SGRec3D: Self-Supervised 3D Scene Graph Learning via Object-Level Scene Reconstruction — Supplementary Material —

Sebastian Koch[1,2,3] Pedro Hermosilla[4] Narunas Vaskevicius[1,2]
Mirco Colosi[2] Timo Ropinski[3]

[1]Bosch Center for Artificial Intelligence [2]Robert Bosch Corporate Research
[3]University of Ulm [4] TU Wien

kochsebastian.com/sgrec3d

This document supplements our work *SGRec3D: Self-Supervised 3D Scene Graph Learning via Object-Level Scene Reconstruction* by providing **(i)** reproducibility information on our implementation and training details (Sec. 1), **(ii)** more details on the dataset and its pre-processing (Sec. 2), **(iii)** further ablations on our architecture design (Sec. 3), **(iv)** a direct qualitative comparison between our method with and without pre-training for 3D scene graph prediction (Sec. 4), **(v)** additional 3D scene graph predictions using our method(Sec. 5), **(vi)** additional scene generations from our method (Sec. 6),

1. Reproducibility Details

1.1. Network

Our encoder consists of two PointNets which pass features of size 256 to a 4-layer GCN, where $g_1(\cdot)$ and $g_2(\cdot)$ are composed of a linear layer followed by a ReLU activation. Additionally, the bounding boxes of the object instances are encoded via a linear layer and are appended to the initial features from the PointNets. The encoder GCN is followed by object and predicate prediction MLPs, consisting of 3 linear layers with batch normalization and ReLU activation.

During pre-training, the resulting features are fed into the decoder part of our network, which consists of a 3-layer GCN with the same $g_1(\cdot)$, $g_2(\cdot)$ MLPs. After the graph convolution, the GCN features are passed to two different heads. One is a Box-Head consisting of a 3-layer MLP with batch normalization and ReLU activation, outputting 7 box parameters $(w, l, h, c_x, c_y, c_z, \theta)$. The other is a Shape-MLP with 3 linear layers, batch normalization and ReLU activation, outputting a 1024-dimensional shape latent code. The original object point clouds are additionally fed into the encoder of a pre-trained AtlasNet, which produces the target shape code for our model.

1.2. Training

The model is trained with a batch size of 4, using an Adam optimizer with a learning rate of 0.0001 and a reduce-on-plateau learning rate scheduler. The pre-training is performed until the validation loss converges. We pre-train our method for approximately 35 epochs until the validation loss for the reconstruction task converges. Similarly, during fine-tuning, we monitor the validation loss. Once the validation loss converges, which occurs after ~20 epochs, we evaluate the scene graph prediction performance by calculating the metrics introduced in the paper on the validation set. Further training results in overfitting, indicated by the validation loss for both object prediction and predicate prediction. Overfitting occurs faster for the non-pre-trained model after around ~15 epochs. However, validation loss and evaluation metrics are worse than our pre-trained model. Further training with smaller learning rates does not improve the results.

The training is performed on 2 NVIDIA Tesla V100 GPUs with 32 GB Memory.

1.3. Losses

During pre-training we use the following reconstruction loss for all objects $i \in N$ in the scene

$$\mathcal{L}_{\text{rec}} = \frac{1}{N}\sum_{i=1}^{N}\left(\eta_1\|\hat{b}_i - b_i\|_1 + \eta_2 CE(\hat{\theta}_i, \theta_i) + \eta_3\|\hat{e}_i - e_i\|_1\right) \tag{1}$$

where $\hat{b}_i, b_i$ are the predicted and ground truth bounding box parameters respectively, $\hat{\theta}_i, \theta_i$ the predicted and ground truth yaw angle of the bounding box and $\hat{e}_i$ the predicted shape code from our model, while e_i is the shape code provided by AtlasNet. We choose $\eta_1 = 0.4$, $\eta_2 = 0.2$, $\eta_3 = 0.4$.

