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

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

Abstract

*In the field of 3D scene understanding, 3D scene graphs have emerged as a new scene representation that combines geometric and semantic information about objects and their relationships. However, learning semantic 3D scene graphs in a fully supervised manner is inherently difficult as it requires not only object-level annotations but also relationship labels. While pre-training approaches have helped to boost the performance of many methods in various fields, pre-training for 3D scene graph prediction has received little attention. Furthermore, we find in this paper that classical contrastive point cloud-based pre-training approaches are ineffective for 3D scene graph learning. To this end, we present SGRec3D, a novel self-supervised pre-training method for 3D scene graph prediction. We propose to reconstruct the 3D input scene from a graph bottleneck as a pretext task. Pre-training SGRec3D does not require object relationship labels, making it possible to exploit large-scale 3D scene understanding datasets, which were off-limits for 3D scene graph learning before. Our experiments demonstrate that in contrast to recent point cloud-based pre-training approaches, our proposed pre-training improves the 3D scene graph prediction considerably, which results in SOTA performance, outperforming other 3D scene graph models by **+10%** on object prediction and **+4%** on relationship prediction. Additionally, we show that only using a small subset of 10% labeled data during fine-tuning is sufficient to outperform the same model without pre-training.*

1. Introduction

Scene graphs provide a graph-based representation of a scene, by not only representing the geometric scene objects, but also their relation among each other. In recent years, 2D scene graphs have seen a wide range of applications in computer vision and robotics [4, 5, 16, 26]. Consequently, many approaches for generating 2D scene graphs based on given input images have been proposed [47, 53, 62]. In the same way as 2D scene graphs capture structured knowledge about scenes represented through images, 3D scene graphs can capture the same information for point clouds or other 3D data structures. Despite the fact that 3D scene graphs are widely used in computer graphics [19], and despite their great potential for solving computer vision or robotics tasks [1, 25, 43, 55], relatively little work has been done to predict 3D scene graphs based on a given 3D scene.

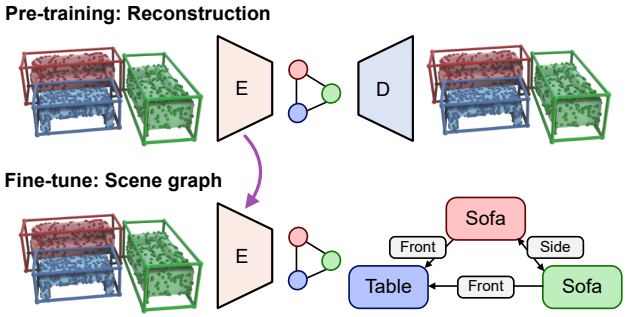


Figure 1. **SGRec3D Overview.** SGRec3D exploits autoencoder-based pre-training to build a 3D scene graph latent space through a reconstruction loss. The resulting encoder can, later on, be fine-tuned for the downstream 3D scene graph prediction.

Predicting 3D scene graphs comes with several challenges on its own. It does not only have to provide a high level representation of a given 3D scene, but it must also derive this from often noisy and incomplete sensor data. Thus, 3D scene graph generation is difficult to tackle with rule-based deterministic algorithms. For instance, two chairs of the *same style* could have different visual appearances or a jacket *lying on* one chair could occlude most of the visible surface. While first approaches have been proposed to learn a 3D scene graph based on a 3D point cloud [49, 50, 64, 65], these approaches require labels to be available, as they perform scene graph prediction in a fully supervised manner. Acquiring data and labels in 3D is very challenging task and

requires extensive human effort, which is underlined by the particular scarcity of labeled training data in the domain of 3D scene graphs. Therefore, our goal is to reduce the need for such labels when learning to predict 3D scene graphs.

In recent years, self-supervised pre-training methods in 3D have shown to be effective in improving results of neural architectures with a high data demand by utilizing the available data more efficiently without requiring additional annotated data [10, 21, 23, 57]. Despite the promising nature of pre-training approaches in low data regimes, self-supervised pre-training for 3D scene graphs has not been investigated so far. In particular, we find in this paper that point cloud-based pre-training approaches are ineffective for 3D scene graphs (see Sec. 4). To solve this issue, we propose a new self-supervised pre-training approach tailored for 3D scene graphs using an encoder-decoder-based method with a graph bottleneck and a graph-based pretext task. We choose graph-based 3D reconstruction as our pretext task which, unlike previous point cloud-based pre-training approaches, considers the graph information directly to learn the optimal information flow through the graph to reconstruct 3D scene point clouds. In contrast to 2D, 3D scene reconstruction remains a challenging problem, mainly due to the sparsity and non-continuous nature of 3D point clouds.

Thus, our main contributions are: **(i)** We propose a novel self-supervised pre-training method designed for 3D scene graph predictions. To the best of our knowledge, this is the first pre-training approach designed for 3D scene graphs. **(ii)** We demonstrate how to utilize additional 3D datasets to boost the effectiveness of our pre-training approach without being dependent on scene graph labels. **(iii)** We outperform fully-supervised methods and our novel pre-training shows greater effectiveness than other point cloud-based pre-training baselines. **(iv)** Our pre-trained method demonstrates significantly improved label efficiency by requiring only 5%-10% of scene graph labels to outperform the same model trained from scratch on a complete labeled dataset.

2. Related Work

Scene graph prediction. A scene graph is a data structure that represents a scene as a graph, where nodes provide semantic descriptions of objects in the scene and edges represent relationships between objects. In computer vision, scene graphs were first introduced by Johnson et al. [27] motivated by image retrieval. Subsequent works focused primarily on the refinement of scene graph prediction from images [22, 32, 33, 61, 63], while utilizing different methods such as message passing [58], GCN [28] or attention [42]. Some works also investigate the incorporation of prior knowledge into the graph learning problem [9, 46]. Much of the progress is accounted to the introduction of visual gnome [29], a large scale dataset for connecting language and vision which contains scene graph annotations for images. Chang et al. [8] provide a comprehensive survey of scene graph generation approaches and their applications. Other works, instead, apply semantic scene graphs to image generation [4, 26], and image manipulation [16].

Applications of scene graphs can be also found in the 3D domain where literature presents two main approaches which explore their potential. Wald et al. [49] introduce the first 3D scene graph dataset 3DSSG, with focus on semantics with 3D graph annotations, build upon the 3RScan dataset [48]. Based on this dataset, subsequent works extended the common principles of 2D scene graph prediction to 3D [49, 50]. Other works explore unique approaches for 3D scene graphs utilizing novel graph neural networks [64], transformers [37], the use of prior knowledge [65], or image-based oracle models [52]. Others explore applications utilizing 3D scene graphs for 3D scene generation and manipulation [17], the alignment of 3D scans with the help of 3D scene graphs [44], or dynamic construction of 3D scene graphs [54, 55] during the exploration of a 3D scene. In contrast, our approach focuses on a novel pre-training strategy for scene graph prediction, without requiring additional scene graph labels.

Pre-training for 3D scene understanding. Deep learning methods are well known for requiring large amounts of training data. Since collecting data and providing labels is costly and time-consuming, pre-training methods have emerged in the field of scene understanding. In the 2D domain, for instance, pre-training on existing large-scale datasets, such as ImageNet [15], is a common practice. More recently, methods such as masked autoencoders [21] have demonstrated, that pre-training alone on the target dataset using a pretext task can improve results by a considerable margin. In 3D, representation learning approaches demonstrate that using only a fraction of available point labels can lead to similar results as obtainable with fully supervised methods when pre-trained with a self-supervised pretext task [10, 23, 24, 57, 66]. However, so far neither of these works have considered 3D scene graph prediction as the downstream task. In this work, we will compare existing pre-training methods designed for point cloud pre-training with our approach designed for 3D scene graph learning.

3D scene reconstruction. Literature shows a number of methods able to generate 3D scenes from images [18, 39, 64]. Other works aim to complete a 3D scene from an incomplete 3D scan [13, 14, 59]. But only few works attempt to do full 3D scene reconstruction from point clouds [40], and most methods for 3D reconstruction are limited to object reconstructions [6, 20, 36, 60] on datasets like ShapeNet [7] or ModelNet [56]. Methods more similar to our approach explore 3D scene generation from graphs [36, 51], however most methods simplify the task of 3D generation. Li et al. [30] for instance introduce

