A. Supplementary Material
Graph-to-3D: End-to-End Generation and Manipulation of 3D Scenes Using Scene Graphs

This document supplements our main paper entitled *Graph-to-3D: End-to-end Generation and Manipulation of 3D Scenes using Scene Graphs* by providing i) more details on the data preparation and ii) inference mode. We iii) give additional information on the employed GCN and discriminators, as well as iv) shape generation networks. Further, we provide v) more details on the employed baselines. We vi) clarify the used metrics, *i.e*. define the computation of our scene graph constraints used in the layout evaluation and provide details of the top-K recall. Finally, vii) we report the results of a user study and viii) demonstrate qualitative and diversity results for the generated 3D scenes.

A.1. Data preparation and annotation

In this section we provide more details on our data preparation pipeline, used to obtain and refine oriented bounding boxes for 3RScan objects (*c.f*. Figure 1 for a common ground truth scene example that illustrates the reconstruction partiality, as well as the respective scene graph from 3DSSG).

Extraction of 3D bounding boxes Figure 2 illustrates the oriented bounding box preparation pipeline presented in the main paper, in top-down view. Given the original point cloud of an object (violet), the algorithm identifies the point cloud rotation (blue) that leads to smallest surface area for the axis-aligned bounding box. This rotation is then used to transform the identified box back in the original point cloud coordinates (green).

Canonical pose annotation We map the 160 object classes in 3DSSG [7] to the RIO27 label set from 3RScan [6] and divide them in three categories, based on their symmetry properties. Table 1 gives the full object class list for each annotation category.

1. *Objects with two symmetry axes* such as tables, bathtubs, desks, walls, are annotated automatically, considering the direction with the largest size as front.

2. For a subset of objects, such as cabinet, shelf and oven, we annotate automatically based on the following observation. Given that such objects are usually attached to a vertical surface (wall) the 3D reconstruction for their back side is missing. Therefore, we first apply the rule of subset 1. to identify the directionless axis and then define the front side of the object as the direction where the center of mass is leaning towards.

3. *Objects with one symmetry axis* such as chair, sofa, sink, bed are annotated manually. The annotator is presented with the object point cloud, inside an oriented bounding box, and is given four choices regarding the front direction of the object.

Category	Classes
1	table desk wall floor door window tv curtain ceiling box bathtub object
2	cabinet nightstand shelf fridge lamp blanket clothes oven towel pillow
3	chair sofa bed toilet sink

Table 1: Annotation categories mapped to RIO27 label set.

A.2. Inference

Generation Given a scene graph, we first sample a random vector per-node from the gaussian prior. Then we feed the augmented scene graph (class embeddings and sampled vectors) to the shape and layout decoders to recover a 3D scene.

Manipulation We first encode the input scene given the scene graph (newly added nodes are again sampled from the gaussian prior). We then run $\mathcal{T}$ to update the latent of the changed nodes w.r.t. the new graph, decode the scene and add the changes to the input scene.

A.3. Implementation details

We use 5 layers for each GCN model. In the encoders $\mathcal{E}_{\text{shape}}$ and $\mathcal{E}_{\text{layout}}$, prior to the GCN computation, the class categories o_i and r_{ij} are fed to embedding layers, while the shape embedding, box and angles are projected through a linear layer. All discriminators consist of fully-connected

