Graph-to-3D: End-to-End Generation and Manipulation of 3D Scenes Using Scene Graphs

Helisa Dhamo [1,*] Fabian Manhardt [2,*] Nassir Navab [1] Federico Tombari [1,2]
[1] Technische Universität München [2] Google

Abstract

Controllable scene synthesis consists of generating 3D information that satisfy underlying specifications. Thereby, these specifications should be abstract, i.e. allowing easy user interaction, whilst providing enough interface for detailed control. Scene graphs are representations of a scene, composed of objects (nodes) and inter-object relationships (edges), proven to be particularly suited for this task, as they allow for semantic control on the generated content. Previous works tackling this task often rely on synthetic data, and retrieve object meshes, which naturally limits the generation capabilities. To circumvent this issue, we instead propose the first work that directly generates shapes from a scene graph in an end-to-end manner. In addition, we show that the same model supports scene modification, using the respective scene graph as interface. Leveraging Graph Convolutional Networks (GCN) we train a variational Auto-Encoder on top of the object and edge categories, as well as 3D shapes and scene layouts, allowing latter sampling of new scenes and shapes.

1. Introduction

Scene content generation, including 3D object shapes, images and 3D scenes is of high interest in computer vision. Applications involve helping the work of designers through automatically generated intermediate results, as well as understanding and modeling scenes, in terms of, *e.g.*, object constellations an co-occurrences. Furthermore, conditional synthesis allows for a more controllable content generation, since users can specify which image or 3D model they want to let appear in the generated scene. Common conditions involve text descriptions [39], semantic maps [34] and scene graphs. Thereby, scene graphs have recently shown to offer a suitable interface for controllable synthesis and manipulation [11, 4, 20], enabling semantic control on the generated scene, even for complex scenes. Compared to dense semantic maps, scene graph structures are more high-level and explicit, simplifying the interaction with the user. Moreover, they enable controlling the semantic relation between entities, which is often not captured in a semantic map.

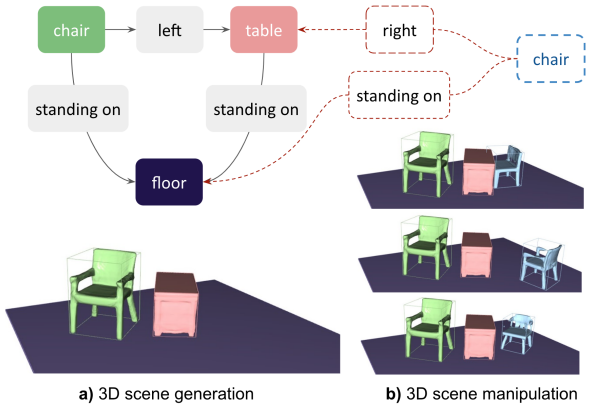

Figure 1. a) *Scene generation*: given a scene graph (top, solid lines), Graph-to-3D generates a 3D scene consistent with it. b) *Scene manipulation*: given a 3D scene and an edited graph (top, solid+dotted lines), Graph-to-3D is able to generate a varied set of 3D scenes adjusted according to the graph manipulation.

While there are a lot of methods for scene graph inference from images [36, 23] as well as the reverse problem [11, 2], in the 3D domain, only a few works on scene graph prediction from 3D data have been very recently presented [31, 35]. With this work, we thus attempt to fill this gap by proposing a method for end-to-end generation of 3D scenes from scene graphs. A few recent works investigate the problem of scene layout generation from scene graphs [32, 20], thereby predicting a set of top-view object occupancy regions or 3D bounding boxes. To construct a 3D scene from this layout, these methods typically rely on retrieval from a database. On the contrary, we employ a fully generative model that is able to synthesize novel context-aware 3D shapes for the scene. Though retrieval leads to good quality results, shape generation is an emerging alternative as it allows further costumizability via interpolation at the object level [8] and part level [22]. Further, retrieval works can achieve at best (sub-) linear complexity

*The first two authors contributed equally to this work

for time and space w.r.t. database size. Our method essentially predicts object-level 3D bounding boxes together with appropriate 3D shapes, which are then combined to create a full 3D scene (Figure 1, left). Leveraging Graph Convolutional Networks (GCNs) we learn a variational Auto-Encoder on top of scene graphs, 3D shapes and scene layouts, enabling latter sampling of novel scenes. Additionally, we employ a graph manipulation network to enable changes, such as adding new objects as well as changing object relationships, while maintaining the rest of the scene (Figure 1, right). To model the one-to-many problem of label to object, we introduce a novel relationship discriminator on 3D bounding boxes that does not limit the space of valid outputs to the annotated box.

To avoid inducing any human bias, we want to learn 3D scene prediction from real data. However, these real datasets, such as 3RScan typically present additional limitations, such as information holes and, oftentimes, lack of annotations for the canonical object pose. We overcome the former limitation by refining the ground truth 3D boxes based on the semantic relationships from 3DSSG [31]. For the latter, we extract oriented 3D bounding boxes and annotate the front side of each object, using a combination of class-level rules and manual annotations. We release these annotations as well as the source code on our project page[1].

Our contributions can be summarized as: i) We propose the first fully learned method for generating a 3D scene from a scene graph. Therefore, we use a novel model for shared layout and shape generation. ii) We also adopt this generative model to simultaneously allow for scene manipulation. iii) We introduce a relationship discriminator loss which is better suited than reconstruction losses due to the one-to-many problem of box inference from class labels. iv) We label 3RScan with canonical object poses.

We evaluate our proposed method on 3DSSG [31], a large-scale real 3D dataset based on 3RScan [30] that contains semantic scene graphs. Thereby, we evaluate on common aspects of scene generation and manipulation, such as quality, diversity and fulfillment of relational constrains, showing compelling results, as well as an advantage of sharing layout and shape features for both tasks.

2. Related work

Scene graphs and images Scene graphs [12, 14] refer to a representation that provides a semantic description for a given image. Whereas nodes depict scene entities (objects), edges represent the relationships between them. A line of works focuses on scene graph prediction from images [36, 9, 27, 38, 18, 37, 17, 23]. Other work explore scene graphs for tasks such as image retrieval [12], image generation [11, 2] and manipulation [4].