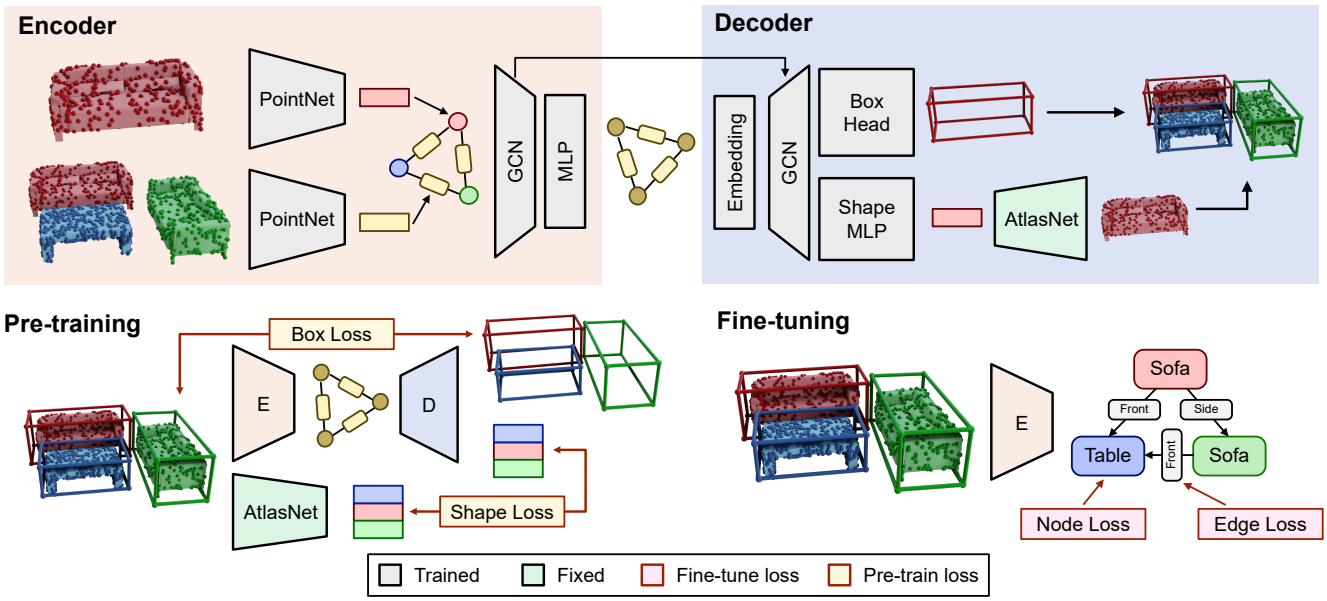


Figure 2. **SGRec3D architecture overview.** SGRec3D utilizes an encoder-decoder structure for pre-training (bottom-left) using a reconstruction loss by reconstructing the bounding boxes and shape encodings of objects with supervision from a pre-trained AtlasNet. The encoder (top-left) generates object and edge embeddings from the input point cloud into a latent graph. The decoder (top-right) reconstructs the input point cloud from the graph bottleneck. During fine-tuning (bottom-right), the decoder is discarded and the encoder is fine-tuned to predict the node and edge classes.

GRAINS, a recursive VAE to generate a 3D layout followed by object retrieval to synthesize a 3D indoor scene. Dhamo et al. [17] go beyond object retrieval and attempt to generate and manipulate 3D scenes by reconstructing objects individually from a scene graph using a generative graph-based model. Similar to this work, we design our decoder to reconstruct the 3D scene from a graph bottleneck. However, in contrast to this work, we reconstruct the input scene, instead of generating plausible object shapes and layouts.

3. Method

We propose SGRec3D, a novel pre-training method to learn 3D scene graphs from 3D data in an autoencoder-like manner, as shown in Fig. 2. Like all autoencoding approaches, our method consists of an encoder that maps the input to a latent representation and a decoder that reconstructs the original input from the latent representation. But unlike most autoencoder approaches, our method maintains a graph representation within the network given a non-graph input and output. The encoder (see Sec. 3.1), takes as input a point cloud partitioned using object instances and their bounding boxes. From this input, the encoder generates a minimal representation as a 3D scene graph in a graph bottleneck (see Sec. 3.2), by learning to reconstruct from this representation the input scene using a decoder (see Sec. 3.3). This pre-trained architecture can then be fine-tuned to predict a semantic 3D scene graph $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$, where nodes $\mathcal{N}$ represent object instances within a corresponding 3D point cloud, while edges $\mathcal{E}$ express predicates that form together with object nodes semantic relationships (see Sec. 3.4). Each edge in the graph can represent zero or more relationships.

3.1. Encoder

Given a point cloud $\mathcal{P}$ of a scene $\mathcal{S}$ with class-agnostic instance segmentation $\mathcal{M}$ provided by an off-the-shelf instance segmentation method such as Mask3D [45] or a dataset, we extract each point set $\mathcal{P}_i$ containing instance i and its axis-aligned or oriented bounding box $\mathcal{B}_i$ using the mask $\mathcal{M}_i$. Moreover, for every instance pair $\langle i, j \rangle \in \|\mathcal{M}\| \times \|\mathcal{M}\|$, we get the point set $\mathcal{P}_{ij}$ using the union of their respective bounding boxes $\mathcal{B}_{ij} = \mathcal{B}_i \cup \mathcal{B}_j$. Note that the point set $\mathcal{P}_{ij}$ contains not only the union of the masked instances $\mathcal{P}_i \cup \mathcal{P}_j$, but also other points falling within the volume $\mathcal{B}_{ij}$. This helps to augment the point cloud with contextual information relating the two objects. $\mathcal{P}_i$, $\mathcal{B}_i$ and $\mathcal{P}_{ij}$ serve as input to our scene encoder. The encoder follows the common principles of scene graph prediction from prior 2D and 3D works [35, 49, 58, 61]. We construct an initial graph with node features ϕ_n and edge features ϕ_p from the extracted instance and bounding box features. Each point set $\mathcal{P}_i$ is fed into a shared PointNet [41] to extract features for object nodes. Every point set $\mathcal{P}_{ij}$ is concatenated with a mask which is equal to 1 if the point corresponds to object i, 2 if the object corresponds to object j, and 0 otherwise. The concatenated feature vector is

then fed into another shared PointNet to extract features for predicate edges. Additionally, the centers of the point sets $\mathcal{P}_i$ and $\mathcal{P}_{ij}$ are normalized before inputting them into the respective PointNet.

The extracted node and edge features are then arranged as triplets $t_{ij} = \langle \phi_{n,i}, \phi_{p,ij}, \phi_{n,j} \rangle$ in a graph structure. This initial feature graph is passed into a GCN [28]. Each GCN layer l_g processes the triplets t_{ij} and propagates the information through the graph in three steps, with a similar message passing procedure to [49]. First, t_{ij} is fed into a MLP $g_1(\cdot)$

$$(\psi_{n,i}^{(l_g)}, \phi_{p,ij}^{(l_g+1)}, \psi_{n,j}^{(l_g)}) = g_1\left(\phi_{n,i}^{(l_g)}, \phi_{p,ij}^{(l_g)}, \phi_{n,j}^{(l_g)}\right) \quad (1)$$

where ψ represents the nodes' processed features. After this first pass, the resulting edge feature $\phi_{p,ij}^{(l_g+1)}$ does not need any further refinement.
Second, an aggregation function averages the incoming information from all the connected edges of each node

$$\rho_{n,i}^{(l_g)} = \frac{1}{N_i}\left(\sum_{k\in\mathcal{R}_i} \psi_{n,k}^{(l_g)} + \sum_{k\in\mathcal{R}_j} \psi_{n,k}^{(l_g)}\right) \quad (2)$$

where N_i denotes the number of edges connected to node i, and $\mathcal{R}_i$ and $\mathcal{R}_j$ are the set of nodes connected to node i and node j respectively.
Finally, the resulting node feature $\rho_{n,i}^{(l_g)}$ passes into a second update MLP $g_2(\cdot)$ and a residual connection is added:

$$\phi_{n,i}^{(l_g+1)} = \phi_{n,i}^{(l_g)} + g_2(\rho_{n,i}^{(l_g)}). \quad (3)$$

In the end, the processed features $\phi_{n,i}^{(l_g+1)}, \phi_{p,ij}^{(l_g+1)}, \phi_{n,j}^{(l_g+1)}$ are passed to the next layer of the network.

3.2. Graph bottleneck

Features are further processed through multiple layers of graph convolutions, propagating them to neighboring nodes. A final MLP $f_n(\cdot)$ is applied to all node features, and a $softmax$ activation function represents the nodes as a probability distribution over the node classes

$$\phi_{n,i}^{(e)} = softmax(f_n(\phi_{n,i}^{(l_n)})). \quad (4)$$

where $\phi_{n,i}^{(e)}$ is the final encoder feature vector for each node which is passed into the decoder.

The edge features, instead, are handled by a different MLP $f_p(\cdot)$ and by a class-wise sigmoid activation function to map the edges to a separate probability distribution for each possible relationship between object nodes

$$\phi_{p,i}^{(e)} = \sigma(f_p(\phi_{p,i}^{(l_n)})). \quad (5)$$

where $\phi_{p,i}^{(e)}$ is the final encoder feature vector for each edge which is passed into the decoder.

3.3. Decoder

The goal of our scene decoder is to reconstruct the original scene from the bottleneck scene graph representation. To preserve the layout and object details, we first pass the low-dimensional features into an embedding MLP which lifts the latent graph representation to a high-dimensional feature-space. Then, we further decode the latent graph using another GCN with the same message passing structure as the encoder. Due to the low-dimensionality of the bottleneck and the ambiguity of the scene graph, the decoding step may be affected by information loss. Thus, we address this problem by introducing an additional skip-connection between the last GCN encoder layer before applying the softmax and sigmoid functions and the first GCN decoder layer by concatenating ($\oplus$) the GCN features with the embedded feature from the bottleneck. For the node and edge features this is defined as follows

$$\phi_{n,i}^{(d_{in})} = (\phi_{n,i}^{(e)} \oplus \phi_{n,i}^{(l_n)}), \quad \phi_{p,i}^{(d_{in})} = (\phi_{p,i}^{(e)} \oplus \phi_{p,i}^{(l_n)}). \quad (6)$$

where $\phi_{n,i}^{(d_{in})}/\phi_{p,i}^{(d_{in})}$ are the input decoder features for each node and edge.