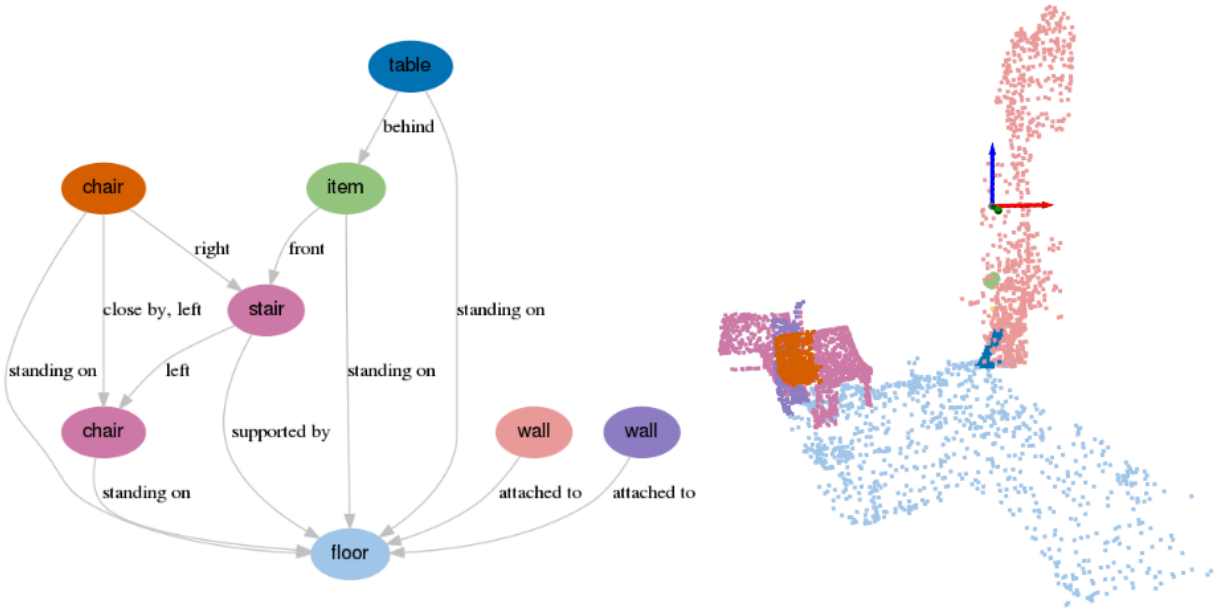


Figure 1: Example of ground truth graph from 3DSSG and the respective 3D scan from the 3RScan dataset.

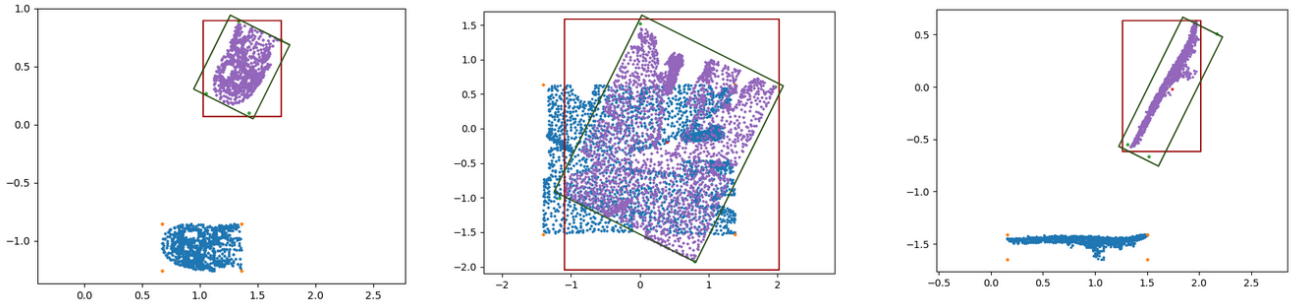


Figure 2: **Data preparation** (Top-down view of 3D point clouds). Violet: Original point cloud rotation. Blue: Point cloud in rotation that gives the smallest axis-aligned surface area. Red: axis-aligned box. Green: Oriented bounding box resulting from our data preparation.

layers, where all layers apart from the last one are followed by batch norm and Leaky-ReLU. For D_{box} (Table 2), consisting of 3 layers, the last fully-connected layer is followed by a sigmoid. Here the class categories o_i and r_{ij} are fed in one-hot form giving a size of 160 and 26. For D_{shape} (Table 3) after two consecutive layers, we employ two branches of fully-connected layers, followed by namely a softmax (for classification, outC) and sigmoid (for discrimination, outD). We use the Adam optimizer with a learning rate of 0.001 and batch size of 8, to train the model for 100 epochs. The training takes one day on one Titan Xp GPU. The loss weights are set to $\lambda_{KL} = 0.1$, $\lambda_{D,b} = 0.1$ and $\lambda_{D,s} = 0.1$. The shape embeddings e_i^s have a size of 128.