[1]Project page: https://he-dhamo.github.io/Graphto3D/

Scene graphs in 3D The 3D computer vision and graphics communities have proposed a diverse set of scene graph representations and related structures. Scenes are often represented through a hierarchical tree, where the leaves are typically objects and the intermediate nodes form (functional) scene entities [16, 19, 40]. Armeni *et al*. [1] propose a hierarchical mapping of 3D models of large spaces in four layers: camera, object, room and building. Wald *et al*. [31] introduce 3DSSG, a large scale dataset with dense semantic graph annotations. These graph representations are utilized to explore tasks related to scene comparison [6], scene graph prediction [31], 2D-3D scene retrieval [31], layout generation [20], object type predictions in query locations [41], as well as to improve 3D object detection [28].

3D scene and layout generation A line of works generates 3D scenes conditioned on images [29, 24]. Jiang *et al*. [10] use probabilistic grammar to control scene synthesis. Other works, more related to ours, incorporate graph structures. StructureNet [22] explores an object-level hierarchical graph, to generate shapes in a part-aware model. Ma *et al*. [21] convert text to a scene graph with pairwise and group relationships, to progressively retrieve subscenes for 3D synthesis. While generative methods were recently explored for layouts of different types [13], some methods focus on generating scene layouts. GRAINS [16] explore hierarchical graphs to generate 3D scenes, using a recursive VAE that generates a layout, followed by object retrieval. Luo *et al*. [20] generate a 3D scene layout conditioned on a scene graph, combined with a rendering approach to improve image generation. Other works use deep priors [33] or relational graphs [32] to learn object occupancy in the top-view of indoor scenes.

Different from our work, these works either explore images as final output, use 3D models based on retrieval, or operate on synthetic scenes. Hence, these methods can either not fully explain the actual 3D scene or are not capable of generating context-aware real compositions.

3. Data preparation

Our approach is built on top of 3DSSG [31], a scene graph extension of 3RScan [30], which is a large-scale indoor dataset with $\sim$1.4k real 3D scans. 3RScan does not contain canonical poses for objects, which is essential to learning object pose and shape as well as many other tasks.

Therefore, we implemented a fast semi-automatic annotation pipeline to obtain canonical tight bounding boxes per instance. As most objects are supported by a horizontal surface, we model the oriented boxes with 7 degrees-of-freedom (7DoF), *i.e*. 3 for size, 3 for translation as well as 1 for the rotation around the z-axis. Since the oriented bounding box should fully enclose the object whilst possessing minimal volume, we use volume as criteria to optimize the

rotational parameter. First, for each object we extract the point set p. Then, we gradually rotate the points along the z-axis using angles α in the range $[0, 90[$ degrees, with a step of 1 degree, $p_t = R(\alpha)p$. At each step, we extract the axis-aligned bounding box from the transformed point set p_t, by simply computing the extrema along each axis. We estimate the area of the 2D bounding box in bird's eye view, after applying an orthogonal projection onto the ground plane. We then label the rotation $\hat{\alpha}$ having the smallest box top-down view area (*c.f.* supplementary material). From this box we extract the final box parameters: width w, length l and height h, rotation $\hat{\alpha}$ as well as centroid (c_x, c_y, c_z).

The extracted bounding box remains still ambiguous, as there are always four possible solutions regarding the facing direction. Hence, for objects with two or more vertical axes of symmetry, such as tables, we automatically define as front the largest size component (in line with ShapeNet [3]). For all other objects such as chair or sofa the facing direction is annotated manually (4.3k instances in total).

As 3D boxes are obtained from the object point clouds, we observe misalignments due to impartial scans. Objects are oftentimes not touching their supporting structures, *e.g.* chair with missing legs leads to a "flying" box detached from the floor. We thus detect inconsistencies using the support relationships from [31]. If an object has a distance of more that $10cm$ from its support, we fix the respective 3D box, such that it reaches the upper level of the parent object. For planar support such as floor, we employ RANSAC [5] to fit a plane in a neighbourhood around the object and extend the object box so that it touches the fitted plane.

4. Methodology

In this work we propose a novel method for generating full 3D scenes from a given scene graph in a fully learned and end-to-end fashion. In particular, given a scene graph $G = (\mathcal{O}, \mathcal{R})$, where nodes $o_i \in \mathcal{O}$ are semantic object labels and edges $r_{ij} \in \mathcal{R}$ are semantic relationship labels with $i \in \{1, ..., N\}$ and $j \in \{1, ..., N\}$, we generate a corresponding 3D scene S. Throughout this paper we will utilize the notation $n_i \in \mathcal{N}$ to refer to nodes more generally. We represent the 3D scene $S = (\mathcal{B}, \mathcal{S})$ as a set of per-object bounding boxes $\mathcal{B} = \{b_0, ..., b_N\}$ and shapes $\mathcal{S} = \{s_0, ..., s_N\}$. Inspired by [20] on layout generation for image synthesis, we base our model on a variational scene graph Auto-Encoder. However, whereas [20] relies on shape retrieval, we jointly learn layouts and shapes via a shared latent embedding, as these are two inherently cohesive tasks strongly supporting each other. Moreover, we enable scene manipulation in the *same* learned model, using the scene graph as interface. In particular, given a scene together with its scene graph, changes can be applied to the scene, by interacting with the graph, such as adding new nodes or changing relationships. We do not need to learn object removal as this can be easily achieved by dismissing the corresponding box and shape for the given node.

The overall architecture is demonstrated in Figure 2. We first process scene graphs through a layout $\mathcal{E}_{\text{layout}}$ and shape $\mathcal{E}_{\text{shape}}$ encoder, section 4.2. We then employ a shared encoder $\mathcal{E}_{\text{shared}}$ which combines features from $\mathcal{E}_{\text{layout}}$ and $\mathcal{E}_{\text{shape}}$, section 4.3. This shared embedding is further fed to a shape $\mathcal{D}_{\text{shape}}$ and layout $\mathcal{D}_{\text{layout}}$ decoder to obtain the final scene. Finally, we use a modification network $\mathcal{T}$ (section 4.5) to enable the model the incorporation of changes in the scene while preserving the unchanged parts.

