

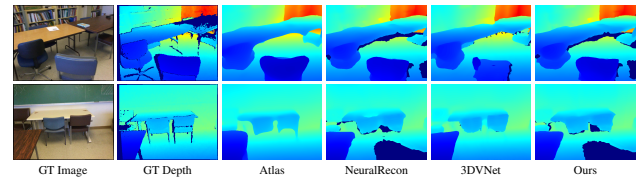


Figure 3. **Quantative and qualitative depth estimation results on ScanNet.** We compare our method with both state-of-the-art depth estimation and volumetric approaches in the upper and lower parts of the table, respectively. Our method is comparable with these representatives in terms of 2D depth estimation, justifying its effectiveness on 3D reconstruction.

GRU. At the fine stage, the output truncation distance of our predicted TSDF value is set as 12 centimeters.

Loss Design. In addition to the reason mentioned in Sec. 3, we also intend to retain the strongest supervisory signals on the coarse stage since it has the strongest level of semantic information with the greatest receptive fields. Therefore, we set $\alpha = \{1, 0.8, 0.64\}$. On the other hand, we balance the semantic volume learning with TSDF volume learning

by scaling down β , i.e., $\beta = \{0.1, 0.08, 0.064\}$. γ and μ is set as 0.5 and 0.1 for each s for the same reason.

Metrics and Evaluation Protocol. We evaluate mIoU by creating the confusion matrix as suggested in the official ScanNet [2] evaluation script. To achieve a better protocol for comparison, the ground-truth mesh is needed be directly used to generate point clouds and conduct point-wise comparison, rather with unnecessary post processing. Atlas [10] compared their post-processed ground truth using TSDF fusion to create new mesh from ground-truth depths, sometimes leading to implausible performance, e.g., on Recall. We avoid this issue by staying with the vanilla ground-truth meshes themselves for evaluation.

5. Discussion on Performance and Impacts

Further Experimental Results. To further justify the accuracy of the reconstructed 3D mesh, we also measure standard 2D depth estimation quality to validate the 3D reconstruction result. In Fig. 3, we compare both state-of-the-art depth estimation methods and volumetric methods in depth metrics. Notice that due to the low albedo and occlusion occurred in the data collection process of the ScanNet [2] ground truth, there are many holes in some of the ground-truth mesh. This problem residing in the dataset causes many incorrect reconstruction and segmentation results for both of our baseline methods, as shown in Fig. 5. For instance, with its over-smooth meshing characteristic, Atlas tends to create artifacts as shown in the third row of our testing result. Meanwhile, NeuralRecon + Semantic-Heads suffers from the contradiction during the learning of TSDF and semantic volume at the same time. On the contrary, our method with 2D refinements from both depth and semantic predictions can effectively alleviate such an issue.

Typical Applications to 3D Perception. Given the in-situ interaction between robots and the environment, the digitization that robots learn about the environment is regarded as simultaneous localization and mapping (SLAM) in the robotics field. Furthermore, with such digitization, the interpretation of the target environment such as semantic categories classification is a significant first step to perform embodied artificial intelligence (AI) on robots [11, 13, 17].

Requirements of Being Real-Time. With a handheld monocular-camera cellphone as the input source, we assume the average human pace as 1.4 m/s. Given the length of each fragment is 96 voxels in CDRNet and voxel size is 4 cm, we can calculate the maximum update time for CDRNet one fragment is $T_{\text{update}} = \frac{96 \times 0.04 \text{ m}}{1.4 \text{ m/s}} = 2.743 \text{ s}$. As tested out in multiple trials of our experiments, the total latency under WiFi data transmission and data streaming on the server is around 2 s. Thus, the processed time that can be accepted for one fragment computing of CDR-

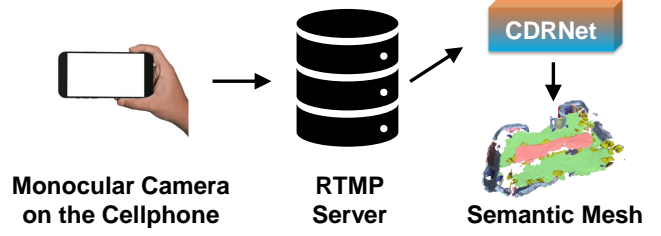


Figure 4. **The pipeline of real-time 3D perception with CDRNet.** RGB information and camera transform matrices are extracted utilizing the IMU and camera on the cellphone and passed to the CDRNet model on GPU for processing via the RTMP server.