During fine-tuning, we use the following loss for nodes

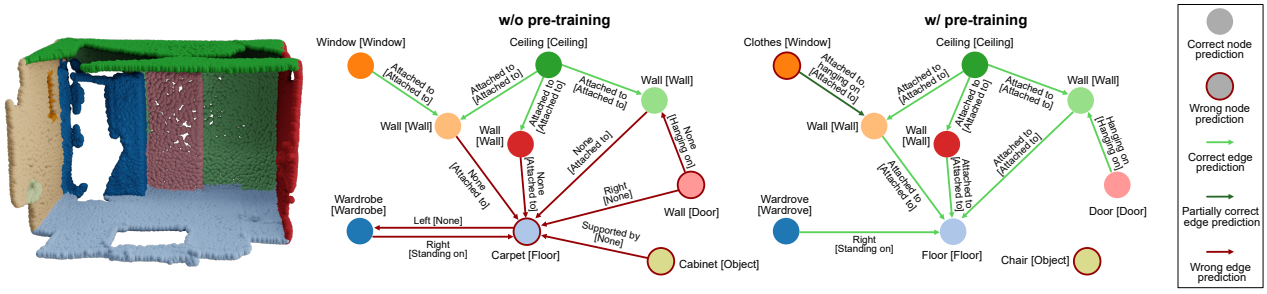


Figure 1. **Qualitative evaluation of the effects of pre-training on 3DSSG.** A qualitative comparison of a scene graph predicted for a 3D scene (*left*), with no pre-training (*middle*) and with pre-training (*right*). We visualize the top-1 object class prediction for each node and the predicates with a probability greater than 0.5 for each edge. Ground truth labels are shown in square brackets. SGRec3D results in 7/9 correct nodes and 9/9 (partially) correct predicate predictions, while the baseline only reaches 5/9 and 4/9 respectively, despite predicting many false positive predicates.

$i \in N$ and edges $j \in M$

$$\mathcal{L}_{\text{SG}} = \frac{1}{N}\sum_{i=1}^{N}\lambda_1 CE\left(\hat{o}_i, o_i\right) + \frac{1}{M}\sum_{j=1}^{M}\lambda_2 BCE\left(\hat{p}_j, p_j\right) \quad (2)$$

where $\hat{o}_i, o_i$ are the predicted and ground truth object node classes and $\hat{p}_j, p_j$ are the predicted and ground truth predicates for edge j. We choose $\lambda_1 = 0.1$ and $\lambda_2 = 1.0$.

To deal with class imbalance during fine-tuning, we use a focal loss for both loss terms

$$\mathcal{L} = -\alpha_t(1-p_t)^{\gamma}\log p_t \quad (3)$$

with $\alpha = 0.25$ and $\gamma = 2$. However, we do not use manual class weighting based on object and predicate occurrences like SGFN [5].

2. Dataset Details

Following Wald et al. [4], our method operates on scene splits with 4-9 objects instead of taking the full 3D scene as input. For comparability, we use the exact same pre-split scenes published by Wald et al. [4]. A full list of the 160 objects and 26 predicates used for the evaluation is in the authors' repository under subset data[1].

For the ScanNet [2] and S3DIS [1] pre-training, we use the most updated versions of each dataset available at the time of publishing this paper. For uniform training samples, we also generate scene splits for the additional datasets to emulate the 3DSSG dataset [4] as best as possible. Moreover, before feeding object point clouds and pairs of objects' point clouds into the PointNets [3], we apply farthest-point-sampling to downsample the point clouds of each object to at most 1000 points.

3. Architecture Ablations

In Tab. 1 we provide ablations examining the design choices for our pre-training method. We present results for: Our method without pre-training; Our pre-training only using the shape-loss reconstruction term; Our pre-training only using the bounding box reconstruction term. Further, we provide architecture ablations for our method without utilizing a GCN and without using our proposed skip-connection from Sec. 3.

Method	Object R@5	Object mR@5	Predicate R@3	Predicate mR@3
Ours (w/o pre-train)	0.63	0.30	0.94	0.57
Ours (shape-loss only)	0.77	0.39	0.94	0.49
Ours (box-loss only)	0.76	0.35	0.96	0.59
Ours (w/o GCN)	0.75	0.31	0.94	0.48
Ours (w/o skip connection)	0.77	0.40	0.96	0.60

Table 1. **Ablations point cloud pre-training approaches**.