4.1. Graph Convolutional Network

At the heart of each building block in our model lies a Graph Convolutional Network (GCN) with residual layers [15], which enables information flow between the connected objects of the graph. Each layer l_g of the GCN operates on directed relationships triplets (*out – p – in*) and consists of three steps. First, each triplet ij is fed in a Multi-Layer Perceptron (MLP) $g_1(\cdot)$ for message passing

$$(\psi_{out,ij}^{(l_g)}, \phi_{p,ij}^{(l_g+1)}, \psi_{in,ij}^{(l_g)}) = g_1(\phi_{out,ij}^{(l_g)}, \phi_{p,ij}^{(l_g)}, \phi_{in,ij}^{(l_g)}). \quad (1)$$

Second, the aggregation step combines the information coming from all the edges of each node:

$$\rho_i^{(l_g)} = \frac{1_g}{M_i}\Big(\sum_{j \in \mathcal{R}_{out}} \psi_{out,ij}^{(l_g)} + \sum_{j \in \mathcal{R}_{in}} \psi_{in,ji}^{(l_g)}\Big) \quad (2)$$

where M_i is the number of edges for node i, and $\mathcal{R}_{out}$, $\mathcal{R}_{in}$ are the set of edges of the node as out(in)-bound objects. The resulting feature is fed to a final update MLP $g_2(\cdot)$

$$\phi_i^{(l_g+1)} = \phi_i^{(l_g)} + g_2(\rho_i^{(l_g)}). \quad (3)$$

4.2. Encoding a 3D Scene

We respectively harness two parallel Graph Convolutional encoders $\mathcal{E}_{\text{layout}}$, and $\mathcal{E}_{\text{shape}}$, for layout and shapes. The layout encoder $\mathcal{E}_{\text{layout}}$ is a GCN that takes the *extended* graph G_b, where nodes $n_i = (o_i, b_i)$ are enriched with the set of 3D boxes b for each object, and generates an output feature $f_{b,i}$ for each node n_i with $f_b = \mathcal{E}_{layout}(G_b)$.

Though it is possible to sample shapes independently from the scene graph, it can lead to inconsistent configurations. For instance, we would expect an office chair to co-occur with a desk. As a consequence, we propose to leverage another GCN to infer consistent scene setups. While a loss directly on the bounding boxes works well, similarly learning a GCN Auto-Encoder on shapes, *e.g.* point clouds, is a much more difficult task due to its uncontinuous output space. To circumvent this issue, we thus propose to instead learn how to generate shapes using a latent canonical shape space. This canonical shape space can be realized by various generative models having an encoder $\mathcal{E}_{\text{gen}}(\cdot)$

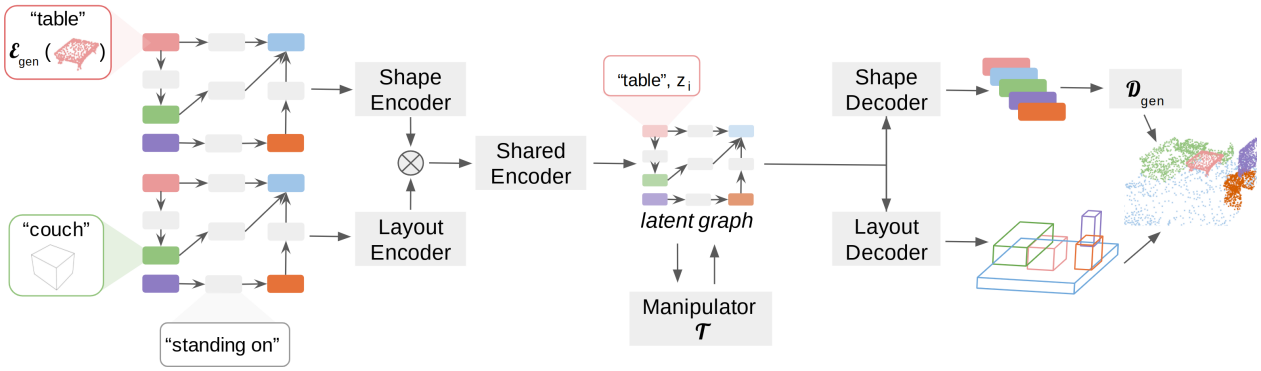

Figure 2. **Graph-to-3D pipeline**. Given a scene graph we generate a set of bounding boxes and object shapes. We employ a graph-based variational Auto-Encoder with two parallel GCN encoders sharing latent box and shape information through a shared encoder module. Given a sample from the learned underlying distribution the final 3D scene is obtained via combining the predictions from individual GCN decoders for 3D boxes and shapes. We further use a GCN manipulator for on the fly incorporation of user modifications to the scene graph.

and decoder $\mathcal{D}_{\text{gen}}(\cdot)$, *e.g.* by means of training an Auto-Encoder/Decoder [8, 26]. We create the extended graph G_s with nodes $n_i = (o_i, e_i^s)$, where $e_i^s = \mathcal{E}_{\text{gen}}(s_i)$. This formulation makes Graph-to-3D agnostic to the chosen shape representation. In our experiments, we demonstrate results with AtlasNet [8] and DeepSDF [26] as generative models. Please refer to the supplement for more details on AtlasNet and DeepSDF. Also here, we employ a GCN as shape encoder $\mathcal{E}_{\text{shape}}$, which we feed with G_s to obtain per node shape features $f_s = \mathcal{E}_{shape}(G_s)$.

4.3. Shape and Layout Communication

As layout and shape prediction are related tasks, we want to encourage communication between both branches. Therefore, we introduce a shared encoder $\mathcal{E}_{\text{shared}}$, which takes the concatenated output features of each encoder and computes a shared feature $f_{\text{shared}} = \mathcal{E}_{\text{shared}}(f_{bs}, \mathcal{R})$ with $f_{bs} = \{f_{b,i} \oplus f_{s,i} \,|\, i \in (1, ..., N)\}$. Further, we feed the shared features to an MLP network to compute the shared posterior distribution (μ, σ) under a Gaussian prior. We sample z_i from this distribution and feed the result to the associated layout and shape decoders. Since sampling is not differentiable, we apply the commonly used re-parameterization trick at training time to obtain z_i.

