·Article·

3D scene graph prediction from point clouds

Fanfan WU, Feihu YAN*, Weimin SHI, Zhong ZHOU*

State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing 100191*, China*

*** Corresponding author,** yanfeihu@buaa.edu.cn, zz@buaa.edu.cn

Fanfan WU and Feihu YAN contributed equally to this work and share the first authorship.

Received: 15 September 2021 **Accepted:** 5 December 2021

Supported by National Natural Science Foundation of China (61872024); National Key R&D Program of China under Grant (2018YFB2100603).

Citation: Fanfan WU, Feihu YAN, Weimin SHI, Zhong ZHOU. 3D scene graph prediction from point clouds. Virtual Reality & Intelligent Hardware, 2022, 4(1): 76—88
DOI: 10.1016/j.vrih.2022.01.005

Abstract **Background** In this study, we propose a novel 3D scene graph prediction approach for scene understanding from point clouds. **Methods** It can automatically organize the entities of a scene in a graph, where objects are nodes and their relationships are modeled as edges. More specifically, we employ the DGCNN to capture the features of objects and their relationships in the scene. A Graph Attention Network (GAT) is introduced to exploit latent features obtained from the initial estimation to further refine the object arrangement in the graph structure. A one loss function modified from cross entropy with a variable weight is proposed to solve the multi-category problem in the prediction of object and predicate. **Results** Experiments reveal that the proposed approach performs favorably against the state-of-the-art methods in terms of predicate classification and relationship prediction and achieves comparable performance on object classification prediction. **Conclusions** The 3D scene graph prediction approach can form an abstract description of the scene space from point clouds.

Keywords Scene understanding; 3D scene graph; Point cloud; DGCNN; GAT

1 Introduction

Scene understanding is a vital research topic as it is the core of many emerging technologies, spanning a wide field of application scenarios from self-driving cars to augmented and virtual reality. Owing to the availability of high-accuracy 3D modeling algorithms, such as simultaneous localization and mapping (SLAM)[1], structure from motion (SfM)[2], and multi-view stereo (MVS)[3], the computer vision community is experiencing a growing interest in scene understanding from 3D point clouds.

Unlike image-based scene understanding represented by semantic segmentation[4], 3D models preserve the complete 3D spatial layout and fine-grained geometry details of the scene, providing richer information for scene understanding. Owing to the irregular format, however, the extraction of local features from a point cloud is challenging. In previous studies, the point cloud was converted to volumetric grids by quantization[5, 6]. This method can easily train the convolutional network, but increase the computational complexity. In several recent studies[7–9], the direct processing of point sets without converting them to other formats has been proposed. PointNet[7] is the pioneering effort that aggregates all learned individual point features into a global point cloud signature. However, to maintain the permutation invariance, PointNet[7] and its extension[8] cannot obtain local features because they treat points independently at the local scale. Wang et al. proposed the Dynamic Graph Convolutional Neural Network (DGCNN), which captures local geometric structure while maintaining permutation invariance and shows efficient performance in many 3D understanding applications including object classification and semantic segmentation[9].

To gain a comprehensive understanding of the objects and their relationships, the use of a concise and clear presentation method for organizing information is necessary. Scene graphs are popular computer graphics models for describing and arranging representations of complex scenes. Typically, in a scene graph, the nodes represent scene entities, and the edges represent relationships between two nodes. In computer vision, 2D scene graph has been widely used to abstract the content of 2D images[10,11]. Recently, 3D scene graphs that describe 3D scenes have gained more popularity[12,13].

In this work, we propose a deep learning system to build a 3D scene graph from a point cloud. More specifically, given a class-agnostic 3D point cloud with instance segmentation, we construct a 3D scene graph, where nodes are abstract scene components and edges represent their relationships. In contrast to previous 3D scene graph methods, we exploit the standard DGCNN[9] to learn an initial estimation of the nodes and edges of the graph. Unlike in previous studies in which a Graph Convolutional Network (GCN)[14] was employed to process the acquired information, we employ a Graph Attention Network (GAT) [15] and introduce the attention mechanism in the network to predict the final scene graph. Extensive experiments are conducted on the 3DSSG dataset[13], and the experimental results show that our method can significantly improve the performance in object classification and relationship prediction.

The contributions of our proposed method can be summarized as follows:

(1) We propose a novel network that predicts a 3D scene graph from a 3D point cloud, where objects in the scene are abstracted as nodes and their relationships are modeled as edges.

(2) We propose a one loss function modified from cross entropy with a variable weight to solve the multi-category problem in the prediction of object and predicate.

The remainder of this paper is organized as follows. In Section 2, we review the advances in 3D scene understanding research. In Section 3, a deep learning method for building the 3D scene graph is proposed. In Section 4, the validity of the proposed method is verified through experiments. Finally, in Section 5, we summarize this work and discuss its further development.

2 Related work

2.1 Scene understanding

Scene understanding significantly impacts various applications to perceive, analyze, and interpret visual scenes. Initial work[16] was motivated by human visual perception of natural scenes and understanding of high-level scene structures. Owing to the development of deep learning, we have witnessed the success of applying the deep learning framework to scene understanding.

In the past decade, numerous studies on scene understanding based on 2D images have been conducted, including semantic segmentation[4], object detection[17,18], and monocular depth estimation[19–21], in which data-driven feature representations learned by deep neural networks have been shown to perform effectively for describing visual data.