Reconstructing a full 3D scene is a highly complex task, giving the sparsity of 3D data. Therefore, we choose to reconstruct each object individually rather than the complete scene. To this end, we combine the 3D bounding box of each object, predicted by the Box-Head, with the corresponding object reconstruction provided by the Shape-Head. For the final scene reconstruction, we place each generated object within its matching bounding box.

For the Box-Head, we implement an MLP to predict the box extents $[h, w, d]$ and the center location $[c_x, c_y, c_z]$. Predicting the 3D orientation of objects using regression has shown to be difficult given the non-linearity of the 3D rotation space [38]. Therefore, we predict the object's orientation angle α separately by means of classifying it into 1 out of 24 discrete bins, rather than regressing the angle directly. The Shape-Head consists of an MLP that predicts a 1D latent vector which is further processed by the decoder of AtlasNet [20] pre-trained on ShapeNet [7], which reconstructs the object from the latent vector.

3.4. Pre-training using scene reconstruction

For pre-training we learn to reconstruct the 3D scene by predicting the bounding box and the shape of the objects. The loss for the object-level scene reconstruction is composed of three components: **(i)** a bounding box regression loss $\mathcal{L}_{\text{bbox}}$ which uses the L_1 distance for the bounding box parameters, **(ii)** a cross entropy classification loss $\mathcal{L}_{\text{angle}}$, and **(iii)** an L_1 loss $\mathcal{L}_{\text{shape}}$ for the shape embedding before applying the AtlasNet decoder:

$$\mathcal{L}_{\text{rec}} = \eta_1 \mathcal{L}_{\text{bbox}} + \eta_2 \mathcal{L}_{\text{angle}} + \eta_3 \mathcal{L}_{\text{shape}} \quad (7)$$

where η_i are weighting factors. Note that this loss does not rely on scene graph labels which allows for the use of additional training data from larger 3D data sets, as we will demonstrate in Section 4.

After pre-training, our model needs to be fine-tuned on the downstream task of predicting 3D scene graphs. For this, we discard the decoder and fine-tune the pre-trained encoder using the scene graph annotations with a fully supervised loss $\mathcal{L}_{\text{SG}}$. It consists of a cross-entropy loss $\mathcal{L}_{obj}$ for the node classification and a per-class binary cross entropy loss $\mathcal{L}_{pred}$ for the predicate prediction. The latter is used to learn different predicates separately from one another to support multi-predicate relationships. The combined loss is defined as

$$\mathcal{L}_{\text{SG}} = \lambda_1 \mathcal{L}_{\text{obj}} + \lambda_2 \mathcal{L}_{\text{pred}} \tag{8}$$

where λ_1 and λ_2 are the respective weighting factors.

Further details and documentation of our model architecture, training procedure, chosen loss functions and weighting factors are provided in the supplementary material.

4. Experiments

4.1. Experimental setup

Datasets. To prove the effectiveness of our proposed method, we evaluate it on real-world 3D scans from the 3DSSG [49] dataset. 3DSSG is currently the only real-world dataset that provides semantic 3D scene graph annotations. Another 3D scene graph dataset is [2], however, the scene graphs modeled in this dataset focus on hierarchical structuring and lack semantic relationship labels. In contrast, 3DSSG provides 3D scene graph labels for 160 distinct object classes and 27 relationship categories, corresponding to over 1,000 3D indoor point cloud reconstructions. The 3D scene graphs present in 3DSSG are further split into smaller sub-graphs spanning a small selection of objects per scene, yielding over 4,000 samples for training and evaluation. We follow the previous work [49] and use the same scene graph and training/evaluation splits first introduced by Wald et al. [48]. The 3DSSG dataset, however, is a rather small dataset, including only 478 different scenes, which may be challenging for training large deep learning architectures. To alleviate this problem, we additionally pre-train on existing indoor object detection datasets ScanNet [12] and S3DIS [3]. ScanNet and S3DIS are much larger indoor datasets, including 1513 and 727 annotated scenes respectively. The available baselines cannot use additional datasets since they require ground truth scene graph annotations. In contrast, our pretext task does not require these annotations, which makes it possible for us to utilize additional datasets for pre-training.

Evaluation metrics. Following previous works [49, 50, 58, 61, 64], we evaluate object node classification and predicate edge prediction separately. To analyze the overall scene graph prediction performance, we jointly compute the accuracy of relationships consisting of triples formed by two nodes (subject & object) and their connecting edge (predicate). Since we predict object nodes and predicate edges independently, we adapt the approach first introduced by Yang et al. [61] for relationship evaluation. Through this method, we obtain a scored list of triplet predictions by multiplying the object node confidences with the predicate edge probability. For comparison with previous works, we follow [49, 50, 64] and use the top-k recall metric first introduced by Lu et al. [35] for scene graph prediction. For objects and predicates, we further do a class-wise evaluation by splitting the classes based on the frequency of number of labels in the train set into head, body and tail respectively. A class-wise evaluation over all categories as well as for the head, body, and tail splits enables a more precise understanding of the prediction performance. For this we use the same top-k metric formulation, which is also known as the more precise but less commonly used mR@k metric.

	Object		Predicate		Relationship	
Method	R@5	R@10	R@3	R@5	R@50	R@100
3D + MSDN [33]	0.61	0.72	0.86	0.94	0.47	0.53
3D + KERN [9]	0.67	0.77	0.83	0.96	0.51	0.58
3D + BGNN [31]	0.71	0.82	0.87	0.94	0.55	0.60
SGGPoint [64]	0.28	0.36	0.68	0.87	0.08	0.10
3DSSG [49]	0.68	0.78	0.89	0.93	0.40	0.66
Liu et al. [34]	0.74	0.83	0.90	0.96	0.62	0.68
SGFN [55]	0.70	0.80	**0.97**	**0.99**	0.85	0.87
Ours	**0.80**	**0.87**	**0.97**	**0.99**	**0.89**	**0.91**

Table 1. **3D scene graph prediction on 3DSSG.** Experimental results for 3D scene graph prediction on 3DSSG. We report the top-k recall values for object classification, predicate prediction as well as relationship prediction. For a fair comparison, all works use ground-truth class-agnostic instance segmentation.

4.2. 3D scene graph prediction

Comparison with fully supervised methods. To show the impact of our pre-training, we compare it against recent fully supervised 3D scene graph baselines (SGGPoint [64], 3DSSG [49], SGFN [55] and Liu et al. [34]) and adopted 2D scene graph methods (MSDN [33], KERN [9] and BGNN [31]). For the 2D scene graph methods, the 2D object detector was replaced by a PointNet-based feature extractor. We note that, to alleviate the severe object class imbalance in the scene graph prediction task, SGGPoint only provides a model for 27 object classes and 16 relation classes in the 3DSSG dataset.

Results in Tab. 1 show that SGRec3D outperforms most existing fully supervised methods by a large margin, and our closest competitor SGFN [55] by a considerable amount. Especially on object node classification SGRec3D outper-

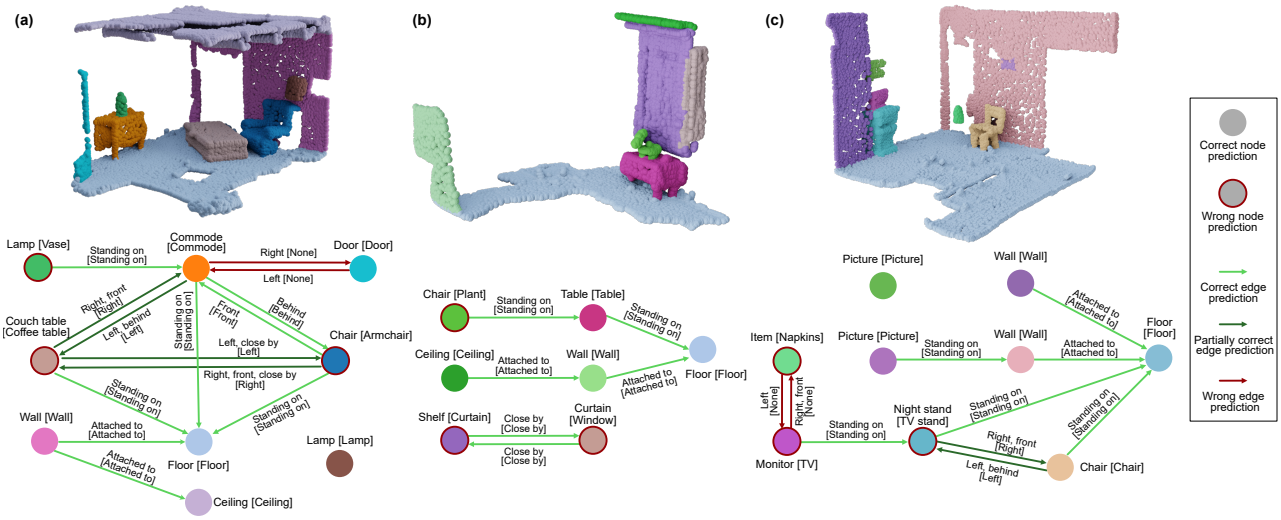


Figure 3. **3D scene graph visualizations for 3DSSG scene splits.** We visualize the top-1 object class prediction for each node and the predicates with a probability greater than 0.5 for each edge. Ground truth labels are shown in square brackets.