We observe that only using the shape-loss during pre-training greatly improves object prediction performance. However, the impact on predicate prediction is small. Only using the bounding box loss term during pre-training improves objects and predicates alike, but the results are worse than using the shape-loss and bounding box-loss together. A fundamental aspect of our method is the introduced GCN. Without a GCN as a backbone, we observe that our pre-training becomes considerably less effective in the learning of predicates. In our architecture, we further introduce a singular skip-connection in Eq. 6. This skip-connection serves the role of conserving more context features from the encoder. This skip-connection improves the reconstruction loss and consequently pre-training effectiveness for the entire encoder.

4. Qualitative effect of pre-training

in Fig. 1, we provide a direct comparison between our method pre-trained using our pre-training approach and the

[1] https://github.com/3DSSG/3DSSG.github.io/

same method trained from scratch on 3D scene graph prediction. We observe that using our pre-training drastically improves the prediction accuracy of nodes and edges. The predicted 3D scene graph using our pre-training is near perfect except for two misclassified objects, while the method trained from scratch predicts many incorrect predicates and also misclassified objects.

5. Scene Graph Results

In Fig. 2, we provide additional scene graph visualizations. We observe that our network is able to produce almost perfect scene graphs in very diverse scenes. Some common misclassification cases include incorrect edges where either the ground truth is none and our network predicts a relationship or our network does not predict a relationship when it is present.

6. Scene Generation Results

In Fig. 3, we provide additional scene reconstructions. The shown scene generations support those shown in the paper. Although the generated shapes are not perfect, our model seems to preserve the relationships in the original scenes.

References

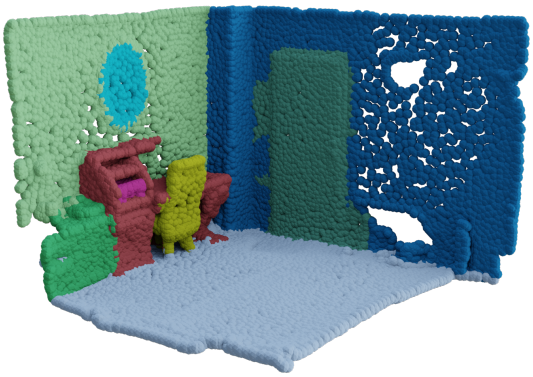
[1] Iro Armeni, Sasha Sax, Amir R. Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *CoRR*, abs/1702.01105, 2017. 2

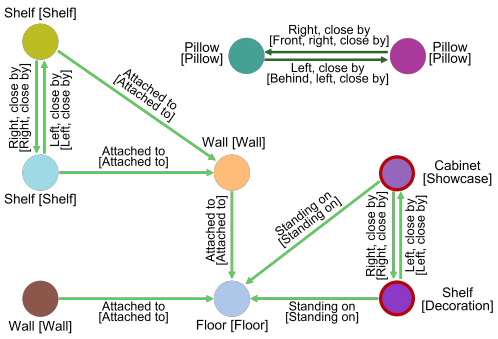
[2] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5828–5839, June 2017. 2

[3] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

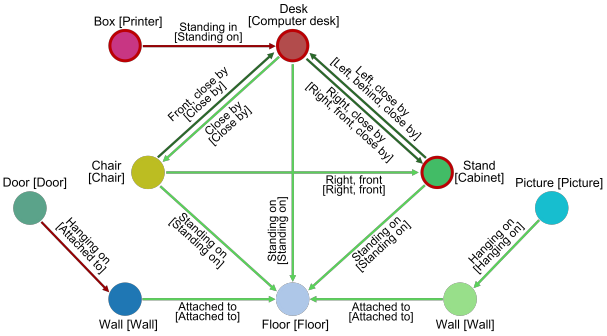
[4] Johanna Wald, Helisa Dhamo, Nassir Navab, and Federico Tombari. Learning 3d semantic scene graphs from 3d indoor reconstructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[5] Shun-Cheng Wu, Johanna Wald, Keisuke Tateno, Nassir Navab, and Federico Tombari. Scenegraphfusion: Incremental 3d scene graph prediction from rgb-d sequences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7515–7525, June 2021. 2