Net is $T_{\text{proc}} = T_{\text{update}} - 2 \text{ s} = 0.743 \text{ s}$. On average, each fragment entertains 67 frames (as $N_k = 9$ with key-frame selection), which makes the FPS of the according T_{proc} as $\text{FPS}_{\text{real-time}} = \frac{67 \text{ Frames}}{0.743 \text{ s}} = 90.17$.

Therefore, CDRNet can be regarded as a real-time method by achieving 158 FPS as shown in Table 2, whereas the other SOTA method, Atlas, fails to achieve $\text{FPS}_{\text{real-time}}$ to be in real time.

Interactive Real-Time Perception Demo. In Fig. 4, we present a progressive 3D perception system that is capable of the real-time interaction with monocular camera on cellphones. We chose ARKit [6] on iOS platform to synchronize the RGB, camera intrinsics, and camera pose recordings, meanwhile there is also a counterpart in Android platform named ARCore [7]. We built up a real-time messaging protocol (RTMP) server to conduct the streaming to the GPU server for the real-time input uploading. After the input fragment is recorded on the cellphone, each fragment input is passed to CDRNet thanks to the RTMP server, and the semantic mesh inference within the current FBV can be achieved readily. Please refer to the video *Jcdrnet_demo_medium.mp4* for details.

References

- [1] Gwangbin Bae, Ignas Budvytis, and Roberto Cipolla. Multi-view depth estimation by fusing single-view depth probability with multi-view geometry. In *CVPR*, pages 2842–2851, 2022. 3
- [2] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, pages 5828–5839, 2017. 3, 4
- [3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. 2
- [4] Ian Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. *ICLR*, 2015. 2
- [5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 3