A.4. Generation of 3D scenes $\mathcal{F}_{gen}$

Point clouds We base our point cloud approach on AtlasNet [2]. In particular, we employ AtlasNet to learn a low-dimensional latent space embedding on the point clouds. AtlasNet is grounded on PointNet [5] and consumes a whole point cloud which it then encodes into a global feature descriptor $\mathcal{E}_{atlas}$. AtlasNet is particularly suited since the sampling on the uv-map allows to generate point clouds at arbitrarily resolutions while only using a small set of points during training. This significantly speeds up training while saving memory, thus allowing larger batch sizes. The 3D point cloud can be inferred by using this global fea-

layer id	layer type	input layer	input channels	output channels
L1	Linear	$(o_i, o_j, r_{ij}, b_i, b_j)$	360	512
L2	Batch Norm	L1	512	512
L3	Leaky-ReLU	L2	512	512
L4	Linear	L3	512	512
L5	Batch Norm	L4	512	512
L6	Leaky-ReLU	L5	512	512
L7	Linear	L6	512	1
out	Sigmoid	L7	1	1

Table 2: Architecture of D_{box}

layer id	layer type	input layer	input channels	output channels
L1	Linear	e_i^s	128	512
L2	Batch Norm	L1	512	512
L3	Leaky-ReLU	L2	512	512
L4	Linear	L3	512	512
L5	Batch Norm	L4	512	512
L6	Leaky-ReLU	L5	512	512
L7	Linear	L6	512	1
outD	Sigmoid	L7	1	1
L9	Linear	L6	512	160
outC	Softmax	L9	160	160

Table 3: Architecture of D_{shape}

ture descriptor together with sampled 2D points from the aforementioned uv-map running them through the decoder $\mathcal{D}_{atlas}$. We train AtlasNet on a mixture of synthetic data from ShapeNet and real 3RScan objects in canonical pose.

Implicit functions In addition, we also employ implicit functions as shape representation using DeepSDF [4]. To this end, we train an individual Auto-Decoder for each class using ShapeNet [1]. Thereby, we use 350 shapes in canonical pose and learn a 128-dimensional continuous shape space. We then label each object in 3RScan with the best fitting descriptor. Initially, we attempted to use a similar partial scan alignment as originally proposed in [4]. Yet, this did not work well in practice as the point quality was too low. Hence, we instead simply queried each learned descriptor from our shape space with the 3D points of the object, and labeled the object with the descriptor giving minimal average error in SDF. Notice that since we learn a generative model on top of these labels, Graph-to-3D can still exploit the full potential of the continuous shape space.

A.5. Baseline details

For the **variational AtlasNet** (shape model without without GCN) we enforce a Gaussian distribution onto the embedding space of AtlasNet (AtlasNet VAE). In this model, the shapes are generated without awareness of the neighbouring objects. For a given point cloud s_i we can compute the posterior distribution $(\mu_i, \sigma_i) = \mathcal{E}_{gen}(s_i)$ with (μ, σ) being the mean and log-variance of a diagonal Gaussian distribution. Sampling from the posterior allows to generate on the fly new shapes during inference.

The **progressive model** receives at each step the current scene, together with a new node n_a to be added. Thereby, for the current scene nodes n, the model $\mathcal{A}$ receives the 3D boxes b as well as the category labels o for nodes and edges r and predicts the new box according to $b_i = \mathcal{A}(o_i, r_i j, z_i, o, r, z)$. Here z_i denotes randomly sampled noise from a normal distribution with zero mean and unit standard deviation. Note that for the new node n_i, we only feed the object category o_i as well as its relationships $r_i j$ with existing objects j. During inference, the method assumes the first node given, then gradually adds nodes and connections. In manipulation mode, the method receives a ground truth scene and a sequence of novel nodes to be added. We train the progressive baseline with varying graph sizes (2-10), such that it can learn to predict the consecutive node for different generation steps. We order the nodes based on the graph topology of the support relationships, *e.g.* pillow is generated after the supporting bed. In addition, we place the disconnected nodes last in order.

A.6. Metrics

Scene graph constraints For layout evaluation *w.r.t* the employed scene graph constraints, our metrics follow the definitions from Table 4. Though ideally we want to validate all edges in 3DSSG, not all of them can be captured with a geometric rule, as they are manually annotated (*e.g.* *belonging to*, *leaning against*).

Relationship	Rule
left of right of	$c_{x,i} < c_{x,j}$ and $\text{iou}(b_i, b_j) < 0.5$ $c_{x,i} > c_{x,j}$ and $\text{iou}(b_i, b_j) < 0.5$
front of behind of	$c_{y,i} < c_{y,j}$ and $\text{iou}(b_i, b_j) < 0.5$ $c_{y,i} > c_{y,j}$ and $\text{iou}(b_i, b_j) < 0.5$
higher than lower than	$h_i + c_{z,i}/2 > h_j + c_{z,j}/2$ $h_i + c_{z,i}/2 < h_j + c_{z,j}/2$
smaller than bigger than	$w_i l_i h_i < w_j l_j h_j$ $w_i l_i h_i > w_j l_j h_j$
same as	$\text{iou}_C(b_i, b_j) > 0.5$

Table 4: Computation of geometric constraint accuracy, for two instances i and j. iou_C refers to iou computation after both objects have been 0-centered.