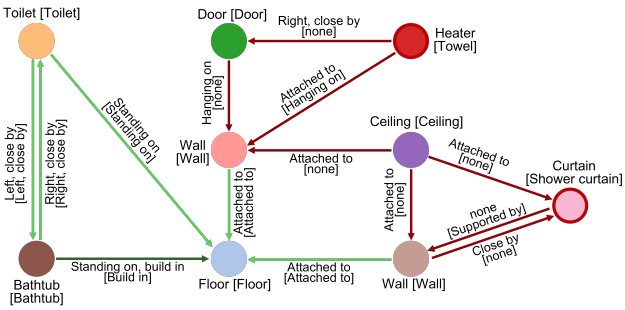



Shelf [Shelf]
Pillow [Pillow]
Right, close by [Front, right, close by]
Left, close by [Behind, left, close by]
Pillow [Pillow]
Right, close by [Right, close by]
Left, close by [Left, close by]
Attached to [Attached to]
Wall [Wall]
Attached to [Attached to]
Shelf [Shelf]
Cabinet [Showcase]
Attached to [Attached to]
Standing on [Standing on]
Right, close by [Right, close by]
Left, close by [Left, close by]
Wall [Wall]
Attached to [Attached to]
Floor [Floor]
Standing on [Standing on]
Shelf [Decoration]

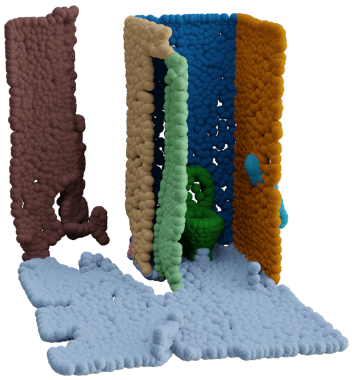
(a)

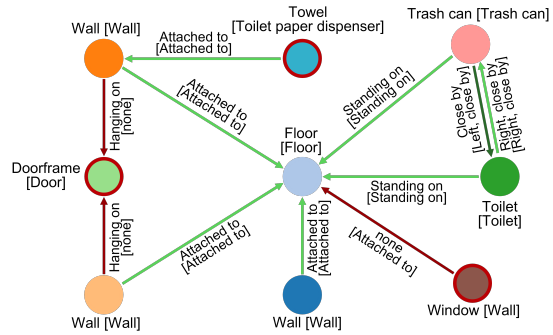

Box [Printer]
Standing in [Standing on]
Desk [Computer desk]
Front, close by [Close by]
Close by [Close by]
[Left, close by [Left, behind, close by]
Right, close by [Right, front, close by]
Chair [Chair]
Stand [Cabinet]
Right, front [Right, front]
Door [Door]
Picture [Picture]
Hanging on [Attached to]
Standing on [Standing on]
Standing on [Standing on]
Standing on [Standing on]
Hanging on [Hanging on]
Wall [Wall]
Attached to [Attached to]
Floor [Floor]
Attached to [Attached to]
Wall [Wall]

(b)


Toilet [Toilet]
Door [Door]
Right, close by [none]
Heater [Towel]
Left, close by [Left, close by]
Right, close by [Right, close by]
Standing on [Standing on]
Hanging on [none]
Attached to [Hanging on]
Wall [Wall]
Ceiling [Ceiling]
Attached to [none]
Attached to [none]
Curtain [Shower curtain]
Attached to [Attached to]
Attached to [none]
none [Supported by]
Close by [none]
Bathtub [Bathtub]
Standing on, build in [Build in]
Floor [Floor]
Attached to [Attached to]
Wall [Wall]

(c)




Wall [Wall]
Attached to [Attached to]
Towel [Toilet paper dispenser]
Trash can [Trash can]
Hanging on [none]
Attached to [Attached to]
Standing on [Standing on]
Close by [Left, close by]
Right, close by [Right, close by]
Floor [Floor]
Doorframe [Door]
Standing on [Standing on]
Toilet [Toilet]
Hanging on [none]
Attached to [Attached to]
Attached to [Attached to]
none [Attached to]
Wall [Wall]
Wall [Wall]
Window [Wall]

(d)