With the improvement in hardware performance and the expansion of deep learning networks, the data for scene understanding are also extended to RGB-D[22,23] and point cloud structures[7–9]. Intuitively, it is highly challenging to handle the irregularity of point clouds directly. This approach was pioneered in PointNet[7], which operates on each point independently and then applies a symmetric function to accumulate features to achieve the invariance of point arrangement. The extension, PointNet++[8], considers the neighborhoods of points instead of operating on each independently, utilizing local features and improving the performance of the basic model. The DGCNN[9] is one of the promising extensions of PointNet[7], and it presents a novel operation, EdgeConv, to better capture local geometric features of point clouds while maintaining permutation invariance.

2.2 Scene graph

An essential task of scene understanding is to describe objects and their relationships, which are very suitable for the organization of a graph structure. The objects appearing in the scene are displayed as nodes and their relationships constitute edges in the scene graph, which contributes to scene understanding and interpretable reasoning. The scene graph can be widely used in computer vision and computer graphics tasks. Commonly, a preestablished scene graph is used to describe the scene structure[24–26], and the scene understanding relies on the calculation of the similarity between the actual image and the pictorial graph.

The scene graph has been chiefly used to abstract the content of 2D images as a sparse representation of image semantic information, where nodes represent entities in the image and edges represent their relationships. Further, 2D scene graphs have been widely used in various applications, including image retrieval[27], visual question-answering[28], and scene parsing[29,30].

Recently, 3D graph prediction methods have attracted considerable attention to scene understanding owing to the application prospects in robotics and computer vision. Armeni et al. innovatively proposed a hierarchical 3D scene graph structure that represents semantics, 3D space, and camera, where elements with certain attributes are nodes in the graph and edges are formed between them to denote relationships[31]. Rosinol et al. presented DSGs, extending this notion to represent dynamic scenes. Although this hierarchical structure contains richer information, the types of nodes are changeable, and the corresponding edges have more complex descriptions[32]. Another type of method[13,33] abandons multiple levels and only includes objects as nodes. This consistent data structure is more effective in describing a single indoor scene.

3 Method

In this section, we first explain the problem formally and then give a detailed description of our 3D scene graph system. As illustrated in Figure 1, based on the method presented by Wald et al.[13], the proposed system consists of two stages: the initial scene graph construction stage and the scene graph refinement stage. Two separate backbone networks are employed in the initial construction stage to extract independent object and relationship features. Then, in the refinement stage, a GAT is used to refine the initial predictions via the scene context information.

3.1 Problem definition

Given the input point cloud, P, and the class-agnostic instance segmentation, M, of a scene s, indicating that the point cloud has instance segmentation labels without specific semantic categories, our aim is to generate the corresponding scene graph, which is a graph topology structure composed of the category label of each instance and the label of the relationship between the categories.

The scene graph is defined as:

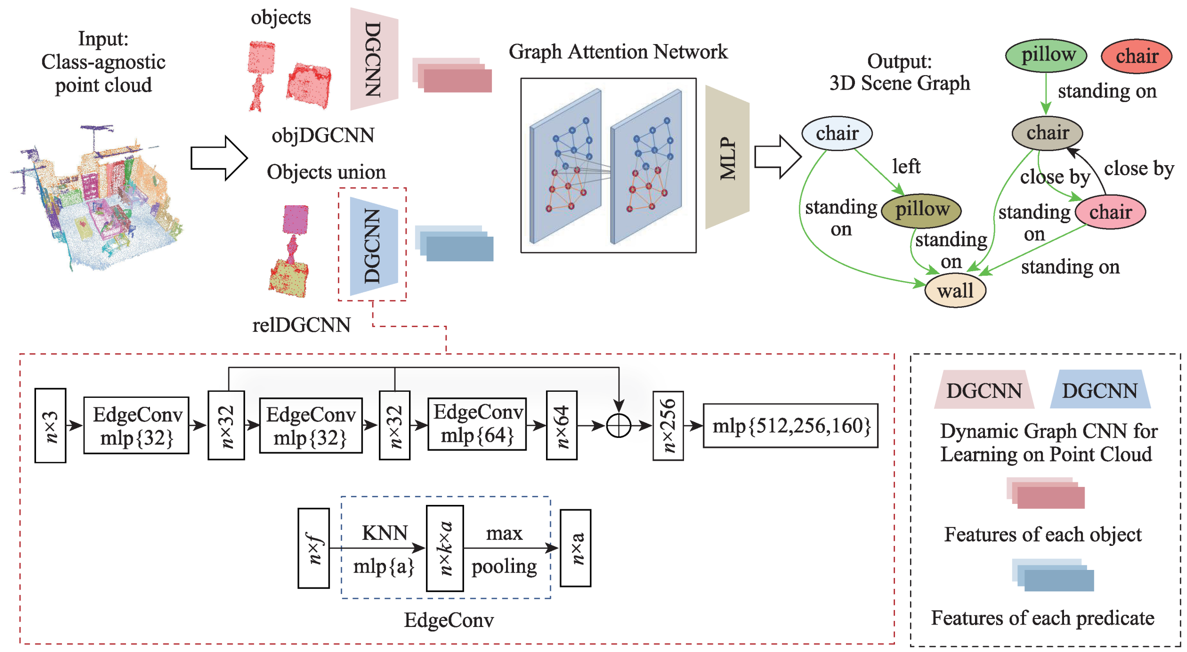

Figure 1 Scene graph prediction network using the DGCNN and graph attention network.

$$G = (O, E), \tag{1}$$

where nodes $O = \left\{o_1, \cdots, o_n\right\}$ denote the object set of the scene, and $E \subseteq O \times \mathcal{R} \times O$ depicts the edge set, which describes the relationship between two objects.