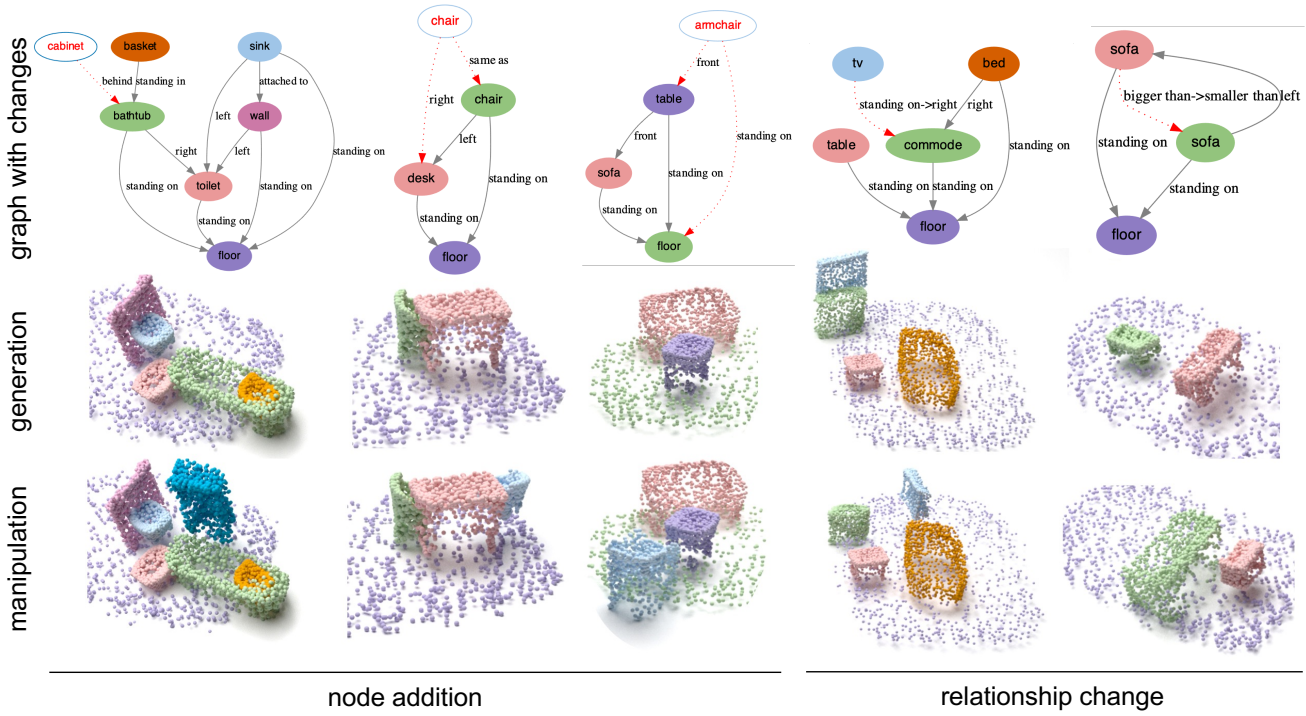


Figure 3: Generation (middle) and manipulation (bottom) of full 3D scenes from scene graphs (top) for the Graph-to-3D model based on AtlasNet encodings for shape. The graph also contains the applied changes, in the form of dashed lines for new/changed relationship, and empty nodes for added objects.

Top-K recall We utilize the same top-K recall metric as in 3DSSG [7] to evaluate the SGPN predictions. For each object node or predicate, the top-K metric checks if the ground truth label is within the corresponding top k classification scores. To obtain a triplet score, we multiply the scores of the two respective objects as well as the relationship predicate. Then, similarly, we check if the ground truth triplet is among the top-K scores.

A.7. User study

We conducted a perceptual study with 20 people evaluating ≈30 scenes each. Each sample features a scene graph, the 3D-SLN [3] baseline with retrieved shapes from ShapeNet and our shared model (given anonymously and in random order). The users then rated each scene 1-7 on three aspects 1) global correctness, 2) functional and style fitness between objects and 3) correctness of graph constraints. 3D-SLN reported $2.8, 3.7, 3.6$, respectively, while ours exceeded them with $4.6, 4.9, 5.4$. Our method was preferred in namely $72\%, 62\%,$ and 68% of the cases.