		Head	Body	Tail	All
Objects	w/o pre-train	0.88	0.45	0.06	0.30
	w/ pre-train	0.92	0.78	0.24	0.45
Predicates	w/o pre-train	0.94	0.83	0.41	0.57
	w/ pre-train	0.97	0.96	0.65	0.69

Table 2. **Frequency based class evaluation.** We sort objects and predicates into head, body, tail classes based on their occurrence and compare our pre-training method with the same model not pre-trained. We compare on the mR@5 metric objects and the mR@3 metric for predicates.

forms SGFN by a large margin (+10%/+7%). For relationship prediction, we also report favorable results with a +4% increase to SGFN on both metrics. For the predicate edge prediction, we observe similar results as SGFN. Given the overall high score for this metric, we assume that we reached a saturation point for this task on this dataset. In Fig. 3, we provide predicted 3D scene graphs for three different scenes. Our method is able to predict accurate and mostly correct scene graphs for the given scenes. Objects are predicted well, with most nodes being predicted correctly and only some nodes being predicted incorrectly, where our method often chooses an object class of a similar meaning. Similarly, predicates between objects are also predicted with a high accuracy with only a few false positive predictions.

Class-wise evaluation. To further investigate the impact of our proposed pre-training, in Tab. 2 we provide a detailed comparison of our method with and without pre-training for individual classes grouped into head, body and tail based on their frequency. Additionally, we provide *All* which is the average recall over all classes individually also known as the mR@k metric. The improvement of our pre-training

		pre-train		Object		Predicate	
	GCN	PCL	SG	R@5	mR@5	R@3	mR@3
STRL [11]		✓		0.75	0.35	0.94	0.50
STRL [11]	✓	✓		0.63	0.23	0.92	0.48
DepthContrast [66]		✓		0.77	0.36	0.94	0.51
DepthContrast [66]	✓	✓		0.60	0.22	0.93	0.50
Ours (no pre-train)	✓			0.63	0.30	0.94	0.57
Ours (no GCN)			✓	0.75	0.31	0.94	0.48
Ours	✓		✓	**0.80**	**0.45**	**0.97**	**0.69**

Table 3. **Pre-training comparison.** We compare SGRec3D with existing recent point cloud-based pre-training approaches (PCL). Our novel scene graph pre-training (SG) shows high effectiveness outperforming point cloud-based pre-training approaches. Adding a GCN to our method is beneficial since we optimize it during pre-training, while point cloud-based pre-training approaches do not benefit from a GCN which is not pre-trained.

over all classes is large with a +27% gain for object classification and a +24% gain for predicate prediction on the mR@k metric. We observe, that this improvement originates mostly from a very large improvement on rare body and tail classes. The improvement on more frequent head classes is smaller since the baseline method already produces good results for frequent categories.

Comparison with point cloud-based pre-training. We are the first to investigate pre-training designed for 3D scene graphs by considering the graph nature of scene graphs during pre-training. In Tab. 3, we show a comparison with recent point cloud-based pre-training approaches. In contrast to our method, these approaches do not model the graph structure of the scene graph during pre-training. We compare with the pre-trained 3D feature encoders from both STRL [11] and DepthContrast [66]. We choose these two

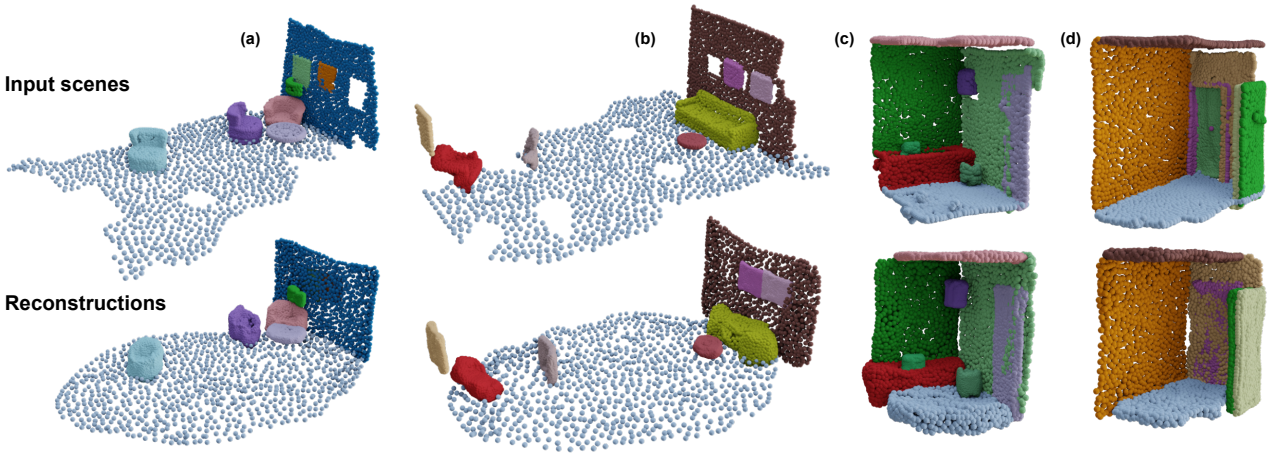


Figure 4. **Scene reconstruction on 3DSSG.** Qualitative results of the reconstructions for four different input scenes. While the reconstruction does not represent the input scenes perfectly, it is faithful to the input scenes, as object locations are well-preserved. Furthermore, the shapes look similar and the reconstruction quality is stable.

approaches because they have been proven to be highly effective in pre-training 3D scene understanding models for tasks such as 3D segmentation and detection. Similar to our method, they rely on a PointNet++ feature extraction backbone and ScanNet as pre-training data. We add two prediction heads on top of their pre-trained PointNet++ backbones for objects and predicates and fine-tune them on the 3DSSG dataset. For further comparisons, we add a GCN between the pre-trained backbone and the prediction heads to make their network architecture very similar to ours. The major difference is that while our GCN contains pre-trained weights, their GCN is randomly initialized because only the 3D feature extractors can be pre-trained by STRL and DepthContrast. Tab. 3 demonstrates, that our scene graph pre-training (SG) tailored for 3D scene graph prediction produces drastically better results than our point cloud-based pre-training baselines (PCL), with 9% improvement on the mR@5 metric for objects and 18% on the mR@3 metric for predicates. We observe that our novelty of a pre-trained graph neural network greatly improves the pre-training effectiveness of our method (+14% object, +11% predicate prediction), the same is not true for the point cloud-based pre-training methods. We assume this is because, in contrast to our method where the graph layers are optimized during pre-training, the point cloud-based approaches do not optimize the graph. Adding the graph layers during fine-tuning consequently adds a considerable number of untrained weights which are challenging to train on a rather small dataset such as 3DSSG.

4.2.1 Scene reconstruction

The benefit of our pre-training is influenced by the ability of our model to learn the reconstruction pretext task. Since our downstream task includes learning relationships in scene graphs, it is crucial that the relationships present in the original scene remain preserved in the reconstructed scene. This indicates that the model learns transferable knowledge for the downstream scene graph prediction during pre-training.

Relationship	Rule	GraphTo3D	Ours
left of	$c_{x,i} < c_{x,j}$ and $iou(b_i, b_j) < 0.5$	0.85	0.92
right of	$c_{x,i} > c_{x,j}$ and $iou(b_i, b_j) < 0.5$	0.85	0.92
front of	$c_{y,i} < c_{y,j}$ and $iou(b_i, b_j) < 0.5$	0.79	0.90
behind of	$c_{y,i} > c_{y,j}$ and $iou(b_i, b_j) < 0.5$	0.79	0.90
higher than	$h_i + c_{z,i}/2 > h_j + c_{z,j}/2$	0.96	0.96
lower than	$h_i + c_{z,i}/2 < h_j + c_{z,j}/2$	0.96	0.96
smaller than	$w_i l_i h_i < w_j l_j h_j$	0.98	0.96
bigger than	$w_i l_i h_i > w_j l_j h_j$	0.98	0.96
same as	$iou(b_i, b_j) > 0.5$	1.00	1.00
average		0.90	0.94

Table 4. **Rule-based scene generation verification.** A simple rule-based verification of the scene generations from SGRec3D, for a subset of the predicates in the 3DSSG dataset.

Fig. 4 shows some qualitative results for reconstructing 3D scene splits from 3DSSG. In general, our model correctly reconstructs the layouts of the scenes. In all predicted scenes, the reconstructed objects are located in similar positions to the ones in the original scene. Relationships that describe the relative proximity of objects are clearly preserved, such as *Close by, Left, Right, etc.* Sometimes objects hanging on walls are generated on the wrong side of the wall (see Fig. 3a), however relationships like *hanging on, attached to* are still maintained in the generation. The shape of the objects differs in detail compared to the original scenes. This is because we do not fine-tune the AtlasNet [20] decoder for more stable training. Still, the rough shape of the objects is preserved and relationships like *same as, bigger than, smaller than* are maintained.