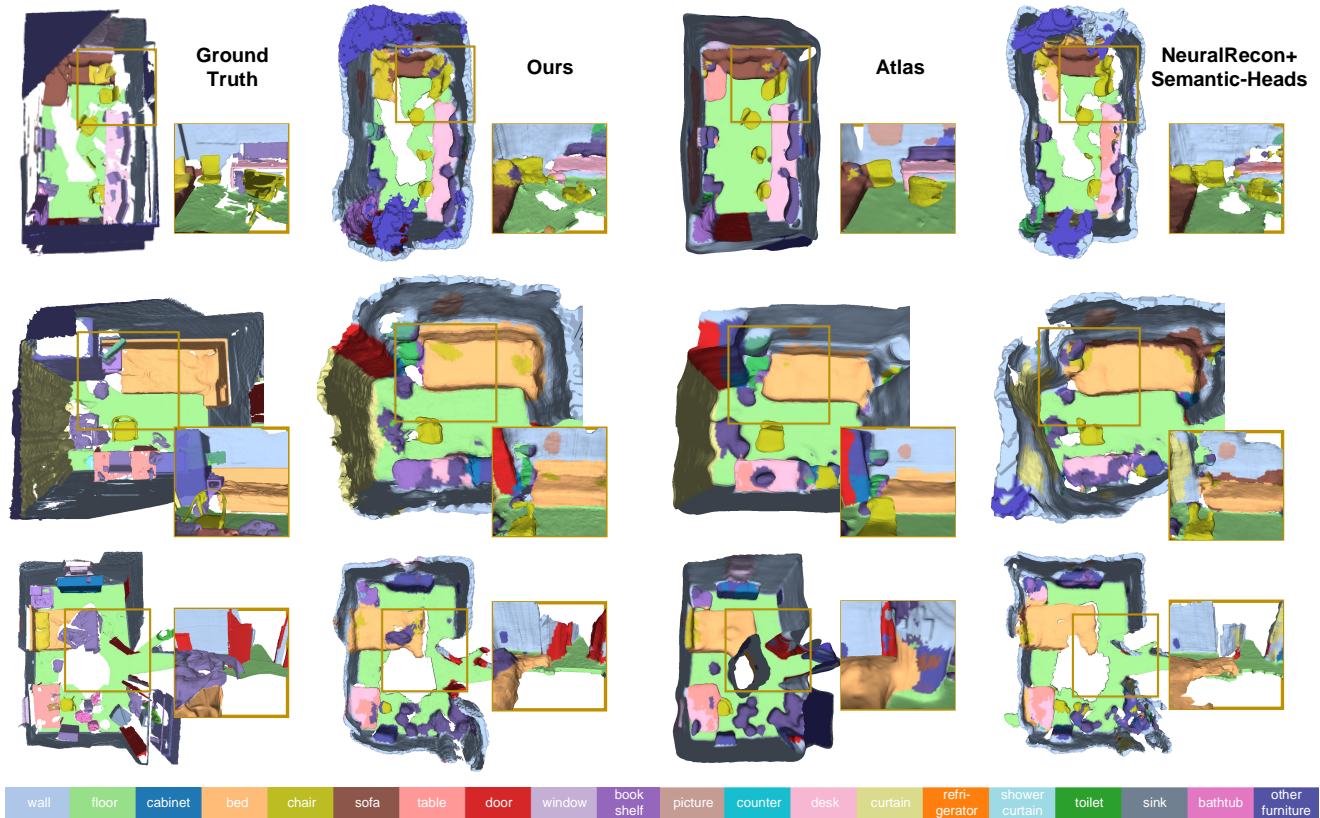


Figure 5. **Additional qualitative 3D reconstruction and semantic segmentation results on ScanNet.** Thanks to the feature refinement brought by the utilization of both depth prior and semantic prior, our method yields better 3D reconstruction and 3D semantic segmentation results at the same time, comparing to two main baseline methods, Atlas [10] and NeuralRecon [15].

- [6] Apple Inc. ARKit Developer Document. <https://developer.apple.com/documentation/arkit> (Mar. 2023). 4
- [7] Alphabet Inc. Overview of ARCore and Dev. <https://developers.google.com/ar/develop> (Mar. 2023). 4
- [8] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *NeurIPS*, 31, 2018. 2
- [9] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 2117–2125, 2017. 3
- [10] Zak Murez, Tarrence van As, James Bartolozzi, Ayan Sinha, Vijay Badrinarayanan, and Andrew Rabinovich. Atlas: End-to-end 3d scene reconstruction from posed images. In *ECCV*, pages 414–431. Springer, 2020. 3, 4, 5
- [11] Medhini Narasimhan, Erik Wijmans, Xinlei Chen, Trevor Darrell, Dhruv Batra, Devi Parikh, and Amanpreet Singh. Seeing the un-scene: Learning amodal semantic maps for room navigation. In *ECCV*, pages 513–529. Springer, 2020. 4
- [12] Alexander Rich, Noah Stier, Pradeep Sen, and Tobias Höllerer. 3dvnnet: Multi-view depth prediction and volumetric refinement. In *2021 International Conference on 3D Vision (3DV)*, pages 700–709. IEEE, 2021. 3
- [13] 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*, pages 9339–9347, 2019. 4
- [14] Mohamed Sayed, John Gibson, Jamie Watson, Victor Prisacariu, Michael Firman, and Clément Godard. Simplerecon: 3d reconstruction without 3d convolutions. In *ECCV*, 2022. 3
- [15] Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, and Hujun Bao. Neuralrecon: Real-time coherent 3d reconstruction from monocular video. In *CVPR*, pages 15598–15607, 2021. 3, 5
- [16] Kaixuan Wang and Shaojie Shen. Mvdepthnet: Real-time multiview depth estimation neural network. In *2018 International Conference on 3D Vision (3DV)*, pages 248–257. IEEE, 2018. 3
- [17] Sriram Yenamandra, Arun Ramachandran, Karmesh Yadav, Austin Wang, Mukul Khanna, Theophile Gervet, Tsung-Yen Yang, Vidhi Jain, Alexander William Clegg, John Turner, et al. Homerobot: Open-vocabulary mobile manipulation. *arXiv preprint arXiv:2306.11565*, 2023. 4