Supplemental Material
RevealNet: Seeing behind objects in RGB-D Scans

Ji Hou   Angela Dai   Matthias Nießner
Technical University of Munich

1. Appendix

In this appendix, we detail our RevealNet network architecture in Section 2; in Section 3, we provide runtime results of our approach; in Section 4, we discuss on differences between model-fitting and prediction-based approaches regarding object-level completion in RGB-D scans; in Section 5 we show the ablation study that how degrees of completeness in the input data influence the semantic instance complete performance.

2. Network Architecture

small anchors	big anchors
(9, 10, 9)	(47, 20, 23)
(17, 21, 17)	(23, 20, 47)
(12, 19, 13)	(16, 18, 30)
(16, 12, 15)	(17, 38, 17)
	(30, 18, 16)

Table 1: Anchor sizes used for region proposal on the ScanNet dataset [3]. Sizes are given in voxel units, with voxel resolution of ≈ 4.69cm

small anchors	big anchors
(8, 6, 8)	(12, 12, 40)
(22, 22, 16)	(8 , 60, 40)
(12, 12, 20)	(38, 12, 16)
	(62, 8 , 40)
	(46, 8 , 20)
	(46, 44, 20)
	(14, 38, 16)

Table 2: Anchor sizes (in voxels) used for SUNCG [5] region proposal. Sizes are given in voxel units, with voxel resolution of ≈ 4.69cm

Table 6 details the layers used in our backbone. 3D-RPN, classification head, and mask completion head are described in Table 7. Additionally, we leverage the residual blocks in our backbone, which is listed in Table 5. Note that both the backbone and mask completion head are fully-convolutional. For the classification head, we use several fully-connected layers; however, we leverage 3D RoI-pooling on its input, so that we can run our method on large 3D scans of varying sizes in a single forward pass [4].

We additionally list the anchors used for the region proposal for our model trained on our ScanNet-based semantic instance completion benchmark [1, 3] and SUNCG [5] datasets in Tables 1 and 2, respectively. Anchors for each dataset are determined through k-means clustering of ground truth bounding boxes. The anchor sizes are given in voxels, where our voxel size is ≈ 4.69cm.

3. Inference Timing

In this section, we present the inference timing with and without color projection in Table 3 and 4. Note that our color projection layer currently projects the color signal into 3D space sequentially, and can be further optimized using CUDA, so that it can project the color features back to 3D space in parallel. A scan typically contains several hundreds of images; hence, this optimization could significantly further improve inference time.

physical size (m)	5.8 x 6.4	8.3 x 13.9	10.9 x 20.1
voxel resolution	124 x 136	176 x 296	232 x 428
forward pass (s)	0.15	0.37	0.72

Table 3: Inference time on entire scenes without color signal. Timings are given in seconds, physical sizes are given in meters and spatial sizes are given in voxel units, with voxel resolution of ≈ 4.69cm

.

physical size (m)	4.7 x 7.7	7.9 x 9.6	10.7 x 16.5
voxel resolution	100 x 164	168 x 204	228 x 352
image count	49	107	121
color projection (s)	1.43	5.16	11.78
forward pass (s)	0.19	0.34	0.64
total (s)	1.62	5.50	12.42

Table 4: Inference timing on entire large scans with RGB input. Timings are given in seconds, physical sizes are given in meters and spatial sizes are given in voxel units, with voxel resolution of ≈ 4.69cm

.

ResBlock	Input Layer	Type	Input Size	Output Size	Kernel Size	Stride	Padding
convres0	CNN feature	Conv3d	(N,X,Y,Z)	(N/2,X,Y,Z)	(1,1,1)	(1,1,1)	(0,0,0)
normres0	convres0	InstanceNorm3d	(N/2,X,Y,Z)	(N/2,X,Y,Z)	None	None	None
relures0	normres0	ReLU	(N/2,X,Y,Z)	(N/2,X,Y,Z)	None	None	None
convres1	relures0	Conv3d	(N/2,X,Y,Z)	(N/2,X,Y,Z)	(3,3,3)	(1,1,1)	(1,1,1)
normres1	convres1	InstanceNorm3d	(N/2,X,Y,Z)	(N/2,X,Y,Z)	None	None	None
relures1	normres1	ReLU	(N/2,X,Y,Z)	(N/2,X,Y,Z)	None	None	None
convres2	relures1	Conv3d	(N/2,X,Y,Z)	(N,X,Y,Z)	(1,1,1)	(1,1,1)	(0,0,0)
normres2	convres2	InstanceNorm3d	(N,X,Y,Z)	(N,X,Y,Z)	None	None	None
relures2	normres2	ReLU	(N,X,Y,Z)	(N,X,Y,Z)	None	None	None

Table 5: Residual block specification in RevealNet.

4. Model-fitting vs. Prediction-based Methods

In terms of object level completion in the RGB-D scan, We discuss about differences between model-fitting and prediction-based methods. Regarding model-fitting methods, *"Scan2CAD"* [1] and *"End-to-End CAD"* [2] define a CAD alignment task for which they require a predefined set of CAD models for each test scene; i.e., GT objects are given at test time and only alignment needs to be inferred (cf. Scan2CAD benchmark). Prediction-based method method, e.g. RevealNet, does not have the GT objects as input in the test time.