4.4. Decoding the 3D Scene

The layout decoder $\mathcal{D}_{\text{layout}}$ is again a GCN having the same structure as the encoders. The last GCN layer is followed by two MLP branches, which predict box extents and location $b_{\text{-}\alpha,i}$ separately from angle α_i. $\mathcal{D}_{\text{layout}}$ is fed with a set of sampled latent vectors z, one for each node, within the learned distribution as well as the semantic scene graph G. It then generates the corresponding object 3D boxes $(\hat{b}_{\text{-}\alpha}, \hat{\alpha}) = \mathcal{D}_{\text{layout}}(z, \mathcal{O}, \mathcal{R})$. The shape decoder $\mathcal{D}_{\text{shape}}$ follows a similar structure as $\mathcal{D}_{\text{layout}}$, with the difference that the GCN is followed by a single MLP producing the final shape encodings $\hat{e}^s = \mathcal{D}_{\text{shape}}(z, \mathcal{O}, \mathcal{R})$.

To obtain the final 3D scene, each object shape encoding is decoded into the respective shape $\hat{s}_i = \mathcal{D}_{gen}(\hat{e}_i^s)$. Each shape $\hat{s}_i$ is then transformed from canonical pose to scene coordinates, using the obtained bounding box $\hat{b}_i$.

4.5. Scene Graph Interaction

To enable scene manipulation that is aware of the current scene, we extend our model with another GCN $\mathcal{T}$, directly operating on the shared latent graph $G_l = (z, \mathcal{O}, \mathcal{R})$ as obtained from the encoders. First, we augment $G_l = (\hat{z}, \widehat{\mathcal{O}}, \widehat{\mathcal{R}})$ with changes. Thereby, $\widehat{\mathcal{O}}$ is composed of the original nodes $\mathcal{O}$ together with the new nodes $\mathcal{O}'$ being added to the graph. Similarly, $\widehat{\mathcal{R}}$ consists of the original edges $\mathcal{R}$ together with the new out-going and in-going edges of $\mathcal{R}'$. Additionally, some edges of $\widehat{\mathcal{R}}$ are modified according to the input from the user. Finally, since we do not have any corresponding latent representations for $\mathcal{O}'$, we instead pad z_i' with zeros to compute $\hat{z}_i$. Note that there can be infinitely possible outputs reflecting a given change. To capture this continuous output space, we concatenate $\hat{z}_i$ with samples z_i^n from a normal distribution having zero mean and unit standard deviation, if the node has been part of a manipulation, otherwise we concatenate zeros. Then, the $\mathcal{T}$ network gives a transformed latent as $z_{\mathcal{T}} = \mathcal{T}(\hat{z} \oplus \hat{z}^n, \widehat{\mathcal{O}}, \widehat{\mathcal{R}})$, as illustrated in Figure 3. Afterwards, the predicted latents for the affected nodes are plugged back into the original latent scene graph G_l. Finally, we feed the changed latent graph to the respective decoders to generate the updated scene, according to the changed scene graph. During inference, a user can directly make changes in the nodes and edges of a graph. At training time, we simulate the user input by creating a copy of the real graph exhibiting random augmentations, such as

node addition, relationship label corruption, or alternatively, leave the scene unchanged.

4.6. Training Objectives

The loss for training Graph-to-3D on the unchanged nodes, *i.e.* generative mode and unchanged parts during manipulation, is composed of a reconstruction term

$$\mathcal{L}_{\mathrm{r}} = \frac{1}{N}\sum_{i=1}^{N}(||\hat{b}_{\text{-}\alpha,i} - b_{\text{-}\alpha,i}||_1 + CE(\hat{\alpha}_i, \alpha_i) + ||\hat{e}_i^s - e_i^s||_1) \tag{4}$$

and a Kullback-Leibler divergence term

$$\mathcal{L}_{KL} = D_{KL}(\mathcal{E}(z|G, \mathcal{B}, e^s)|p(z|G)), \tag{5}$$

with $p(\cdot)$ denoting the Gaussian prior distribution and $\mathcal{E}(\cdot)$ being the complete encoding network. CE represents cross-entropy used to classify the angles, discretized in 24 classes.

4.6.1 Self-supervised Learning for Modifications

To train Graph-to-3D with changes, one requires appropriate pairs of scenes, *i.e.* before and after interaction. Unfortunately, recording such data is very expensive and time consuming. Furthermore, directly supervising the changed nodes with an L_1 loss is not an appropriate modeling for the one-to-many mapping of each relationship. Therefore, we propose the use of a novel relationship discriminator D_{box}, which can directly learn to interpret relationships and layouts from data, and ensure that the occasional relationship changes or node additions are correctly reflected in the 3D scene. We feed D_{box} with two boxes, class labels, and their relationship. D_{box} is then trained to enforce that the generated box will be following the semantic constraints from the relationship. To this end, we feed the discriminator with either real compositions or generated (fake) compositions, *i.e.* boxes after modification. D_{box} is then optimized such that it learns to distinguish between real and fake setups, whereas the generator tries to fool the discriminator by producing correct compositions under manipulations. The loss follows [7] and optimizes the following GAN objective

$$\begin{aligned}\mathcal{L}_{\mathrm{D,b}} = \min_G \max_D [\sum_{(i,j)\in\mathcal{R}'} \mathbb{E}_{o_i,o_j,r_{ij},b_i,b_j}[log D_{box}(o_i, o_j, r_{ij}, b_i, b_j)] \\ + \mathbb{E}_{o_i,o_j,r_{ij}}[log(1 - D_{box}(o_i, o_j, r_{ij}, \hat{b}_i, \hat{b}_j))]].\end{aligned} \tag{6}$$

Notice that this discriminator loss is applied to all edges that contain a change.