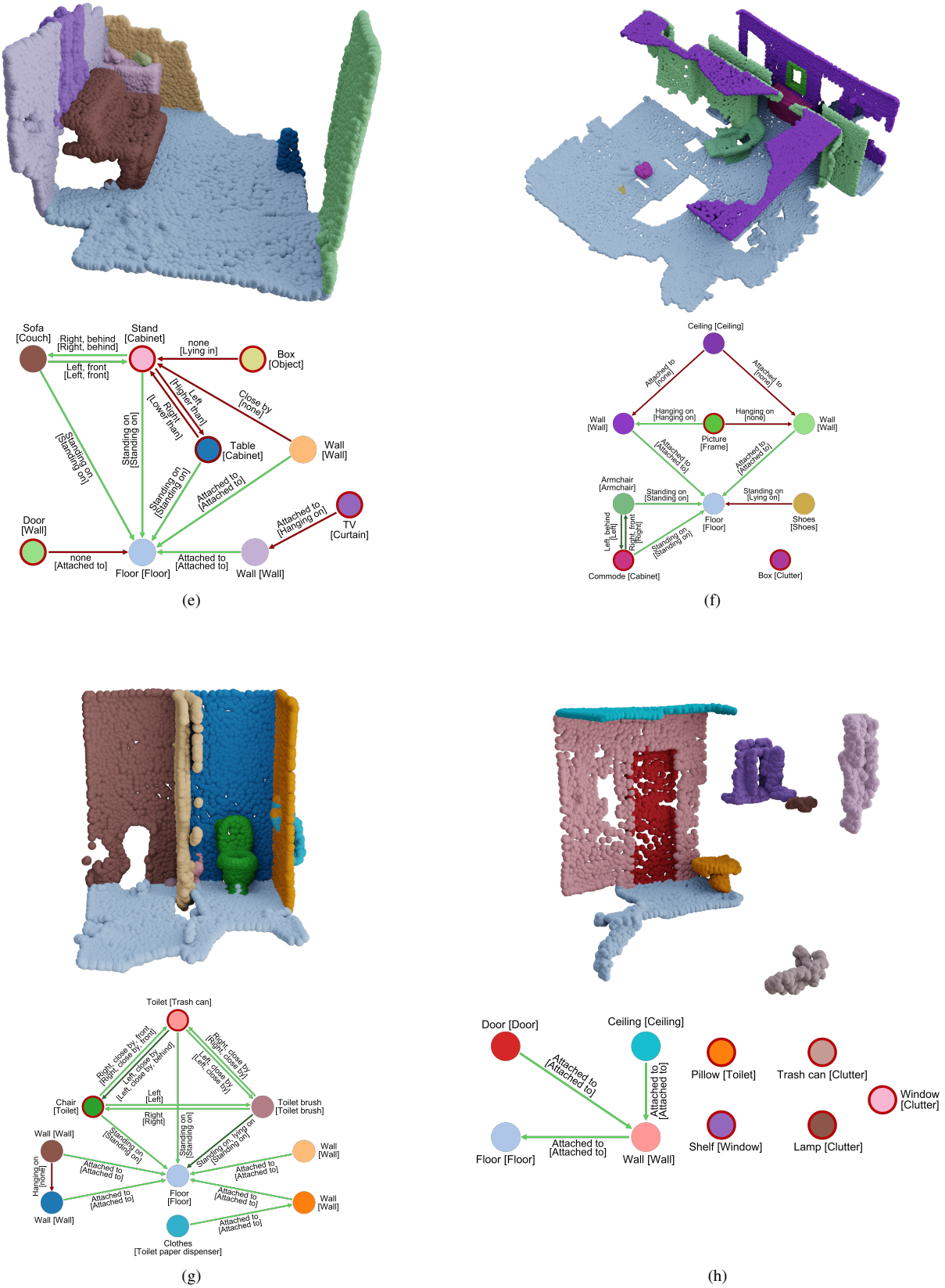
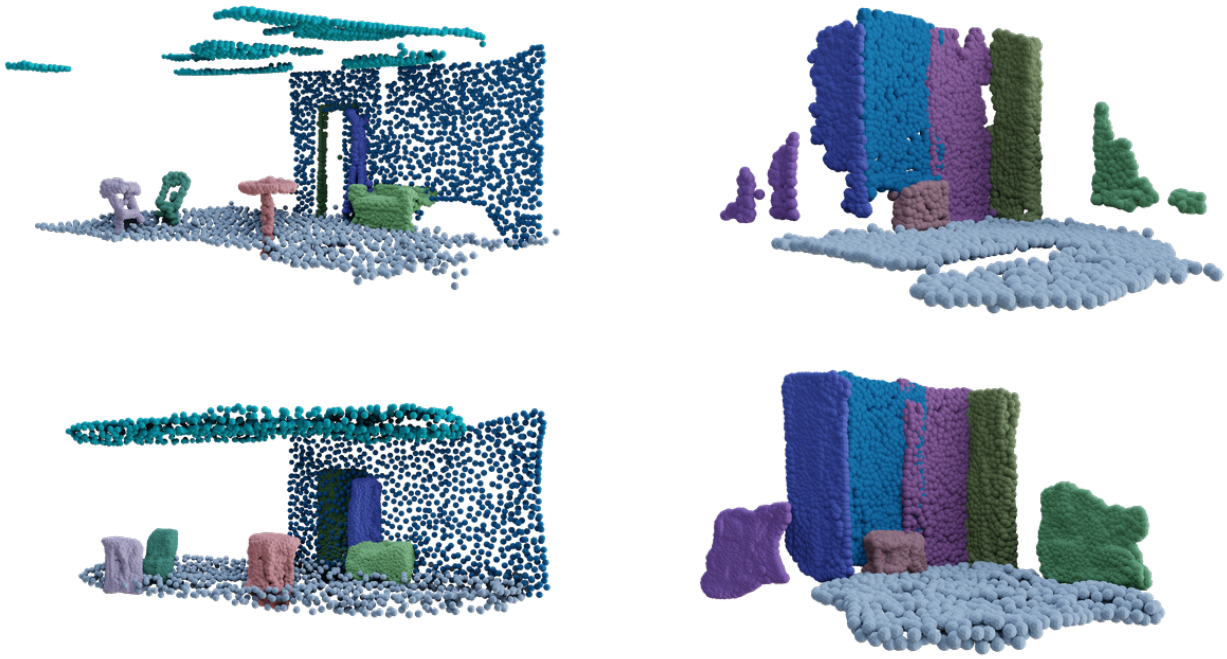
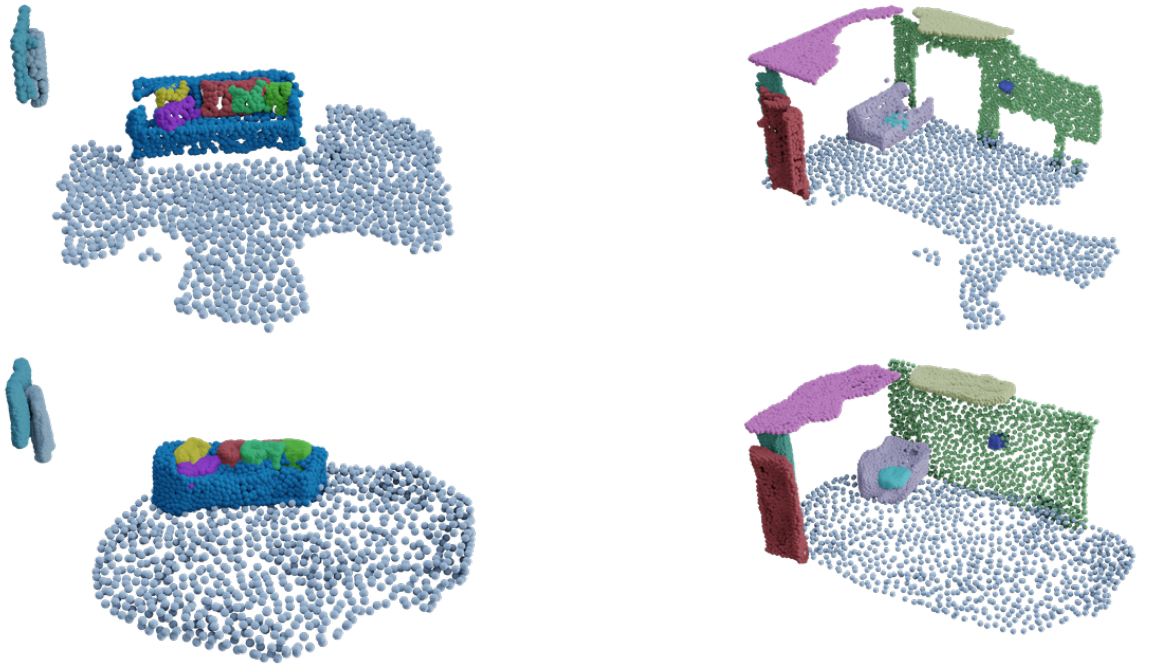