Semantic instance completion is fundamentally more flexible as it operates on a per-voxel basis (vs. fixed CAD models); i.e., allowing construction of much more true-to-observation geometry (e.g., necessary in the context of robotics). Since prediction-based methods complete the missing surface based on observed geometry that performs as an anchor, RevealNet easily overlaps with the ground truth surface, whereas model-fitting methods, which predicts rotation/scale/translation, could have little overlap with the ground truth surface due to slight misalignment (e.g. 10 degrees rotation error). In addition, we want to highlight that CAD model alignment does not help their detection performance, but instance completion in RevealNet does.

5. Performance Study over Degrees of Completeness

Both our SUNCG and ScanNet settings have a large variety of incompleteness. We show a histogram in Fig. 1 of our current results on ScanNet (numbers split from Tab. 1 main paper). From the histogram, We show that with more complete geometry input, semantic instance completion task becomes easier.

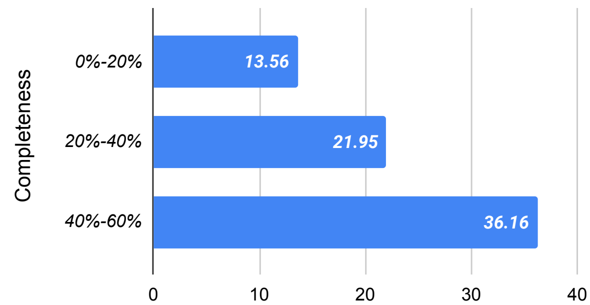

Figure 1: Results of Tab. 1 split by levels of object completeness in mAP@0.5; more complete input is easier.

References

[1] Armen Avetisyan, Manuel Dahnert, Angela Dai, Manolis Savva, Angel X. Chang, and Matthias Nießner. Scan2cad: Learning cad model alignment in rgb-d scans. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2019. 1, 2

[2] Armen Avetisyan, Angela Dai, and Matthias Nießner. End-to-end cad model retrieval and 9dof alignment in 3d scans. In *ICCV*, 2019. 2

[3] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. 1

[4] Ji Hou, Angela Dai, and Matthias Nießner. 3d-sis: 3d semantic instance segmentation of rgb-d scans. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2019. 1

[5] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1

BackBone	Input Layer	Type	Input Size	Output Size	Kernel Size	Stride	Padding
geometry0	TSDF	Conv3d	(2,96,48,96)	(32,48,24,48)	(2,2,2)	(2,2,2)	(0,0,0)
norm0	geometry0	InstanceNorm3d	(32,48,24,48)	(32,48,24,48)	None	None	None
relu0	norm0	ReLU	(32,48,24,48)	(32,48,24,48)	None	None	None
block0	relu0	ResBlock	(32,48,24,48)	(32,48,24,48)	None	None	None
color1	CNN feature	Conv3d	(128,96,48,96)	(32,48,24,48)	(2,2,2)	(2,2,2)	(0,0,0)
norm1	color1	InstanceNorm3d	(32,48,24,48)	(32,48,24,48)	None	None	None
relu1	norm1	ReLU	(32,48,24,48)	(32,48,24,48)	None	None	None
block1	relu1	ResBlock	(32,48,24,48)	(32,48,24,48)	None	None	None
concat2	(block0,block1)	Concatenate	(32,48,24,48)	(64,48,24,48)	None	None	None
combine2	concat2	Conv3d	(64,48,24,48)	(128,24,12,24)	(2,2,2)	(2,2,2)	(0,0,0)
norm2	combine2	InstanceNorm3d	(128,24,12,24)	(128,24,12,24)	None	None	None
relu2	norm2	ReLU	(128,24,12,24)	(128,24,12,24)	None	None	None
block2	relu2	ResBlock	(128,24,12,24)	(128,24,12,24)	None	None	None
encoder3	block2	Conv3d	(128,24,12,24)	(128,24,12,24)	(3,3,3)	(1,1,1)	(1,1,1)
norm3	combine3	InstanceNorm3d	(128,24,12,24)	(128,24,12,24)	None	None	None
relu3	norm3	ReLU	(128,24,12,24)	(128,24,12,24)	None	None	None
block3	relu3	ResBlock	(128,24,12,24)	(128,24,12,24)	None	None	None
skip4	(block, block3)	Conv3d	(128,24,12,24)	(64,48,24,48)	(2,2,2)	(2,2,2)	(0,0,0)
norm4	combine4	InstanceNorm3d	(64,48,24,48)	(64,48,24,48)	None	None	None
relu4	norm4	ReLU	(64,48,24,48)	(64,48,24,48)	None	None	None
block4	relu4	ResBlock	(64,48,24,48)	(64,48,24,48)	None	None	None
concat5	(block2,block4)	Concatenate	(64,48,24,48)	(128,48,24,48)	None	None	None
decoder5	block5	ConvTranspose3d	(128,48,24,48)	(32,96,48,96)	(2,2,2)	(2,2,2)	(0,0,0)
norm5	combine5	InstanceNorm3d	(32,96,48,96)	(32,96,48,96)	None	None	None
relu5	norm5	ReLU	(32,96,48,96)	(32,96,48,96)	None	None	None
block5	relu5	ResBlock	(32,96,48,96)	(32,96,48,96)	None	None	None
proxy5	block5	ConvTranspose3d	(32,96,48,96)	(1,96,48,96)	(1, 1, 1)	(1,1,1)	(0,0,0)

Table 6: Backbone layer specifications in RevealNet.