A.8. Additional Qualitative Results

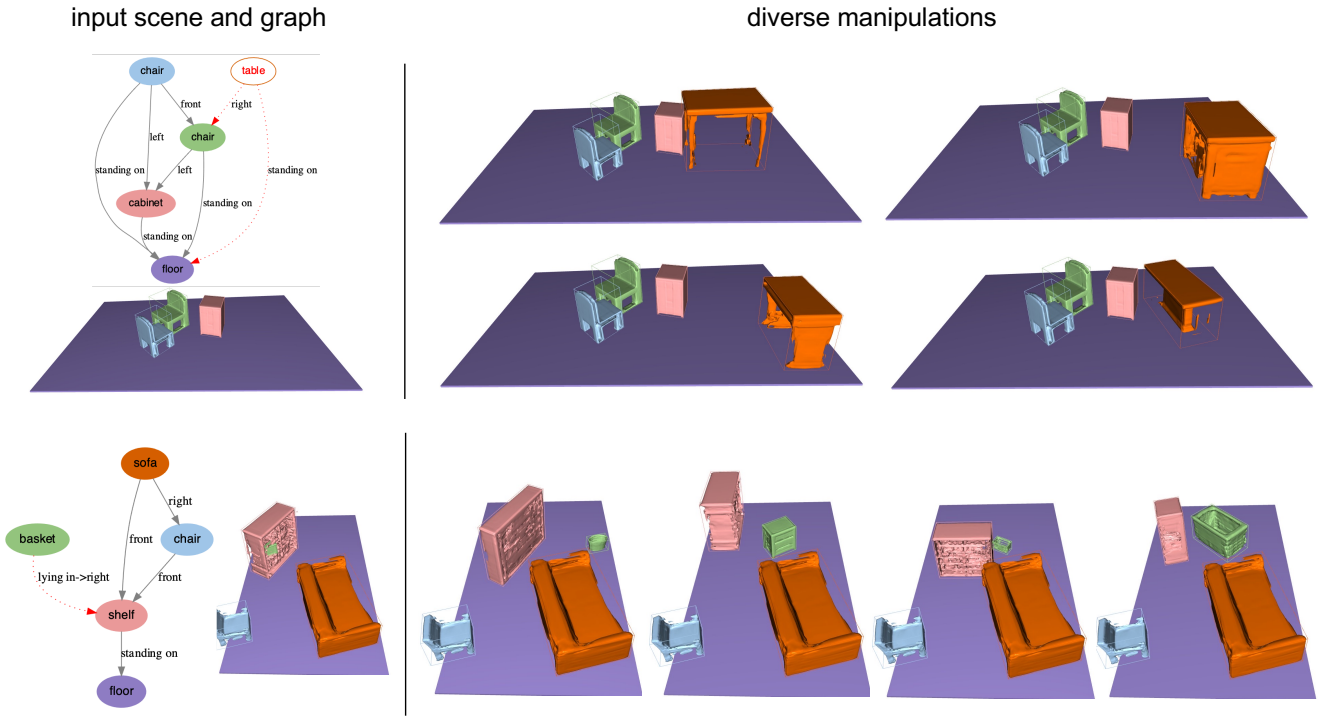
In this section we demonstrate additional qualitative results for 3D layouts and full 3D scenes with shapes from AtlasNet [2] as well as DeepSDF [4]. We would like to emphasize that in our manipulation experiments, we intentionally allow the network to also change the shape of the objects that are involved in a relationship change, to demonstrate diversity. Nonetheless, notice that we can alternatively keep the shape unchanged, *i.e.* as in the original scene, via transforming the original shape with the predicted pose.