With a similar motivation, we adopt an auxiliary discriminator [25] for the changed shapes, which in addition to the GAN loss, leverages a classification loss $\mathcal{L}_{aux}$ according to

$$\begin{aligned}\mathcal{L}_{\mathrm{D,s}} = \mathcal{L}_{\mathrm{aux}} + \min_G \max_D [\sum_{i=1}^{N} \mathbb{E}_{o_i,e_i^s}[log D_{shape}(e_i^s)] + \\ \mathbb{E}_{o_i}[log(1 - D_{shape}(\hat{e}_i^s))]].\end{aligned} \tag{7}$$

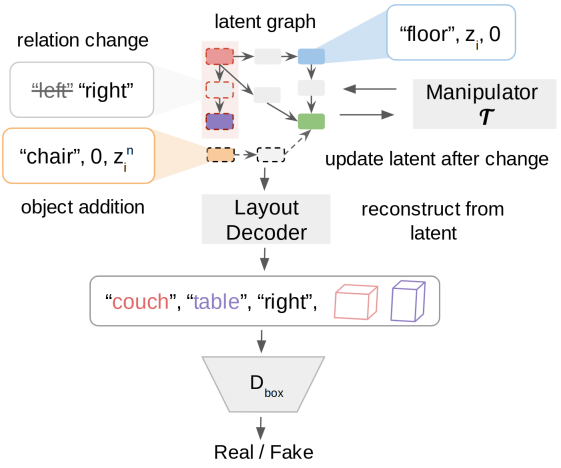

Figure 3. **Modifying scene graphs.** Given a scene graph we make changes in the nodes (object addition) or edges (relation change). Network $\mathcal{T}$ updates the latent graph accordingly. All edges that contain a change are passed to a relationship discriminator to encourage box prediction constrained on the node and edge labels.

Thereby, in addition to the real/fake decision, D_{shape} predicts the class of the given latent shape encoding to encourage that the generated objects represent their true class, *i.e.* $\mathcal{L}_{\mathrm{aux}}$ leverages the cross-entropy loss between the true o_i class and the predicted class from D_{shape}. Therefore, the discriminator can learn the boundary of the underlying shape distribution and ensure that the reconstructed shape stems from this distribution.

To summarize, our final loss becomes

$$\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{r}} + \lambda_{KL}\mathcal{L}_{\mathrm{KL}} + \lambda_{D,b}\mathcal{L}_{\mathrm{D,b}} + \lambda_{D,s}\mathcal{L}_{\mathrm{D,s}} \tag{8}$$

where the λs refer to the respective loss weights. We refer to the supplementary material for implementation details.

5. Results

In this section we describe the evaluation we used to assess the performance of the proposed approach in terms of plausible layout and shape generation that meets the constraints imposed by the input scene graph.

5.1. Evaluation protocol

We evaluate our method on the official splits of 3DSSG dataset [31], with 160 object classes and 26 relationship classes. Since we expect multiple possible results for the same input, typical metrics, such as L_1/L_2 norm or Chamfer loss are not suitable, due to the strict comparison between the predictions and the ground truth. Following [20] we rely on geometric constraints to measure if the input relationships are correctly reflected in the generated layouts. We test the constraint metric on each pair of the predicted boxes that are connected with the following relationships:

Method	Shape Representation	left / right	front / behind	smaller / larger	lower / higher	same	total
3D-SLN [20]	–	0.74	0.69	0.77	0.85	**1.00**	0.81
Progressive	–	0.75	0.66	0.74	0.83	0.98	0.79
Graph-to-Box	–	0.82	0.78	0.90	0.95	**1.00**	0.89
Graph-to-3D	AtlasNet [8]	**0.85**	0.79	0.96	0.96	**1.00**	0.91
Graph-to-3D	DeepSDF [26]	0.81	**0.81**	**0.99**	**0.98**	**1.00**	**0.92**

Table 1. Scene graph constrains on the **generation** task (higher is better). The total accuracy is computed as mean over the individual edge class accuracy to minimize class imbalance bias.

Method	Shape Representation	mode	left / right	front / behind	smaller / larger	lower / higher	same	total
3D-SLN [20]	–	change	0.62	0.62	0.66	0.67	0.99	0.71
Progressive	–	change	**0.81**	**0.77**	0.76	**0.84**	**1.00**	**0.84**
Graph-to-Box	–	change	0.65	0.66	0.73	0.74	0.98	0.75
Graph-to-3D w/o $\mathcal{T}$	AtlasNet [8]	change	0.64	0.66	0.71	0.78	0.96	0.75
Graph-to-3D	AtlasNet [8]	change	0.73	0.67	**0.82**	0.79	**1.00**	0.80
Graph-to-3D w/o $\mathcal{T}$	DeepSDF [26]	change	0.71	0.71	0.80	0.79	0.99	0.80
Graph-to-3D	DeepSDF [26]	change	0.73	0.71	**0.82**	0.79	**1.00**	0.81
3D-SLN [20]	–	addition	0.62	0.63	0.78	0.76	0.91	0.74
Progressive	–	addition	**0.91**	**0.88**	0.79	**0.96**	**1.00**	**0.91**
Graph-to-Box	–	addition	0.63	0.61	0.93	0.80	0.86	0.76
Graph-to-3D w/o $\mathcal{T}$	AtlasNet [8]	addition	0.64	0.62	0.85	0.84	**1.00**	0.79
Graph-to-3D	AtlasNet [8]	addition	0.65	0.71	0.96	0.89	**1.00**	0.84
Graph-to-3D w/o $\mathcal{T}$	DeepSDF [26]	addition	0.70	0.73	0.85	0.88	0.97	0.82
Graph-to-3D	DeepSDF [26]	addition	0.69	0.73	**1.00**	0.91	0.97	0.86

Table 2. Scene graph constraints on the **manipulation** task (higher is better). The total accuracy is computed as mean over the individual edge class accuracy to minimize class imbalance bias. Top: Relationship change mode. Bottom: Node addition mode.