RPN	Input Layer	Type	Input Size	Output Size	Kernel Size	Stride	Padding
rpn6	block2	Conv3d	(128,24,12,24)	(256,24,12,24)	(3,3,3)	(1,1,1)	(1,1,1)
norm6	rpn6	InstanceNorm3d	(256,24,12,24)	(256,24,12,24)	None	None	None
relu6	norm6	ReLU	(256,24,12,24)	(256,24,12,24)	None	None	None
rpncls7a	relu6	Conv3d	(256,24,12,24)	(8,24,12,24)	(1,1,1)	(1,1,1)	(0,0,0)
norm7a	rpncls7a	InstanceNorm3d	(8,24,12,24)	(8,24,12,24)	None	None	None
rpnbbox7b	relu6	Conv3d	(24,24,12,24)	(24,24,12,24)	(1,1,1)	(1,1,1)	(0,0,0)
norm7b	rpnbbox7b	InstanceNorm3d	(24,24,12,24)	(24,24,12,24)	None	None	None
rpn8	block3	Conv3d	(128,24,12,24)	(256,24,12,24)	(3,3,3)	(1,1,1)	(1,1,1)
norm8	rpn8	InstanceNorm3d	(256,24,12,24)	(256,24,12,24)	None	None	None
relu8	norm8	ReLU	(256,24,12,24)	(256,24,12,24)	None	None	None
rpncls9a	relu8	Conv3d	(256,24,12,24)	(8,24,12,24)	(1,1,1)	(1,1,1)	(0,0,0)
norm9a	rpncls9a	InstanceNorm3d	(10,24,12,24)	(10,24,12,24)	None	None	None
rpnbbox9b	relu8	Conv3d	(30,24,12,24)	(30,24,12,24)	(1,1,1)	(1,1,1)	(0,0,0)
norm9b	rpnbbox9b	InstanceNorm3d	(30,24,12,24)	(30,24,12,24)	None	None	None
Class Head	Input Layer	Type	Input Size	Output Size	Kernel Size	Stride	Padding
roipool10	block2/block3	RoI Pooling	(64,arbitrary)	(64, 4, 4, 4)	None	None	None
flat10	roipool10	Flat	(64,4,4,4)	(4096)	None	None	None
cls10a	flat10	Linear	(4096)	(256)	None	None	None
relu10a	cls10a	ReLU	(256)	(256)	None	None	None
cls10b	relu10a	Linear	(256)	(128)	None	None	None
relu10b	cls10b	ReLU	(128)	(128)	None	None	None
cls10c	relu10b	Linear	(128)	(128)	None	None	None
relu10c	cls10c	ReLU	(128)	(128)	None	None	None
clscls10	relu10c	Linear	(128)	(8)	None	None	None
clsbbox10	relu10c	Linear	(128)	(48)	None	None	None
Mask Head	Input Layer	Type	Input Size	Output Size	Kernel Size	Stride	Padding
mask11	block2/block3	Conv3d	(N,arbitrary)	(N,arbitrary)	(9,9,9)	(1,1,1)	(4,4,4)
norm11	mask11	InstanceNorm3d	(N,arbitrary)	(N,arbitrary)	None	None	None
relu11	norm11	ReLU	(N,arbitrary)	(64,arbitrary)	None	None	None
mask12	relu11	Conv3d	(N,arbitrary)	(N,arbitrary)	(7,7,7)	(1,1,1)	(3,3,3)
norm12	mask12	InstanceNorm3d	(N,arbitrary)	(N,arbitrary)	None	None	None
relu12	norm12	ReLU	(N,arbitrary)	(64,arbitrary)	None	None	None
mask13	relu12	Conv3d	(N,arbitrary)	(N,arbitrary)	(5,5,5)	(1,1,1)	(2,2,2)
norm13	mask13	InstanceNorm3d	(N,arbitrary)	(N,arbitrary)	None	None	None
relu13	norm13	ReLU	(N,arbitrary)	(64,arbitrary)	None	None	None
mask14	relu13	Conv3d	(N,arbitrary)	(N,arbitrary)	(3,3,3)	(1,1,1)	(1,1,1)
norm14	mask14	InstanceNorm3d	(N,arbitrary)	(N,arbitrary)	None	None	None
relu14	norm14	ReLU	(N,arbitrary)	(64,arbitrary)	None	None	None
mask15	relu14	Conv3d	(N,arbitrary)	(N,arbitrary)	(1,1,1)	(1,1,1)	(0,0,0)

Table 7: Head layer specifications of RPN, Classification and Mask Completion in RevealNet.