We use the relationship set provided by the 3DSSG dataset[13], including spatial/proximity relationships, support relations, and comparative relationships. Please refer to[13] for more details.

3.2 Initial scene graph construction

Similar to [13], our learning method is based on the common principle in scene graph prediction[10–11]. Thus, we need to extract visual features for every node and edge. We use two separate DGCNNs[9], named ObjDGCNN and RelDGCNN, to extract nodes and edges, respectively. The DGCNN is an efficient graph-based model capable of learning contextual features of point clouds. It dynamically constructs graphs at every layer conditioned on the feature space rather than in the Euclidean space, therefore, it can better capture high-level contextual information in the scene. Hence, it is very suitable for our scene graph prediction task.

More specifically, the DGCNN employs a module, Edgeconv, to collect features in the local region of the point cloud, where the relationship between the central point and its neighbors is considered.

In our work, the point set, P_i, of each instance is extracted and fed into the ObjDGCNN to capture the point-wise features. To predict the relation between two objects, i and j, point set P_{ij} is extracted for the pair of objects union and propagated to the RelDGCNN. The extracted features are arranged in the form of relationship triples (subject, predicate, and object). Given the disorder of the point cloud, the function that aggregates all point features should be symmetrical. Here, the max pooling layer is introduced to solve the problem. The transformation independence of the point cloud is solved by aligning all the inputs into a standard space before feature extraction.

3.3 Scene graph refinement

We employ a GAT[15] to process the acquired node and edge predictions to further refine the constructed scene graph. Compared with the fixed weight of the node feature update in the GCN[14], the self-attention mechanism is used to learn the weight of the node during the update stage.

For a single-layer graph network structure, the input is a series of node features, $h = \left\{\vec{h}_1, \cdots, \vec{h}_n\right\}, \vec{h}_l \in R^F$, where n is the number of nodes in the topological graph and F is the dimension of each node feature. After this layer, a series of new node features, $h' = \left\{\vec{h}'_1, \cdots, \vec{h}'_n\right\}, \vec{h}'_l \in R^{F'}$, can be obtained as the output. The feature dimension of the output is not necessarily consistent with the input. As part of network initialization, a shared linear transformation is applied to each node, and the input feature vector is mapped to the high-dimensional feature vector and propagated to the graph attention network using matrix $W \in R^{F'} \times R^F$. Then, we apply the self-attention mechanism on each node:

$$e_{ij} = a\left(W\vec{h}_l, W_h\right), \tag{2}$$

where a represents the attention score, that is, the importance coefficient of node j to node i.

To make the attention scores comparable between different nodes, the softmax function is used to normalize all attention scores:

$$\alpha_{ij} = \operatorname{softmax}\left(e_{ij}\right) = \frac{\exp\left(\operatorname{LeakyReLU}\left(\vec{a}^T\left[W\vec{h}_l | W\vec{h}_j\right]\right)\right)}{\sum_{k \in N_i} \exp\left(\operatorname{LeakyReLU}\left(\vec{a}^T\left[W\vec{h}_\iota | W\vec{h}_k\right]\right)\right)}, \tag{3}$$

where N_i means the neighborhood of node i.

The attention mechanism is a single-layer forward neural network in the realization of the network, and it is parameterized by a weight vector, $\vec{a} \in R^{2F'}$, and a LeakyReLU nonlinear function is applied to activate the output vector. Then, the normalized attention score is used to calculate the linear combination of feature vectors as the output feature of each node. For each message transfer layer, l, in the network, triplet t_{ij} needs to be passed as input to the first defined MLP, g_1:

$$\left(\psi^{(l)}_{s,ij}, \phi^{(l+1)}_{p,ij}, \psi^{(l)}_{o,ij}\right) = g_1\left(\phi^{(l)}_{s,ij}, \phi^{(l)}_{p,ij}, \phi^{(l)}_{o,ij}\right), \tag{4}$$

where ψ is the intermediate feature vector to be processed, s is the subject, p is the predicate, and o is the object.

Then, the intermediate features of the nodes need to be aggregated. At this stage, each node receives all the features of the nodes that have an association relationship with it. The attention score is introduced in the summation process to obtain each node:

$$\rho^{(l)}_i = \sigma\left(\sum_{j \in R_s} \alpha_{ij} W\psi^{(l)}_{s,ij} + \sum_{j \in R_o} \alpha_{ij} W\psi^{(l)}_{o,ij}\right), \tag{5}$$

where R_s and R_o are the sets of connection points when the current node, i, is used as the subject and object, respectively.

We pass the output result into another MLP, g_2. Moreover, to avoid the Laplacian smoothing that may exist in the graph structure, the residual connection is added to the final output vector to obtain the output feature of node i:

$$\phi^{(l+1)}_i = \phi^{(l)}_i + g_2\left(\rho^{(l)}_i\right). \tag{6}$$

To stabilize the self-attention learning process, a multi-head attention mechanism is used. In practice, the number of heads is set to four to learn the weight coefficients separately, that is, the network layers of four independent attention mechanisms are used to learn the weights, and then, the obtained aggregate feature vectors are connected. Finally, the features perform debasing dimension through a GAT layer, and this is used for subsequent modules. Two MLPs are used to predict the node and predicate classes.