Method	Shape Model	Shape Representation	Generation Size	Generation Location	Generation Angle	Generation Shape	Manipulation Size	Manipulation Location	Manipulation Angle	Manipulation Shape
3D-SLN [20]	Retrieval	3RScan Data	0.026	0.064	11.833	**0.088**	0.001	0.002	0.290	0.002
Progressive	–	3RScan Data	0.009	0.011	1.494	–	0.008	0.008	1.559	–
Graph-to-Box	Graph-to-Shape	AtlasNet [8]	0.009	0.024	1.869	0.000	0.007	0.019	2.920	0.000
Graph-to-3D		AtlasNet [8]	**0.097**	**0.497**	**20.532**	0.005	**0.037**	**0.061**	**14.177**	0.007
Graph-to-Box	Graph-to-Shape	DeepSDF [26]	0.009	0.024	1.895	0.011	0.005	0.019	3.391	0.014
Graph-to-3D		DeepSDF [26]	0.091	0.485	19.203	0.015	0.015	0.035	9.364	**0.016**

Table 3. Comparison on diversity results (std) on the generation (left) and manipulation tasks (right), computed as standard deviation over location and size in meters and angles in degrees. For shape we report the average chamfer distance between consecutive generations.

left, right, front, behind, smaller, larger, lower, higher and same (*c.f*. supplementary material for more details).

As a way to quantitatively evaluate the generated scenes and shapes, we perform a cycle-consistency experiment. Given the generated shapes from our models, we predict the scene graph, using the state-of-the-art scene graph prediction network (SGPN) from [31]. We then compare the ground truth scene graphs (*i.e*. input to our models) against the predicted graphs from SGPN. We base this comparison on the standard top-k recall metric for objects, predicates and relationship triplets from [31] (see supplement). This is motivated by the expectation that plausible scenes should result to the same graph as the input graph. Similar metrics have been utilized for image generation from semantics [34], using the inferred semantics from the generated image. In addition, in the supplement we report a user study to assess the global correctness and style fitness.

5.2. Baselines

3D-SLN With the unavailability of SunCG, we train [20] on 3DSSG using their official code repository. As we do not focus on images, we omit the rendering component. To obtain shapes for 3D-SLN, we follow their retrieval approach, in which for every $\hat{b}_i$ we retrieve from 3RScan the object shape from the same class, with the highest similarity.

Progressive Generation A model which naturally supports 3D generation and manipulation would be a progressive (auto-regressive) approach, as also explored in [32] for room planning. At each step a GCN (same as $\mathcal{D}_{layout}$) receives the current scene, together with a new node n_a to

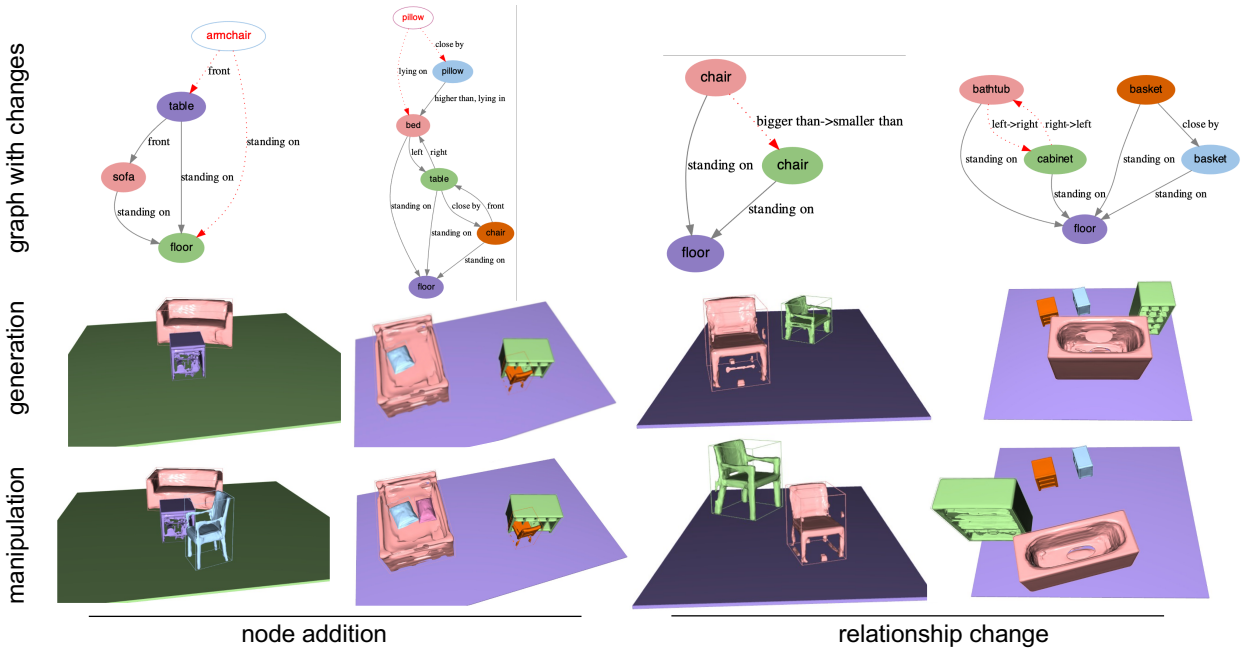

Figure 4. **Qualitative results** of Graph-to-3D (DeepSDF encoding) on 3D scene generation (middle) and manipulation (bottom), starting from a scene graph (top). Dashed lines reflect new/changed relationship, while empty nodes indicate added objects.