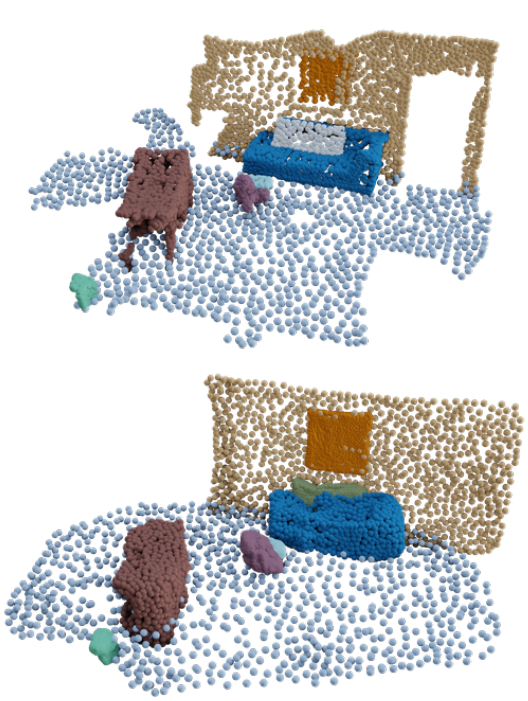
Figure 2. **Qualitative scene graph results from SGRec3D.** Top: 3D scene; Bottom: Predicted 3D scene graph.