In Tab. 4, we provide a quantitative evaluation of the

Dataset	Object		Predicate		Relationship	
	R@5	R@10	R@3	R@5	R@50	R@100
3DSSG	0.75	0.83	0.96	**0.99**	0.88	0.89
S3DIS	0.76	0.85	0.96	**0.99**	**0.89**	0.90
ScanNet	0.77	0.85	0.96	**0.99**	**0.89**	0.90
S3DIS+ScanNet	0.79	0.86	0.96	**0.99**	**0.89**	**0.91**
S3DIS+ScanNet+3DSSG	**0.80**	**0.87**	**0.97**	**0.99**	**0.89**	**0.91**

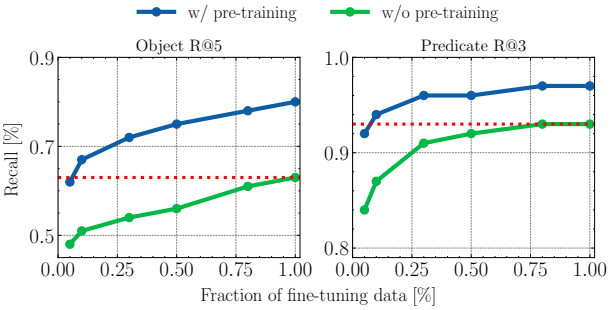
Table 5. **Pre-training dataset.** A comparison of SGRec3D pre-trained on different datasets. The datasets are ordered by size.

preserved relationships for those predicates where a simple rule can be approximated. We compare our results with the results from Graph-to-3D [17]. The reported results confirm that our method preserves the original and reconstructed relationships in the scene. With an overall accuracy of 94% the pretext task has been learned well by the model indicating an effective pre-training task. As shown in the table, we outperform Graph-to-3D, with great improvements for *front of/behind of* relationships. While Graph-to-3D generates a scene solely using a scene graph as input, we reconstruct the scene from 3D using the graph bottleneck. This retains more context information of the original scene compared to Graph-to-3D which tries to generate a novel scene.

4.3. Ablations

Pre-training dataset. Our method allows to leverage large-scale 3D datasets without scene graph labels during pre-training. In Tab. 5 we investigate the role of a larger pre-training dataset by reporting the fine-tuned performance of our method, given different pre-training datasets. We observe that pre-training on the 3DSSG [49] dataset only, which is also used for fine-tuning, leads to competitive results compared to existing methods from Tab. 1. Pre-training on larger datasets like ScanNet [12] and S3DIS [3] further improves fine-tuning results. Finally, increasing pre-training data by combining individual datasets gives the best fine-tuned model. We can highlight considerable improvements in the object classification scores. The predicate scores improve only slightly, which we assume is correlated to a saturated metric. The relationship scores improve marginally with increasing dataset size, achieving best results by combining 3DSSG, ScanNet and S3DIS.

Limited fine-tuning data. The goal of our proposed pre-training is to reduce the need for labeled scene graph data, which is often hard to annotate. To prove this contribution, in Tab. 6b we provide an ablation of our pre-trained model fine-tuned on a fraction of labeled data from 3DSSG [49]. We observe that reducing the number of labeled samples during fine-tuning only affects marginally the performance on object, predicate, and relationship prediction. For example, using only a small fraction of the labeled data (~10%-30%) results in competitive performance compared to the works from Tab. 1. Moreover, we would like to emphasize that even with the 5% of labeled data – around 200 training samples – SGRec3D is able to achieve acceptable results. In Tab. 6a, we compare the effects of limited labeled training data for our pre-trained model against the same model trained from scratch without pre-training. The pre-trained model outperforms the model trained from scratch on all data quantities by a large margin for object classification and predicate prediction. Furthermore, the pre-trained model requires only 5%-10% of labeled data to outperform the model trained from scratch.



(a) Comparison of SGRec3D with and without pre-training under limited labeled fine-tuning data.

Method	Object		Predicate		Relationship	
	R@5	R@10	R@3	R@5	R@50	R@100
$\text{SGRec3D}_{100\%}$	**0.80**	**0.87**	**0.97**	**0.99**	**0.89**	**0.91**
$\text{SGRec3D}_{80\%}$	0.78	0.86	0.94	0.98	0.89	0.90
$\text{SGRec3D}_{50\%}$	0.75	0.83	0.93	0.97	0.88	0.89
$\text{SGRec3D}_{30\%}$	0.72	0.82	0.92	0.97	0.87	0.88
$\text{SGRec3D}_{10\%}$	0.67	0.75	0.90	0.95	0.84	0.85
$\text{SGRec3D}_{05\%}$	0.62	0.71	0.89	0.93	0.81	0.83

(b) Affects of limited labeled fine-tuning data for SGRec3D.

Table 6. **Limited fine-tuning data.** A comparison of SGRec3D fine-tuned using different quantities of 3DSSG labels.

5. Conclusion

Pre-training for scene graphs has received little attention so far, despite its success for a variety of other downstream tasks. In this paper we find that existing point cloud-based pre-training approaches are ineffective for 3D scene graph prediction. To this end, we present SGRec3D, a novel self-supervised pre-training method for the downstream task of 3D scene graph prediction. To the best of our knowledge, this is the first approach addressing pre-training for 3D scene graph prediction. Our experiments show that SGRec3D significantly improves the predictions of objects and relationships in 3D scene graphs compared to existing fully supervised methods. We achieve these results thanks to our pre-training contribution and the use of additional 3D datasets for pre-training.We show that even using a small percentage of limited fine-tuning data, SGRec3D produces competitive results with recent methods.

References

[1] Christopher Agia, Krishna Murthy Jatavallabhula, Mohamed Khodeir, Ondrej Miksik, Vibhav Vineet, Mustafa Mukadam, Liam Paull, and Florian Shkurti. Taskography: Evaluating robot task planning over large 3d scene graphs. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 46–58. PMLR, 08–11 Nov 2022. 1

[2] Iro Armeni, Zhi-Yang He, JunYoung Gwak, Amir R. Zamir, Martin Fischer, Jitendra Malik, and Silvio Savarese. 3d scene graph: A structure for unified semantics, 3d space, and camera. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 5

[3] Iro Armeni, Sasha Sax, Amir R. Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *CoRR*, abs/1702.01105, 2017. 5, 8

[4] Oron Ashual and Lior Wolf. Specifying object attributes and relations in interactive scene generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 1, 2

[5] Andrzej Bielecki and Piotr Śmigielski. Graph representation for two-dimensional scene understanding by the cognitive vision module. *International Journal of Advanced Robotic Systems*, 14(1):1729881416682694, 2016. 1

[6] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. Learning gradient fields for shape generation. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 364–381, Cham, 2020. Springer International Publishing. 2

[7] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *CoRR*, abs/1512.03012, 2015. 2, 4

[8] Xiaojun Chang, Pengzhen Ren, Pengfei Xu, Zhihui Li, Xiaojiang Chen, and Alex Hauptmann. Scene graphs: A survey of generations and applications. *arXiv preprint arXiv:2104.01111*, 2021. 2

[9] Tianshui Chen, Weihao Yu, Riquan Chen, and Liang Lin. Knowledge-embedded routing network for scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2, 5

[10] Yujin Chen, Matthias Nießner, and Angela Dai. 4dcontrast: Contrastive learning with dynamic correspondences for 3d scene understanding. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 543–560, Cham, 2022. Springer Nature Switzerland. 2

[11] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 6

[12] 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. 5, 8

[13] Angela Dai, Christian Diller, and Matthias Niessner. Sg-nn: Sparse generative neural networks for self-supervised scene completion of rgb-d scans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[14] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 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, 2009. 2

[16] Helisa Dhamo, Azade Farshad, Iro Laina, Nassir Navab, Gregory D. Hager, Federico Tombari, and Christian Rupprecht. Semantic image manipulation using scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1, 2

[17] Helisa Dhamo, Fabian Manhardt, Nassir Navab, and Federico Tombari. Graph-to-3d: End-to-end generation and manipulation of 3d scenes using scene graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16352–16361, October 2021. 2, 3, 8

[18] Georgia Gkioxari, Nikhila Ravi, and Justin Johnson. Learning 3d object shape and layout without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1695–1704, June 2022. 2

[19] Jonathan Granskog, Till N Schnabel, Fabrice Rousselle, and Jan Novák. Neural scene graph rendering. *ACM Transactions on Graphics (TOG)*, 40(4):1–11, 2021. 1

[20] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2, 4, 7

[21] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16000–16009, June 2022. 2

[22] Roei Herzig, Moshiko Raboh, Gal Chechik, Jonathan Berant, and Amir Globerson. Mapping images to scene graphs with permutation-invariant structured prediction. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. 2

[23] Ji Hou, Benjamin Graham, Matthias Niessner, and Saining Xie. Exploring data-efficient 3d scene understanding with

contrastive scene contexts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15587–15597, June 2021. 2

[24] Siyuan Huang, Yichen Xie, Song-Chun Zhu, and Yixin Zhu. Spatio-temporal self-supervised representation learning for 3d point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6535–6545, October 2021. 2

[25] N. Hughes, Y. Chang, and L. Carlone. Hydra: A real-time spatial perception system for 3D scene graph construction and optimization. 2022. 1

[26] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1, 2

[27] Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. Image retrieval using scene graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 2

[28] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017. 2, 4

[29] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017. 2

[30] Manyi Li, Akshay Gadi Patil, Kai Xu, Siddhartha Chaudhuri, Owais Khan, Ariel Shamir, Changhe Tu, Baoquan Chen, Daniel Cohen-Or, and Hao Zhang. Grains: Generative recursive autoencoders for indoor scenes. *ACM Trans. Graph.*, 38(2), feb 2019. 2