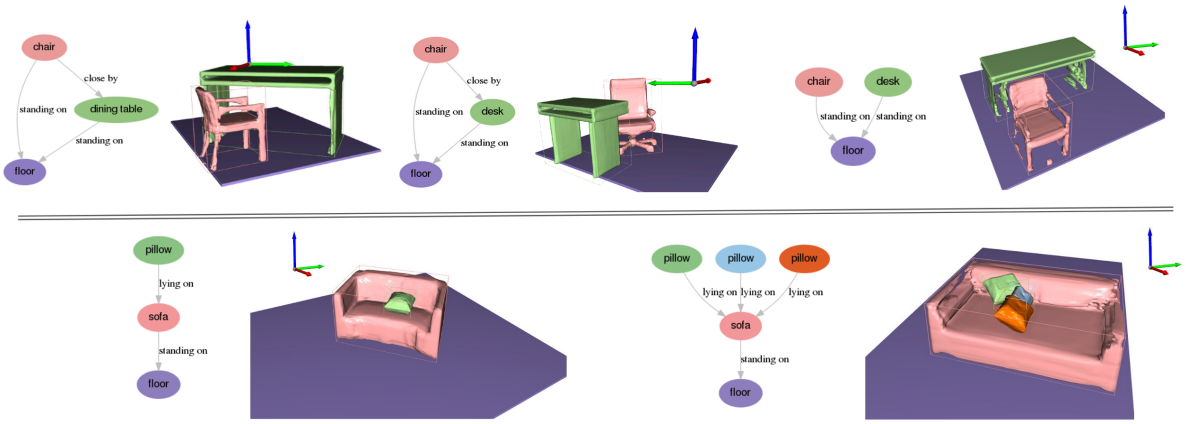

Figure 5. **Effect of scene context in scene generation.** *Top:* Connection to a desk makes a chair look like an office chair. *Bottom:* The number of pillows lying on a sofa affects its size and style.

be added. We refer the reader to the supplement for more details on the progressive baseline.

Ablations To ablate the relevance of using a GCN for the shape generation, we leverage a variational autoencoder directly based on AtlasNet, without awareness of the neighbouring objects. We provide more details in the supplement. Further, we ablate the sharing of layout and shape, by training a model with separate GCN-VAEs for shape (Graph-to-Shape) and layout (Graph-to-Box), which follow the same architecture choices, except $\mathcal{E}_{\text{shared}}$. We also run our method without modification network $\mathcal{T}$.

5.3. Layout evaluation

Table 1 reports the constrain accuracy metric on the generative task. We observe that Graph-to-3D outperforms the baselines as well as the variant decoupled layout and shape Graph-to-box on all metrics. Table 2 evaluates the constraint accuracy metric on the manipulation task. We report the node addition experiment and the relationship change experiment separately. We observe that the progressive model performs best for node addition (Table 2, bottom), while ours is fairly comparable for changes. This is natural as the progressive model is explicitly trained for addition.

Layout Model	Shape Model	Shape Representation	Recall Objects			Recall Predicate			Recall Triplets		
			Top 1	Top 5	Top 10	Top 1	Top 3	Top 5	Top 1	Top 50	Top 100
3D-SLN [20]	Retrieval	3RScan Data	**0.56**	0.81	0.88	0.50	**0.82**	0.86	0.15	0.57	0.82
Progressive	Retrieval		0.35	0.66	0.79	0.41	0.70	0.82	0.09	0.40	0.70
Graph-to-Box	AtlasNet VAE	AtlasNet [8]	0.41	0.74	0.83	0.57	0.80	0.88	0.08	0.46	0.77
‡Graph-to-Box	‡Graph-to-Shape		0.39	0.68	0.77	0.55	0.79	0.88	0.05	0.35	0.69
Graph-to-Box	Graph-to-Shape		0.51	0.81	0.86	0.57	0.80	0.88	**0.23**	0.63	0.84
Graph-to-3D			0.54	**0.84**	**0.90**	**0.60**	**0.82**	**0.90**	0.21	**0.65**	**0.85**
Graph-to-Box	Graph-to-Shape	DeepSDF [26]	0.47	0.74	0.83	0.57	0.80	0.87	0.14	0.57	0.81
Graph-to-3D			0.51	0.80	0.88	0.58	0.80	0.89	0.19	0.59	0.83
3RScan data			0.53	0.82	0.90	0.75	0.93	0.98	0.18	0.61	0.83

Table 4. Scene graph prediction accuracy on 3DSSG, using the SGPN model from [31], measured as top-k recall for object, predicate and triplet prediction (higher is better). ‡Model trained with non-canonical objects, exhibiting significantly worse results.

The models using $\mathcal{T}$ perform better than 3D-SLN or the respective model without $\mathcal{T}$ on the manipulation task, which is expected since these approaches explicitly model an architecture that supports such changes.

In addition, we measure diversity as standard deviation among 10 samples that are generated under the same input. We compute this metric separately over each bounding box parameter, and compute the mean over size, translation in meters and angle in degrees. To measure shape diversity, we report the average chamfer distance between these 10 samples. Results are shown in Table 3. The progressive generation shows the lowest values in diversity for both generation and modification. The other models, on the other hand, exhibit more interpretable diversity results, with larger values for position than for object size. Nevertheless, both shared models come out superior for diversity in layout. As for shape, the two shared models are again superior for manipulation, yet, we perform a bit worse for generation.

5.4. Shape evaluation

Figure 4 shows qualitative results from Graph-to-3D. We first sample a scene conditioned on a scene graph (top), and then apply a change in the graph which is then reflected in the scene. The model understands diverse relationships such as support (lying on), proximity (left, front) and comparison (bigger than). For instance, the model is able to place a pillow on the bed, or change chair sizes in accordance with the edge label. In addition, the object shapes and sizes well represent the class categories in the input graph.

In Figure 5 we illustrate the effect of scene context on shape generation. For instance, chairs tend to have an office style (middle) while connected to a desk, and a more standard style when connected to a dining table (left), or when there is no explicit connection to the desk (right). In addition, having many pillows on a sofa contributes to its style and larger size. These patterns learned from data show another interesting advantage of the proposed graph-driven approach based on learned shapes.

The quantitative results on 3D shapes and complete 3D scenes are shown on Table 4. The object and predicate recall metric is mostly related to namely shape generation and layout generation quality. The triplet recall measures the combined influence of all components. The table compares different shape models, such as AtlasNet VAE, Graph-to-Box/Shape and our shared model Graph-to-3D. For reference we present the scene graph prediction results on the ground truth scenes (3RScan data). As expected, the latter has the highest accuracy in predicate prediction. Interestingly, on metrics that rely on shapes, it is comparable to our Graph-to-3D model. Models based on a GCN for shape generation outperform the simple AtlasNet VAE, that does not consider inter-object relationships. Comparing the shared and disentangled models we observe that there is a consistent performance gain for both the layout generation as well as shape, meaning that the two tasks benefit from the joint layout and shape learning. Finally, we also run our baseline Graph-to-Box/Shape using shapes in non-canonical pose. The performance of this model drops significantly, demonstrating the relevance of our annotations.