A.8.1 3D scene generation and manipulation with the AtlasNet based model

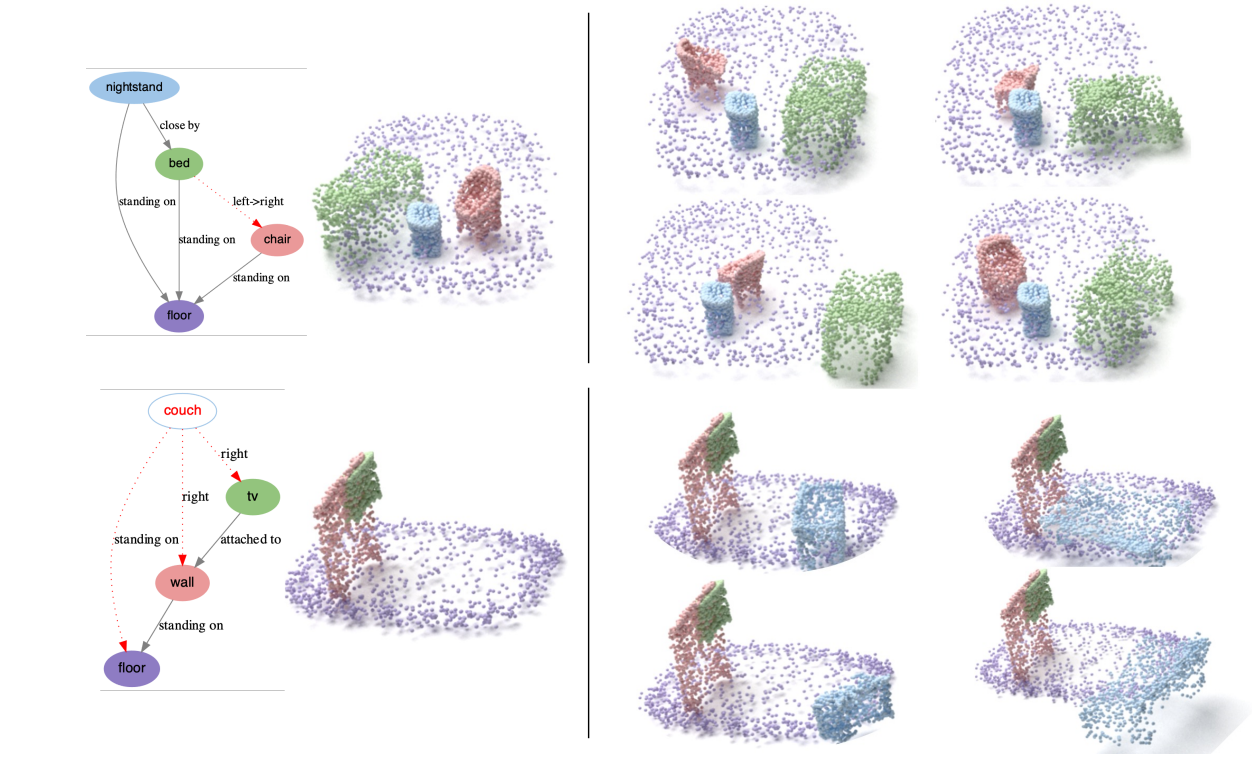
Figure 3 shows generation and manipulation of 3D scenes from scene graphs using the Graph-to-3D model together with AtlasNet [2]. It can be noted that similarly to the DeepSDF [4] based encodings (*c.f.* Figure 4 in the main paper), the model based on AtlasNet encodings is capable of generating correct point clouds under their diverse class categories, which are consistent with their semantic relationships. Further the manipulations are also appropriate with respect to changes in the graphs.

A.8.2 Diverse scene generation

In Figure 4 we want to demonstrate that Graph-to-3D is able to generate a diverse set of manipulations. To this end, we



(a) DeepSDF encodings



(b) AtlasNet encodings

Figure 4: Diverse generation of shapes and layout during manipulation. Given an input graph and correspondingly generated scene (left), we obtain diverse results (right) for the added or changed objects.

first generate a scene given only a semantic scene graph. Subsequently, we apply changes including additions and relationship changes to the graph and let the model repeatedly incorporate them. Notice that we run this experiment on top of both generative models, *i.e*. AtlasNet and DeepSDF (*c.f*. Figure 4 a) and b)). Hence, for the same input, Graph-to-3D is capable of incorporating diverse manipulations in terms of both – 3D shape as well as 3D location and orientation.

References

[1] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[2] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[3] Andrew Luo, Zhoutong Zhang, Jiajun Wu, and Joshua B. Tenenbaum. End-to-end optimization of scene layout. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[4] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[5] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016.

[6] Johanna Wald, Armen Avetisyan, Nassir Navab, Federico Tombari, and Matthias Nießner. RIO: 3D Object Instance Re-Localization in Changing Indoor Environments. In *International Conference on Computer Vision (ICCV)*, 2019.

[7] Johanna Wald, Helisa Dhamo, Nassir Navab, and Federico Tombari. Learning 3d semantic scene graphs from 3d indoor reconstructions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.