[31] Rongjie Li, Songyang Zhang, Bo Wan, and Xuming He. Bipartite graph network with adaptive message passing for unbiased scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11109–11119, June 2021. 5

[32] Yikang Li, Wanli Ouyang, Bolei Zhou, Jianping Shi, Chao Zhang, and Xiaogang Wang. Factorizable net: An efficient subgraph-based framework for scene graph generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 2

[33] Yikang Li, Wanli Ouyang, Bolei Zhou, Kun Wang, and Xiaogang Wang. Scene graph generation from objects, phrases and region captions. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2, 5

[34] Yuanyuan Liu, Chengjiang Long, Zhaoxuan Zhang, Bokai Liu, Qiang Zhang, Baocai Yin, and Xin Yang. Explore contextual information for 3d scene graph generation. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–13, 2022. 5

[35] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. Visual relationship detection with language priors. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 852–869, Cham, 2016. Springer International Publishing. 3, 5

[36] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2837–2845, June 2021. 2

[37] Changsheng Lv, Mengshi Qi, Xia Li, Zhengyuan Yang, and Huadong Ma. Revisiting transformer for point cloud-based 3d scene graph generation. *arXiv preprint arXiv:2303.11048*, 2023. 2

[38] Siddharth Mahendran, Haider Ali, and Rene Vidal. 3d pose regression using convolutional neural networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2017. 4

[39] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[40] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 523–540, Cham, 2020. Springer International Publishing. 2

[41] 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. 3

[42] Mengshi Qi, Weijian Li, Zhengyuan Yang, Yunhong Wang, and Jiebo Luo. Attentive relational networks for mapping images to scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[43] Antoni Rosinol, Arjun Gupta, Marcus Abate, Jingnan Shi, and Luca Carlone. 3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans. 2020. 1

[44] Sayan Deb Sarkar, Ondrej Miksik, Marc Pollefeys, Daniel Barath, and Iro Armeni. Sgaligner : 3d scene alignment with scene graphs. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2023. 2

[45] Jonas Schult, Francis Engelmann, Alexander Hermans, Or Litany, Siyu Tang, and Bastian Leibe. Mask3D for 3D Semantic Instance Segmentation. 2022. 3

[46] Sahand Sharifzadeh, Sina Moayed Baharlou, and Volker Tresp. Classification by attention: Scene graph classification with prior knowledge. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(6):5025–5033, May 2021. 2

[47] Mohammed Suhail, Abhay Mittal, Behjat Siddiquie, Chris Broaddus, Jayan Eledath, Gerard Medioni, and Leonid Sigal. Energy-based learning for scene graph generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13936–13945, 2021. 1

[48] Johanna Wald, Armen Avetisyan, Nassir Navab, Federico Tombari, and Matthias Niessner. Rio: 3d object instance re-

localization in changing indoor environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 2, 5

[49] 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. 1, 2, 3, 4, 5, 8

[50] Johanna Wald, Nassir Navab, and Federico Tombari. Learning 3d semantic scene graphs with instance embeddings. *International Journal of Computer Vision*, 130(3):630–651, 2022. 1, 2, 5

[51] Kai Wang, Yu-An Lin, Ben Weissmann, Manolis Savva, Angel X. Chang, and Daniel Ritchie. Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks. 38(4), jul 2019. 2

[52] Ziqin Wang, Bowen Cheng, Lichen Zhao, Dong Xu, Yang Tang, and Lu Sheng. Vl-sat: Visual-linguistic semantics assisted training for 3d semantic scene graph prediction in point cloud. *arXiv preprint arXiv:2303.14408*, 2023. 2

[53] Sanghyun Woo, Dahun Kim, Donghyeon Cho, and In So Kweon. Linknet: Relational embedding for scene graph. *Advances in neural information processing systems*, 31, 2018. 1

[54] Shun-Cheng Wu, Keisuke Tateno, Nassir Navab, and Federico Tombari. Incremental 3d semantic scene graph prediction from rgb sequences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5064–5074, June 2023. 2

[55] 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. 1, 2, 5

[56] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 2

[57] Saining Xie, Jiatao Gu, Demi Guo, Charles R. Qi, Leonidas Guibas, and Or Litany. Pointcontrast: Unsupervised pretraining for 3d point cloud understanding. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 574–591, Cham, 2020. Springer International Publishing. 2

[58] Danfei Xu, Yuke Zhu, Christopher B. Choy, and Li Fei-Fei. Scene graph generation by iterative message passing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2, 3, 5

[59] Siming Yan, Zhenpei Yang, Haoxiang Li, Li Guan, Hao Kang, Gang Hua, and Qixing Huang. Implicit autoencoder for point cloud self-supervised representation learning. *CoRR*, abs/2201.00785, 2022. 2

[60] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 2

[61] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph r-cnn for scene graph generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 2, 3, 5

[62] Guojun Yin, Lu Sheng, Bin Liu, Nenghai Yu, Xiaogang Wang, Jing Shao, and Chen Change Loy. Zoom-net: Mining deep feature interactions for visual relationship recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 322–338, 2018. 1

[63] Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. Neural motifs: Scene graph parsing with global context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[64] Chaoyi Zhang, Jianhui Yu, Yang Song, and Weidong Cai. Exploiting edge-oriented reasoning for 3d point-based scene graph analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9705–9715, June 2021. 1, 2, 5

[65] Shoulong Zhang, Shuai Li, Aimin Hao, and Hong Qin. Knowledge-inspired 3d scene graph prediction in point cloud. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 18620–18632. Curran Associates, Inc., 2021. 1, 2

[66] Zaiwei Zhang, Rohit Girdhar, Armand Joulin, and Ishan Misra. Self-supervised pretraining of 3d features on any point-cloud. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10252–10263, October 2021. 2, 6

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. 6), **(ii)** more details on the dataset and its pre-processing (Sec. 7), **(iii)** further ablations on our architecture design (Sec. 8), **(iv)** a direct qualitative comparison between our method with and without pre-training for 3D scene graph prediction (Sec. 9), **(v)** additional 3D scene graph predictions using our method(Sec. 10), **(vi)** additional scene generations from our method (Sec. 11),

6. Reproducibility Details

6.1. 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.

6.2. 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.

6.3. Losses

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

$$\mathcal{L}_{\text{rec}} = \frac{1}{N}\sum_{i=1}^{N}\left(\eta_1\|\hat{b}_i - b_i\|_1 + \eta_2 CE(\hat{\theta}_i, \theta_i) + \eta_3\|\hat{e}_i - e_i\|_1\right) \tag{9}$$

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

$$\mathcal{L}_{\text{SG}} = \frac{1}{N}\sum_{i=1}^{N}\lambda_1 CE\left(\hat{o}_i, o_i\right) + \frac{1}{M}\sum_{j=1}^{M}\lambda_2 BCE\left(\hat{p}_j, p_j\right) \tag{10}$$

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

$$\mathcal{L} = -\alpha_t(1 - p_t)^{\gamma}\log p_t \tag{11}$$

with $\alpha = 0.25$ and $\gamma = 2$. However, we do not use manual class weighting based on object and predicate occurrences like SGFN [55].

7. Dataset Details

Following Wald et al. [49], 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. [49]. 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 [12] and S3DIS [3] 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 [49] as best as possible. Moreover, before feeding object point clouds and pairs of objects' point clouds into the PointNets [41], we apply farthest-point-sampling to downsample the point clouds of each object to at most 1000 points.

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

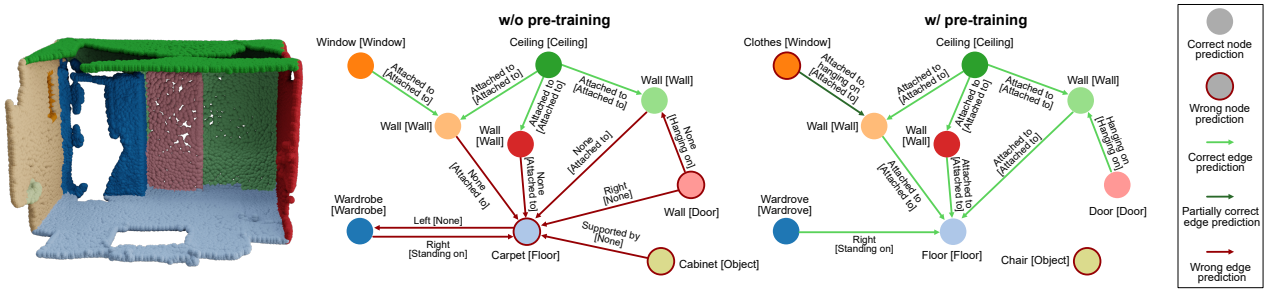


Figure 5. **Qualitative evaluation of the effects of pre-training on 3DSSG.** A qualitative comparison of a scene graph predicted for a 3D scene (*left*), with no pre-training (*middle*) and with pre-training (*right*). We visualize the top-1 object class prediction for each node and the predicates with 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.

	Object		Predicate	
Method	R@5	mR@5	R@3	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 7. **Ablations point cloud pre-training approaches**.

8. Architecture Ablations

In Tab. 7 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.

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.