3.4 Loss function

When training the model end-to-end, we follow[13] to use the object classification loss, $\mathcal{L}_{\text{obj}}$, with weight λ_{obj} and predicate classification loss $\mathcal{L}_{\text{pred}}$:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{obj}} \mathcal{L}_{\text{obj}} + \mathcal{L}_{\text{pred}}, \tag{7}$$

where $\mathcal{L}_{\text{obj}}$ is formulated using the cross entropy loss and λ_{obj} is set to 0.1 in practice. As there may be more than one type of predicate between objects, $\mathcal{L}_{\text{pred}}$ is formulated using per-class binary cross entropy, similar to the multiple binary classification problem.

Object prediction and predicate prediction can be regarded as two independent classification tasks. According to the observation of actual scene data, the classification task can be regarded as a single classification task or as a multi-classification task. It is a relatively straightforward idea to treat the classification task as a single classification task. Each point cloud object and the relationship between the objects have only one semantic label corresponding to it. The focal loss[34] is introduced to alleviate the imbalance between the classes in the data set. In this case, the loss function of object classification and predicate classification is:

$$\mathcal{L} = -\alpha\left(1 - p_t\right)^{\gamma} \log p_t, \tag{8}$$

where α is introduced to balance the importance of positive and negative samples and overcome the problem of uneven proportions of positive and negative samples. γ is used to adjust the speed at which the weight of simple samples decreases. When γ is set to 0, it is converted into a cross-entropy loss function. When γ increases, the adjustment impact on difficult and easy samples will also increase. The final loss function accounts for a relatively small proportion, while for difficult samples, its proportion in the loss function is larger.

There is a hierarchical relationship between object labels. More detailed label classification should not put the wrong labels in the prediction process. Furthermore, less attention should be paid to coarser object labeling, and the optimization direction of the network should be towards a finer division direction. Rather than a rougher label prediction. Therefore, the loss function under the multi-classification task can be designed as:

$$\mathcal{L}_{obj} = \begin{cases} -\alpha\left(1 - p_t\right)^{\gamma} \log p_t, & l_p \geq l_{gt} \\ -\alpha\beta^{\left(l_p - l_{gt}\right)}\left(1 - p_t\right)^{\gamma} \log p_t, & l_p < l_{gt} \end{cases}, \tag{9}$$

where α and γ are the hyperparameters in focal loss, l_p is defined as the predicted label level, l_{gt} is the label level of ground truth, and β is defined to measure the degree of reduction in the proportion of rough prediction results in the loss function. The lower β means that even if the rough prediction result is not wrong, it has little effect on the loss function. When the network obtains a more refined prediction result, because the correctness of the prediction result cannot be judged based on the true value given in the data set, the design of its loss function is consistent with the loss of the predicted correct result.

The predicate relationship should not be constrained by a single label in the labeling and prediction process. The predicate relationship description of a pair of instance objects can be corresponding to multiple labels. When the predicate relationship is regarded as a multi-classification task, because there is no mutual exclusion relationship between the tags, each category can be independently predicted. Each category is regarded as a binary classification task, that is, to determine whether the association relationship exists between the objects. The loss function is:

$$\mathcal{L}_{\text{pred}} = \sum_{i=1}^{c} -\alpha\left(1 - p_{t,i}\right)^{\gamma} t_i \log p_{t,i} - \alpha\left(1 - p_{t,i}\right)^{\gamma}\left(1 - t_i\right)\log\left(1 - p_{t,i}\right), \tag{10}$$

where t_i is the true value of the i-th category under the binary classification task, $p_{t,i}$ is the prediction confidence of the i-th category, and c is the total number of predicate label categories.

4 Experiments

In this section, we present extensive evaluations of the proposed method on the task of scene graph prediction. In all experiments, our networks are trained by Adam optimizer with a learning rate of 1e-4. We decrease it by 0.5 after finishing 30 and 40 epochs. The weight decay parameter is 1e-5. The training time is proportional to the total number of scenes and the number of epochs. The batch-size is up to two. As the total number of scenes is considerably large, the training time is sensible. It takes approximately two days to train the model based on a server with 2.4GHz CPU and GTX TITAN X GPU for 50 epochs.

4.1 Dataset

We use the 3DSSG dataset[13] for training and evaluation; it provides 1482 scene graphs with 48k object nodes and 544k edges. A ground-truth semantic scene graph is defined by a set of tuples between nodes and edges where nodes represent specific 3D object instances in a 3D scan. Nodes are defined by their semantics, a hierarchy of classes, and a set of attributes that describe the visual and physical appearance of the object instance and their affordances. The edges are the semantic relationships (predicates) between the nodes. To reduce the training time, each scan split is preprocessed through the following steps:

(1) A segment dictionary is built for the preprocessed point cloud, where the key is the index of the segment, and the value is all points belonging to this segment.

(2) The mapping relationship is established between objects and the point cloud.

(3) A farthest point sampling (FPS) algorithm is adopted to reduce the data burden of the network input. Then the point set of each object is normalized to solve the scalability problem.

(4) Relationship triplets between every two objects are established, and all points contained in the bounding box of the two objects are combined into a union.

(5) Similar to step (3), the related point clouds are sampled and normalized.

(6) All information including index, category, point number, triplets, etc., are stored in a dictionary.