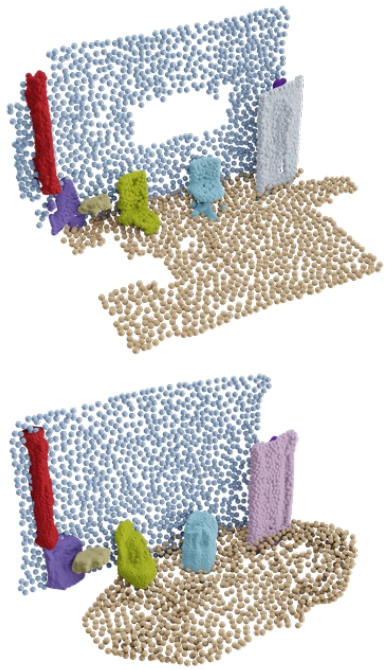
(a) (b)



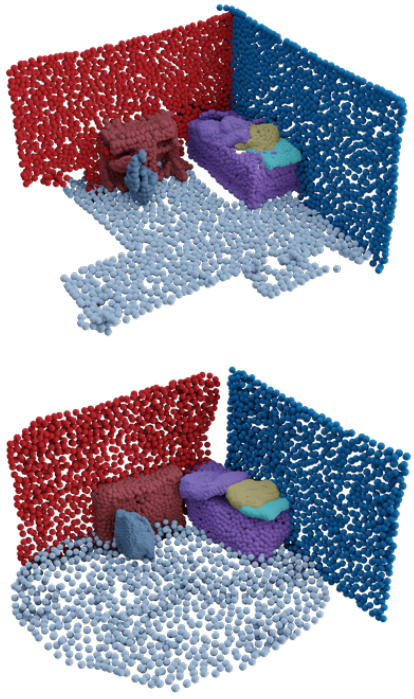
(c) (d)



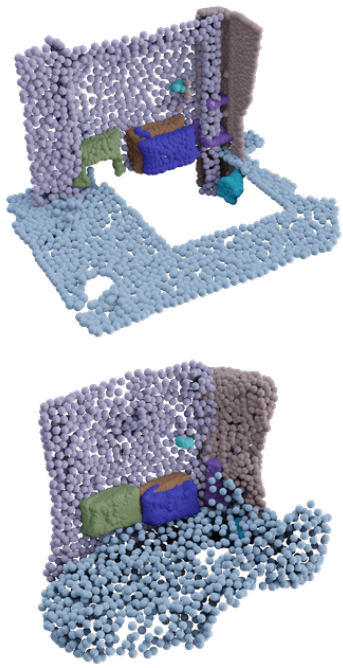
(e)



(f)



(g)



(h)

Figure 3. **Qualitative scene generation results from SGRec3D.** Top: Original 3D scene; Bottom: Reconstructed scene using SGRec3D.