9. Qualitative effect of pre-training

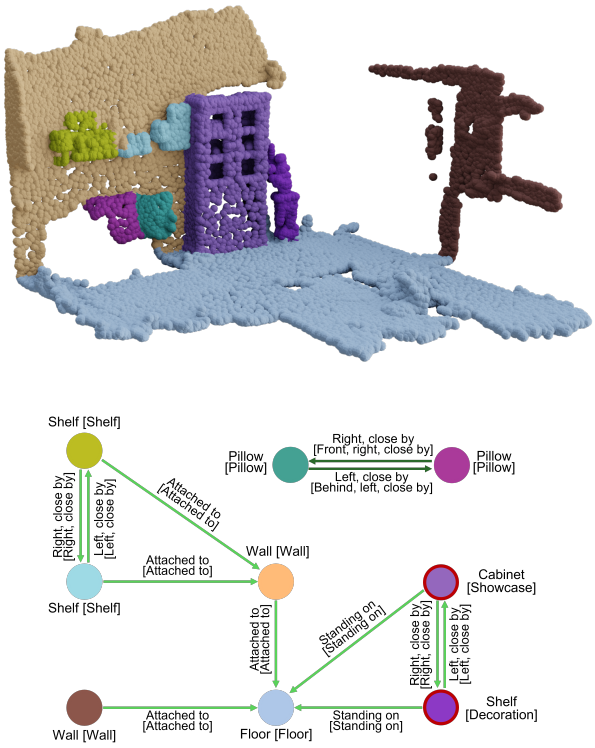
in Fig. 5, we provide a direct comparison between our method pre-trained using our pre-training approach and the same method trained from scratch on 3D scene graph prediction. We observe that using our pre-training drastically improves the prediction accuracy of nodes and edges. The predicted 3D scene 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.

10. Scene Graph Results

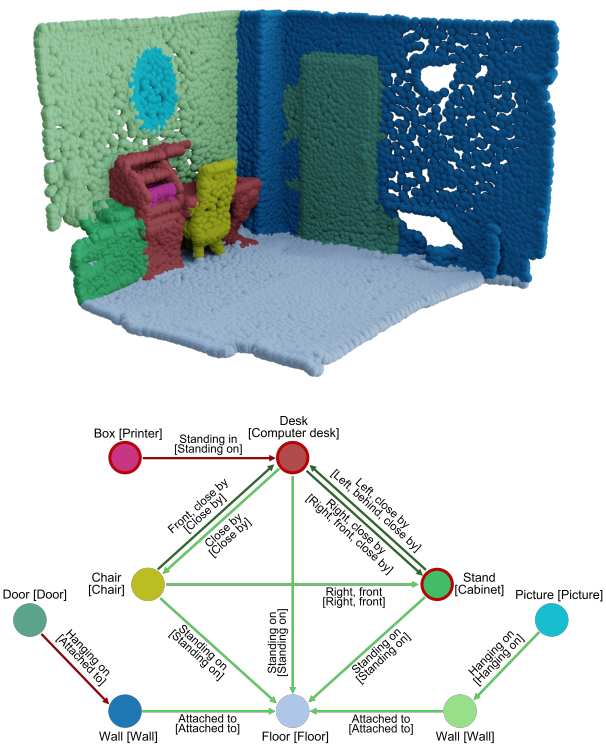
In Fig. 6, we provide additional scene graph visualizations. We observe that our network is able to produce almost perfect scene graphs in very diverse scenes. Some common misclassification cases include incorrect edges where either the ground truth is none and our network predicts a relationship or our network does not predict a relationship when it is present.

11. Scene Generation Results

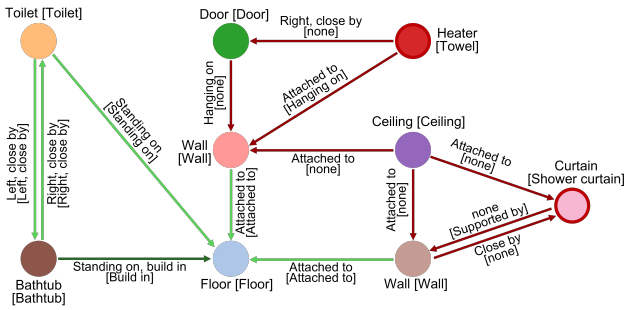
In Fig. 7, we provide additional scene reconstructions. The shown scene generations support those shown in the paper. Although the generated shapes are not perfect, our model seems to preserve the relationships in the original scenes.


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]

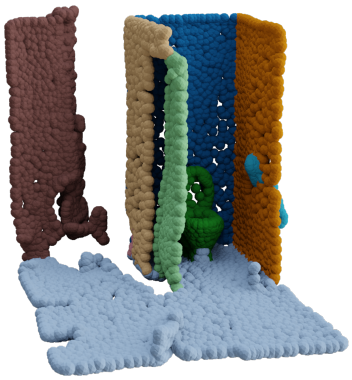
(a)

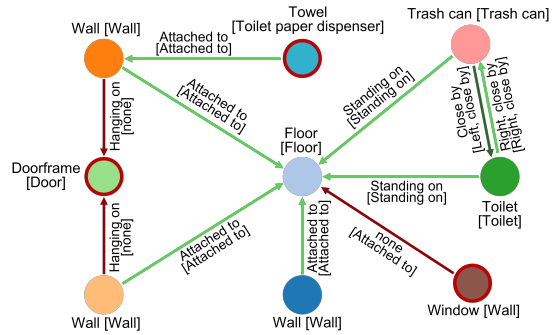

Box [Printer]
Standing in [Standing on]
Desk [Computer desk]
Front, close by [Close by]
Close by [Close by]
[Left, close by [Left, behind, close by]
Right, close by [Right, front, close by]
Chair [Chair]
Stand [Cabinet]
Right, front [Right, front]
Door [Door]
Picture [Picture]
Hanging on [Attached to]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