4.2 Scene graph prediction

Based on[10,13], we use three metrics to evaluate the performance of scene graph prediction: the relationship triplet prediction, the object class prediction, and the predicate class prediction. As mentioned in Sec. 3.2, the relationship triplet prediction is jointly generated as an ordered list of (subject, predicate, object) triplets. Thus, we can obtain the confidence score for each triplet by multiplying each respective score and use the most confident one for evaluation against the ground truth. The object and predicate metrics are calculated directly with the respective classification scores. We adopt the top-n recall metric for accuracy evaluation.

As shown in Table 1, we compare our method with the current state-of-the-art approach proposed by Wald et al.[13]. The single model means that the classification task is regarded as a single classification task, that is, for each object and predicate prediction, there is only one correct label corresponding to it; the multi model means that the classification task is regarded as multiple binary classification tasks, that is, there may be multiple correct labels corresponding to each prediction. The multi model is more in line with the real situation. On the one hand, there is a hierarchical relationship between the categories of objects.

Table 1 Evaluation of the scene graph prediction performance on 3DSSG

Model	Method	Relationship prediction		Object class prediction		Predicate prediction	
		R@50	*R*@100	*R*@5	*R*@10	*R*@3	*R*@5
Single	Wald et al.[13]	0.5530	0.7971	0.4887	0.5428	0.8284	0.8731
	Ours (PointNet+GAT)	0.5973	0.8528	0.3624	0.4997	0.8586	0.8948
	Ours (DGCNN+GCN)	0.5428	0.8270	0.4383	0.5828	0.8486	0.8976
	Ours	0.5591	0.8500	0.4611	0.5949	0.8739	0.9197
Multi	Wald et al.[13]	0.4151	0.4285	0.5448	0.6764	0.7319	0.7334
	Ours (PointNet+GAT)	0.6075	0.7799	0.3379	0.4774	0.7828	0.7846
	Ours (DGCNN+GCN)	0.5627	0.8016	0.4727	0.6017	0.8094	0.7735
	Ours	0.6133	0.8707	0.5793	0.6954	0.8730	0.9246

Notes: Single/Multi means single/multi object and predicate class.

When an armchair is recognized as a chair, it can also be considered a correct prediction. On the other hand, the relationship between objects is not unique. Two objects may be the same type (same as), and also have a spatial neighbor relationship (close by). To verify the effectiveness of our method, we also conduct ablation experiments using PointNet[7] for feature extraction and the GCN[14] for scene graph structure regression, which have become two versions of our model, "PointNet+GAT" and "DGCNN+GAT."

Table 1 shows that in the single model our method and the other two versions outperform[13] in relation-related metrics while achieving comparable results on object classification metrics. The decline in the performance of object classification tasks may be due to the increase in network structure complexity, which increases the difficulty of network fitting. In the multi model, our method achieves the best results in all indicators. It can be seen that compared with the method proposed by Wald et al.[13], the results of the two versions of our method have improved in terms of the relationship prediction and the predicate prediction. Similarly, we can find that using the DGCNN can lead to a significant improvement in the object and predicate accuracy from the comparison between our method and the other two versions.

We also analyze and fully compare the complexity of our model with other existing methods. Specifically, we use parameter numbers (Params), giga floating point operations (GFLOPs) and number of layers as main metrics. The Params is a standard metric to evaluate the complexity of models. It reflects the number of parameters in this model. The GFLOPs reflects the number of operations of all convolutional layers in the model, and it is a metric to estimate the time complexity of models. To analyze the complexity of the model more comprehensively, we also calculated the number of layers containing parameters in the network.

As shown in Table 2, Params of the proposed method is 42.84% of that of Wald et al.[13], and the number of layers of our method is 27 lower than that of Wald et al.[13]. Two metrics, Params and number of layers, both show that the proposed method has lower complexity, which is caused by the more concise and efficient network structure in our method. Our method has higher time complexity; however, it leads to a significant performance improvement of overall accuracy. Compared to the accuracy improvement, the increase of time complexity is ignorable.

Table 2 PARAMS, GFLOPS and NUMBER of layers of different methods

Method	Params	GFLOPs	Number of layers
Wald et al.[13]	12704067	0.2125	65
Ours	5442820	16.59	38

Notes: Number of layers means the number of layers containing parameters in the network.

4.3 Visual comparison

To further illustrate the effectiveness of the proposed method for 3D scene graph generation, we compared the visualization results of Wald et al.[13] and our method.

As shown in Figure 2, Figure 2a is the input instance point cloud without category information, each 3D point is colored according to different instance tags, Figure 2b is the 3D scene graph generated by applying the method of Wald et al.[13], and Figure 2c is the 3D scene graph generated by employing our method. Each node and edge in the scene graph is marked with the predicted category and the ground truth. We select the predicted category with the highest confidence for display and place the ground truth in parentheses. Each node is displayed in a different color to distinguish, and each edge is colored according to the prediction accuracy, where green indicates the correct prediction and black indicates the wrong prediction. As shown in the figure, the two methods are quite different from the ground truth in the object category prediction, and only three and two objects are correctly predicted respectively. In terms of relationship prediction, our method has better accuracy. For example, the method of Wald et al.[13] cannot identify the relationship between walls, floors, and armchairs, while our method obtains accurate relationship predictions. We also notice that there are some uncertainties in the prediction of the ground truth about spatial relationships. Taking the relationship between the armchair and the adjacent wall as an example, we observe that the true prediction (the armchair is on the right side of the wall and the wall is in front of the armchair) is not a paired relationship.