6. Conclusion

In this work, we propose Graph-to-3D a novel model for end-to-end 3D scene generation and interaction using scene graphs, and explored the advantages of joint learning of shape and layout. We show that the same model can be trained with different shape representations, including point clouds and implicit functions (SDFs). Our evaluations on quality, semantic constrains and diversity show compelling results on both tasks. Future work will be dedicated to generating objects textures, combined with scene graph attributes that describe visual properties.

7. Acknowledgements

This research work was supported by the Deutsche Forschungsgemeinschaft (DFG), project 381855581. We thank all the participants of the user study.

References

[1] 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 *International Conference on Computer Vision (ICCV)*, 2019.

[2] Oron Ashual and Lior Wolf. Specifying object attributes and relations in interactive scene generation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4561–4569, 2019.

[3] 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.

[4] Helisa Dhamo, Azade Farshad, Iro Laina, Nassir Navab, Gregory D. Hager, Federico Tombari, and Christian Rupprecht. Semantic Image Manipulation Using Scene Graphs. In *Computer Vision and Pattern Recognition (CVPR)*, 2020.

[5] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[6] Matthew Fisher, Manolis Savva, and Pat Hanrahan. Characterizing structural relationships in scenes using graph kernels. *ACM Trans. Graph*, 2011.

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*. 2014.

[8] 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.

[9] Roei Herzig, Moshiko Raboh, Gal Chechik, Jonathan Berant, and Amir Globerson. Mapping Images to Scene Graphs with Permutation-Invariant Structured Prediction. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

[10] Chenfanfu Jiang, Siyuan Qi, Yixin Zhu, Siyuan Huang, Jenny Lin, Lap-Fai Yu, Demetri Terzopoulos, and Song-Chun Zhu. Configurable 3D Scene Synthesis and 2D Image Rendering with Per-pixel Ground Truth Using Stochastic Grammars. *International Journal of Computer Vision (IJCV)*, 2018.

[11] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image Generation from Scene Graphs. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[12] J. Johnson, R. Krishna, M. Stark, L. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei. Image retrieval using scene graphs. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[13] Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. Layoutvae: Stochastic scene layout generation from a label set. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[14] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalanditis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations. *International Journal of Computer Vision (IJCV)*, 2017.

[15] Guohao Li, Matthias Mller, Ali Thabet, and Bernard Ghanem. DeepGCNs: Can GCNs Go as Deep as CNNs? In *International Conference on Computer Vision (ICCV)*, 2019.

[16] 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 Transactions on Graphics (TOG)*, 2018.

[17] 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 *European Conference on Computer Vision (ECCV)*, 2018.

[18] Yikang Li, Wanli Ouyang, Bolei Zhou, Kun Wang, and Xiaogang Wang. Scene Graph Generation from Objects, Phrases and Region Captions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[19] Tianqiang Liu, Siddhartha Chaudhuri, Vladimir Kim, Qixing Huang, Niloy Mitra, and Thomas Funkhouser. Creating Consistent Scene Graphs Using a Probabilistic Grammar. *ACM Transactions on Graphics (TOG)*, 2014.

[20] 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.

[21] Rui Ma, Akshay Gadi Patil, Matthew Fisher, Manyi Li, Sren Pirk, Binh-Son Hua, Sai-Kit Yeung, Xin Tong, Leonidas Guibas, and Hao Zhang. Language-Driven Synthesis of 3D Scenes from Scene Databases. In *SIGGRAPH Asia, Technical Papers*, 2018.

[22] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas Guibas. StructureNet: Hierarchical Graph Networks for 3D Shape Generation. *ACM Transactions on Graphics (TOG)*, 2019.

[23] Alejandro Newell and Jia Deng. Pixels to graphs by associative embedding. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

[24] 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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[25] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *ICML*, pages 2642–2651, 2017.

[26] 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.

[27] Mengshi Qi, Weijian Li, Zhengyuan Yang, Yunhong Wang, and Jiebo Luo. Attentive Relational Networks for Mapping

Images to Scene Graphs. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[28] Yifei Shi, Angel Xuan Chang, , Zhelun Wu, Manolis Savva, and Kai Xu. Hierarchy Denoising Recursive Autoencoders for 3D Scene Layout Prediction. In *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[29] Shubham Tulsiani, Saurabh Gupta, David Fouhey, Alexei A. Efros, and Jitendra Malik. Factoring shape, pose, and layout from the 2d image of a 3d scene. In *CVPR*, 2018.

[30] 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.

[31] 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.

[32] Kai Wang, Y. Lin, Ben Weissmann, M. Savva, Angel X. Chang, and D. Ritchie. Planit: planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Trans. Graph.*, 38:132:1–132:15, 2019.

[33] Kai Wang, Manolis Savva, Angel X Chang, and Daniel Ritchie. Deep convolutional priors for indoor scene synthesis. *ACM Transactions on Graphics (TOG)*, 37(4):70, 2018.

[34] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[35] Shun-Cheng Wu, Johanna Wald, Keisuke Tateno, Nassir Navab, and Federico Tombari. Scenegraphfusion: Incremental 3d scene graph prediction from rgb-d sequences. In *Proceedings IEEE Computer Vision and Pattern Recognition (CVPR)*, 2021.

[36] Danfei Xu, Yuke Zhu, Christopher Choy, and Li Fei-Fei. Scene Graph Generation by Iterative Message Passing. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[37] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph R-CNN for Scene Graph Generation. In *European Conference on Computer Vision (ECCV)*, 2018.

[38] Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. Neural Motifs: Scene Graph Parsing with Global Context. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[39] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[40] Yibiao Zhao and Song chun Zhu. Image Parsing with Stochastic Scene Grammar. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2011.

[41] Yang Zhou, Zachary While, and Evangelos Kalogerakis. Scenegraphnet: Neural message passing for 3d indoor scene augmentation. In *IEEE Conference on Computer Vision (ICCV)*, 2019.