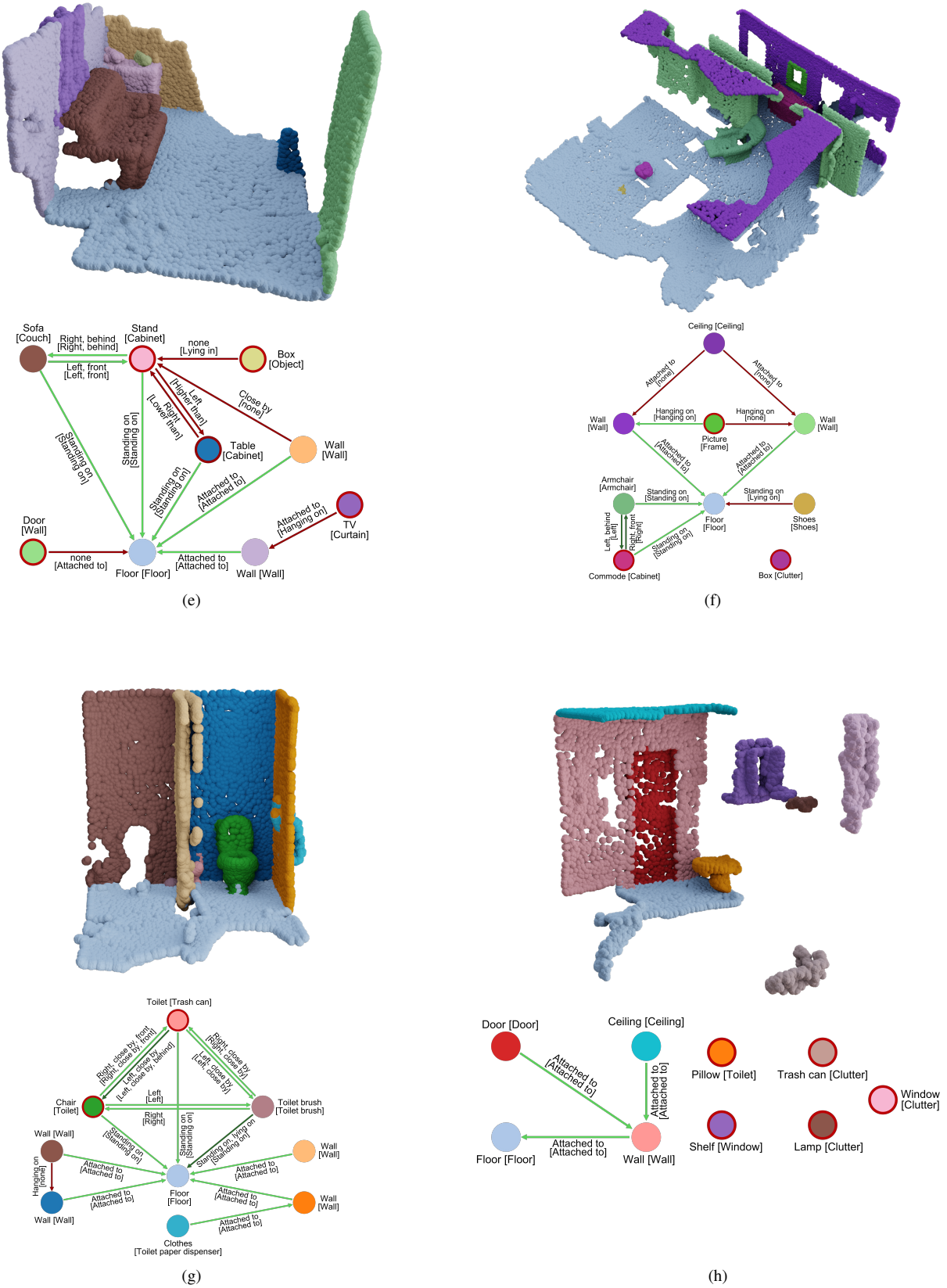
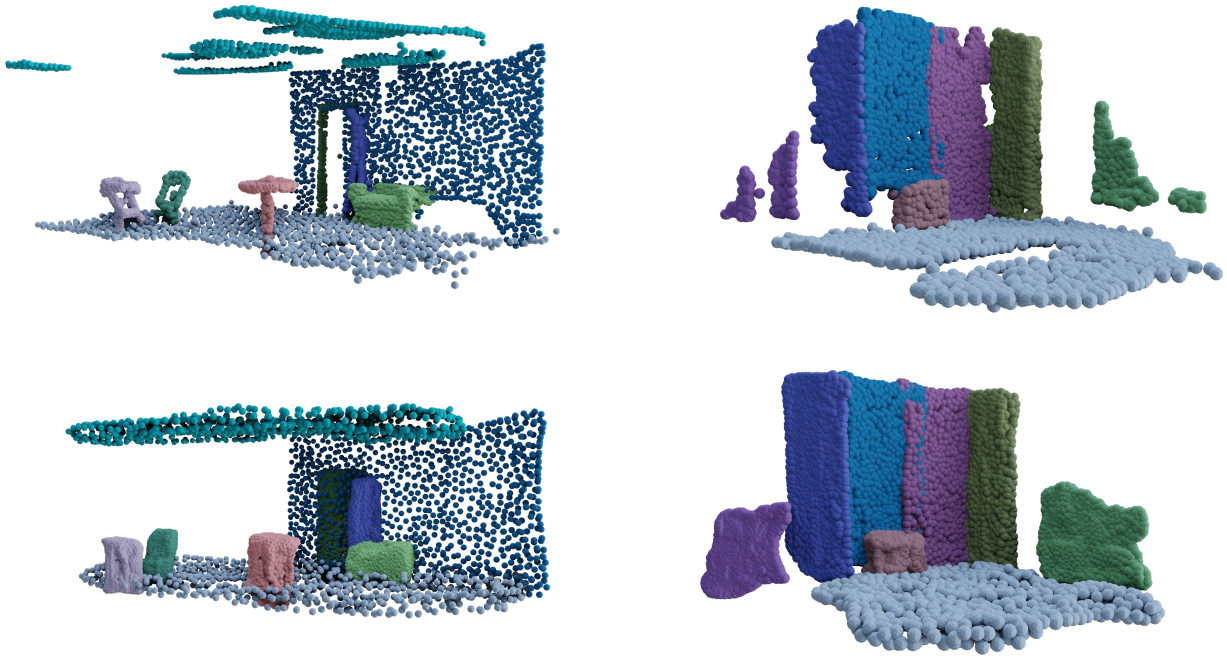
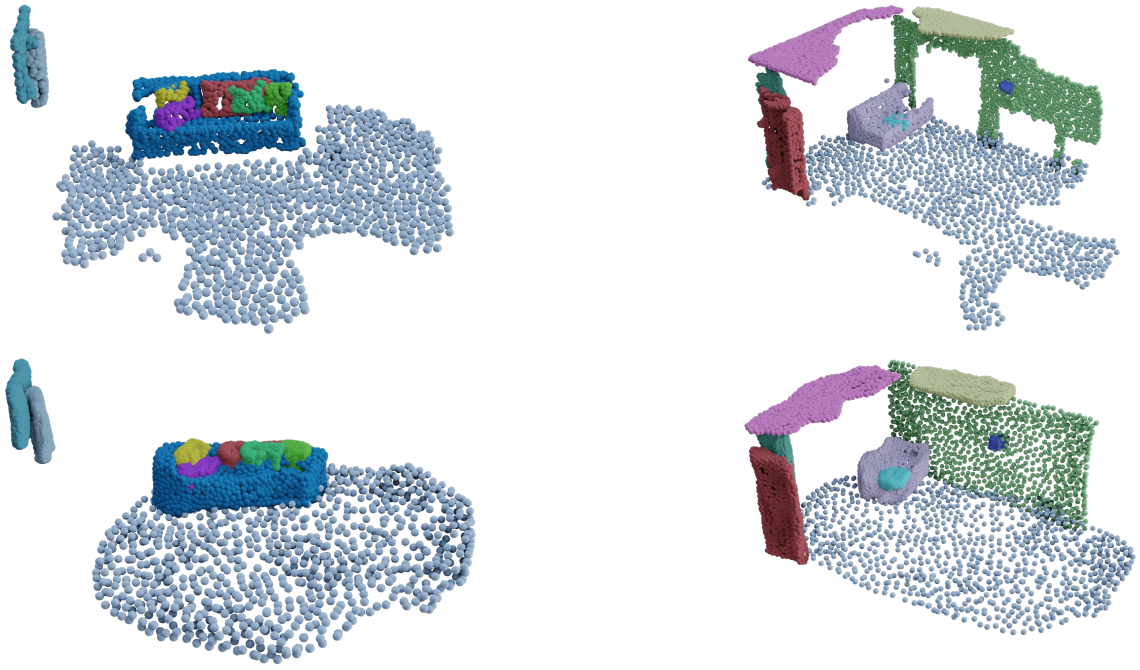


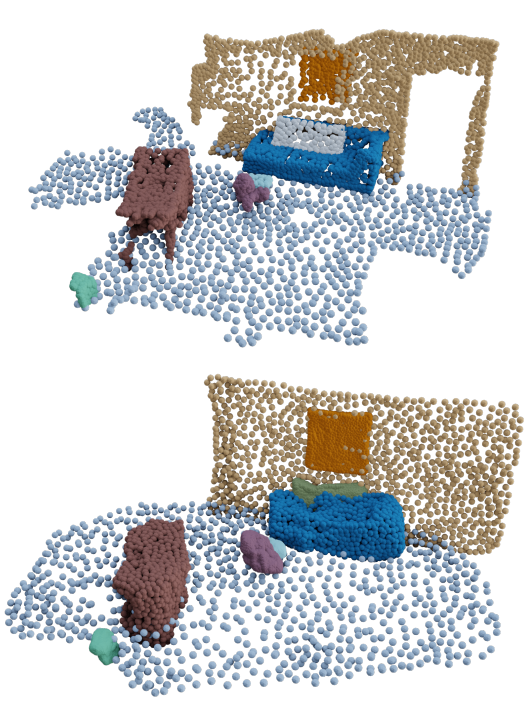
Figure 6. **Qualitative scene graph results from SGRec3D.** Top: 3D scene; Bottom: Predicted 3D scene graph.



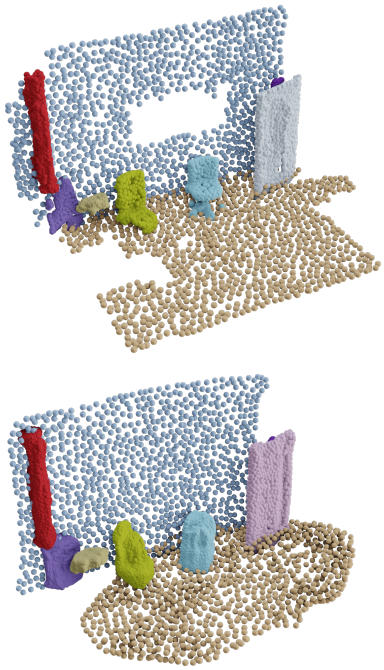
(a) (b)



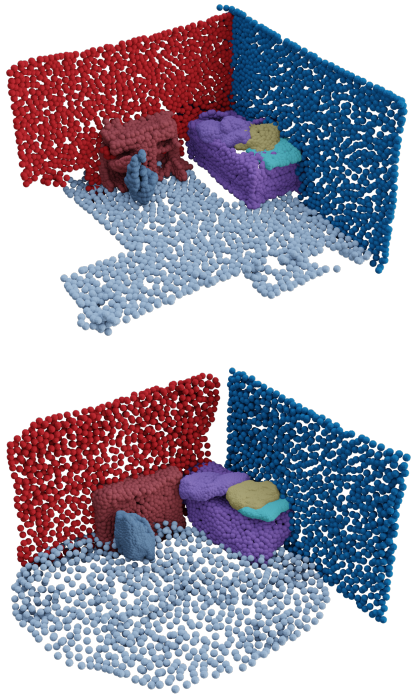
(c) (d)



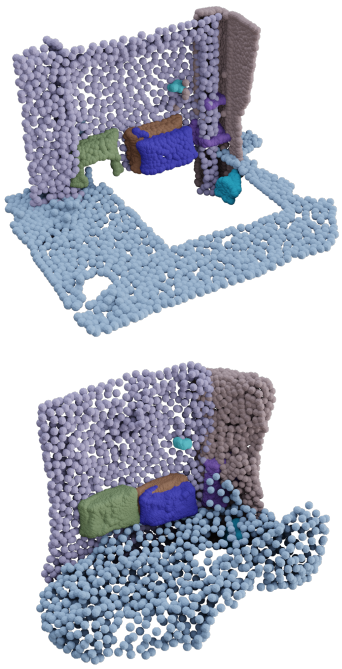
(e)



(f)



(g)



(h)

Figure 7. **Qualitative scene generation results from SGRec3D.** Top: Original 3D scene; Bottom: Reconstructed scene using SGRec3D.