We show the comparison results of the 3D scene graph prediction of another scene in Figure 3. Figure 3a is the 3D mesh model of the scene, where different instance objects are marked with different colors,

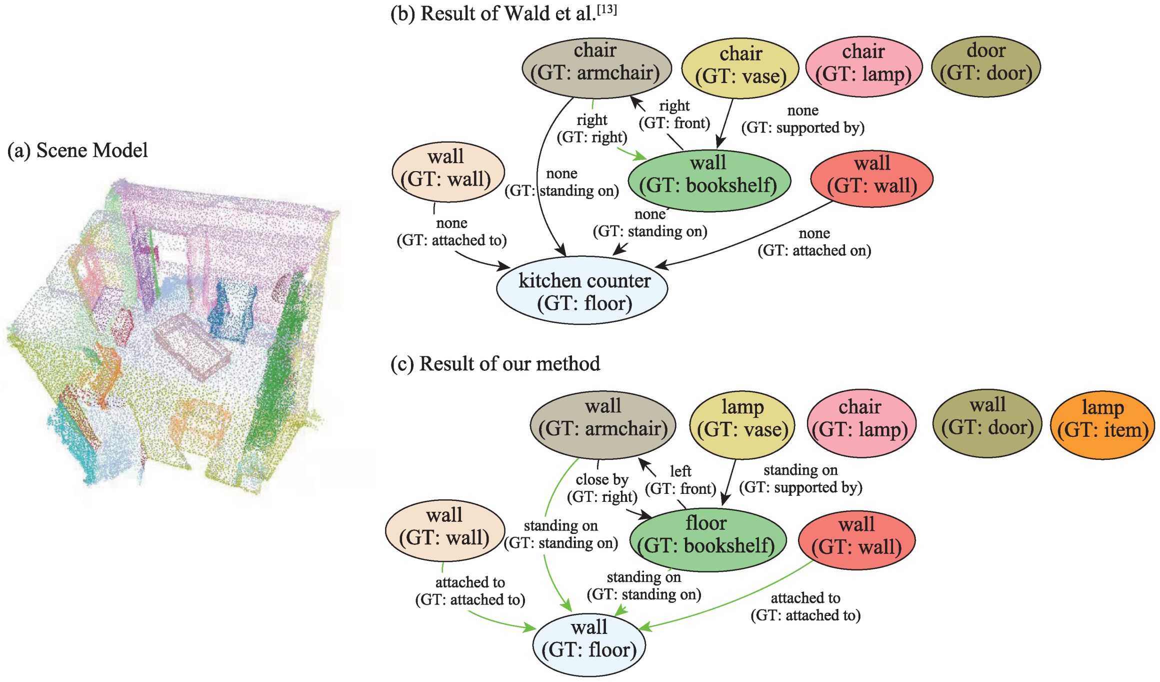

Figure 2 Qualitative comparison of 3D scene graph generation. (a) is the input point clouds annotated with class-agnostic instance segmentation; (b) is the 3D scene graph prediction of Wald et al.[13]; (c) is our result. In the scene graph, each node is an object of a different instance (color-coded), and each edge is marked with a predicted predicate and colored according to the prediction accuracy (green: correctly predicted edges, blue: missing ground truth, red: miss-classified edges, gray: wrongly predicted as none when GT is a valid relationship). In addition, the ground truth is noted in parentheses.

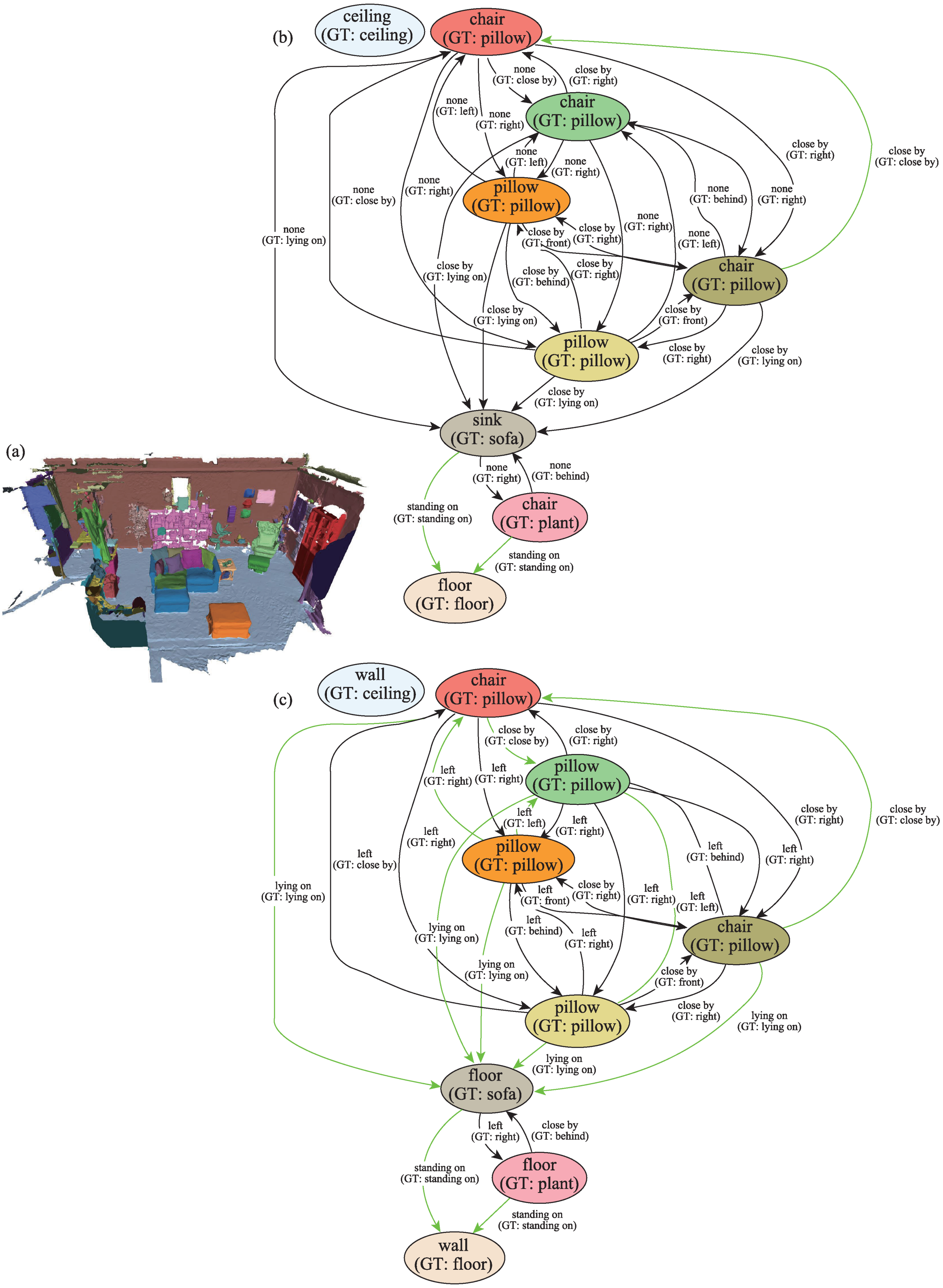

Figure 3 Qualitative comparison of 3D scene graph generation.

Figure 3b is the 3D scene graph predicted by the method of Wald et al.[13], and Figure 3c is the prediction result of our method. As shown in the figure, the object relationships in this scene are mostly left and right spatial relationships. Compared with the method of Wald et al.[13], our method has higher relationship prediction accuracy.

5 Conclusion

In this study, we propose a 3D scene graph prediction method based on the DGCNN and GAT. We use the class-agnostic point cloud as the input to discover the latent graph structure, where nodes and edges depict the objects and their relationships in the scene, respectively. Two DGCNNs are employed to extract the object and relationship features independently, and these are propagated to a GAT to further refine the graph structure. Empirical evaluations on the 3DSSG dataset show that the proposed method outperforms the state-of-the-art methods. However, many other factors, e.g., the offline scene graph prediction manner, need to be studied further by combining with SLAM systems; this would be our future work. Open source link (https://github.com/wffancy/3dssg).

Declaration of competing interest

We declare that we have no conflict of interest.

References

1 Cadena C, Carlone L, Carrillo H, Latif Y, Scaramuzza D, Neira J, Reid I, Leonard J J. Past, present, and future of simultaneous localization and mapping: toward the robust-perception age. IEEE Transactions on Robotics, 2016, 32(6): 1309−1332
DOI:10.1109/tro.2016.2624754

2 Özyeşil O, Voroninski V, Basri R, Singer A. A survey of structure from motion. Acta Numerica, 2017, 26: 305−364
DOI:10.1017/s096249291700006x

3 Furukawa Y, Hernández C. Multi-view stereo: a tutorial. Foundations and Trends® in Computer Graphics and Vision, 2015, 9(1/2): 1−148
DOI:10.1561/0600000052

4 Geng Q C, Zhou Z, Cao X C. Survey of recent progress in semantic image segmentation with CNNs. Science China Information Sciences, 2017, 61(5): 1−18
DOI:10.1007/s11432-017-9189-6

5 Wu Z R, Song S R, Khosla A, Yu F, Zhang L G, Tang X O, Xiao J X. 3D ShapeNets: a deep representation for volumetric shapes. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Boston, MA, USA, IEEE, 2015, 1912−1920
DOI:10.1109/cvpr.2015.7298801

6 Qi C R, Su H, Nießner M, Dai A, Yan M Y, Guibas L J. Volumetric and multi-view CNNs for object classification on 3D data. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, NV, USA, IEEE, 2016, 5648−5656
DOI:10.1109/cvpr.2016.609

7 Charles R Q, Hao S, Mo K C, Guibas L J. PointNet: deep learning on point sets for 3D classification and segmentation. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, HI, USA, IEEE, 2017, 77−85
DOI:10.1109/cvpr.2017.16

8 Qi C R, Yi L, Su H, Guibas L J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. arXiv preprint arXiv:1706.02413.

9 Wang Y, Sun Y B, Liu Z W, Sarma S E, Bronstein M M, Solomon J M. Dynamic graph CNN for learning on point clouds. ACM Transactions on Graphics, 2019, 38(5): 1−12
DOI:10.1145/3326362

10 Xu D F, Zhu Y K, Choy C B, Li F F. Scene graph generation by iterative message passing. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, HI, USA, IEEE, 2017, 3097−3106
DOI:10.1109/cvpr.2017.330

11 Krishna R, Chen V, Varma P, Bernstein M, Ré C, Li F F. Scene graph prediction with limited labels. In: 2019 IEEE/CVF

International Conference on Computer Vision (ICCV). Seoul, Korea (South), IEEE, 2019, 2580–2590
DOI:10.1109/iccv.2019.00267

12 Armeni I, He Z Y, Gwak J Y, Zamir A R, Fischer M, Malik J, Savarese S. 3D scene graph: A structure for unified semantics, 3D space, and camera. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, 5664-5673

13 Wald J, Dhamo H, Navab N, Tombari F. Learning 3D semantic scene graphs from 3D indoor reconstructions. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Seattle, WA, USA, IEEE, 2020, 3960–3969
DOI:10.1109/cvpr42600.2020.00402

14 Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks. 2016: arXiv: 1609.02907[cs.LG]. https: //arxiv.org/abs/1609.02907

15 Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y. Graph attention networks. arXiv preprint arXiv: 1710.10903, 2017

16 Marr D. Vision: a computational investigation into the human representation and processing of visual information. The Modern Schoolman, 1985, 62(2): 141–142

17 Liu W, Anguelov D, Erhan D, Szegedy, Reed S, Fu C Y, Berg A C. SSD: Single shot multibox detector. European Conference on Computer Vision. Springer, Cham, 2016, 21–37

18 He K M, Gkioxari G, Dollár P, Girshick R. Mask R-CNN. In: 2017 IEEE International Conference on Computer Vision (ICCV). Venice, Italy, IEEE, 2017, 2980–2988
DOI:10.1109/iccv.2017.322

19 Lee J H, Heo M, Kim K R, Kim C S. Single-image depth estimation based on Fourier domain analysis. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Salt Lake City, UT, USA, IEEE, 2018, 330–339
DOI:10.1109/cvpr.2018.00042

20 Zhao R Q, Wang Y, Martinez A M. A simple, fast and highly-accurate algorithm to recover 3D shape from 2D landmarks on a single image. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018, 40(12): 3059–3066
DOI:10.1109/tpami.2017.2772922

21 Laina I, Rupprecht C, Belagiannis V, Tombari F, Navab N. Deeper depth prediction with fully convolutional residual networks. In: 2016 Fourth International Conference on 3D Vision (3DV). Stanford, CA, USA, IEEE, 2016, 239–248
DOI:10.1109/3dv.2016.32

22 Suchi M, Patten T, Fischinger D, Vincze M. EasyLabel: a semi-automatic pixel-wise object annotation tool for creating robotic RGB-D datasets. In: 2019 International Conference on Robotics and Automation (ICRA). Montreal, QC, Canada, IEEE, 2019, 6678–6684
DOI:10.1109/icra.2019.8793917

23 Marion P, Florence P R, Manuelli L, Tedrake R. Label fusion: a pipeline for generating ground truth labels for real RGBD data of cluttered scenes. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). Brisbane, QLD, Australia, IEEE, 2018, 3235–3242
DOI:10.1109/icra.2018.8460950

24 Fisher M, Savva M, Hanrahan P. Characterizing structural relationships in scenes using graph kernels. ACM Transactions on Graphics, 2011, 30(4): 1–12
DOI:10.1145/2010324.1964929

25 Fisher M, Hanrahan P. Context-based search for 3D models. ACM Transactions on Graphics, 2010, 29(6): 1–10
DOI:10.1145/1882261.1866204

26 Zhao X, Wang H, Komura T. Indexing 3D scenes using the interaction bisector surface. ACM Transactions on Graphics, 2014, 33(3): 1–14
DOI:10.1145/2574860

27 Johnson J, Krishna R, Stark M, Li L J, Shamma D A, Bernstein M S, Fei-Fei L. Image retrieval using scene graphs. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Boston, MA, USA, IEEE, 2015, 3668–3678
DOI:10.1109/cvpr.2015.7298990

28 Zhu Y K, Groth O, Bernstein M, Li FF. Visual7W: grounded question answering in images. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, 4995–5004
DOI:10.1109/cvpr.2016.540

29 Zhao Y B, Zhu S C. Scene parsing by integrating function, geometry and appearance models. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition. Portland, OR, USA, IEEE, 2013, 3119–3126
DOI:10.1109/cvpr.2013.401

30 Huang S Y, Qi S Y, Zhu Y X, Xiao Y X, Xu Y L, Zhu S C. Holistic 3D scene parsing and reconstruction from a single RGB image. Computer Vision-ECCV, 2018, 187–230
DOI:10.1007/978-3-030-01234-2_12

31 Armeni I, He Z Y, Zamir A, Gwak J, Malik J, Fischer M, Savarese S. 3D scene graph: a structure for unified semantics, 3D space, and camera. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). Seoul, Korea (South), IEEE, 2019, 5663–5672
DOI:10.1109/iccv.2019.00576

32 Rosinol A, Gupta A, Abate M, Shi J N, Carlone L. 3D dynamic scene graphs: actionable spatial perception with places, objects, and humans. In: Robotics: Science and Systems XVI. Robotics: Science and Systems Foundation, 2020
DOI:10.15607/rss.2020.xvi.079

33 Kim U H, Park J M, Song T J, Kim J H. 3-D scene graph: a sparse and semantic representation of physical environments for intelligent agents. IEEE Transactions on Cybernetics, 2020, 50(12): 4921–4933
DOI:10.1109/tcyb.2019.2931042

34 Lin T Y, Goyal P, Girshick R, He K M, Dollár P. Focal loss for dense object detection. In: 2017 IEEE International Conference on Computer Vision (ICCV). Venice, Italy, IEEE, 2017, 2999–3007
DOI:10.1109/iccv.2017.324