Interior Layout Generation Based on Scene Graph and Graph Generation Model

Preprint · May 2020

DOI: 10.13140/RG.2.2.17245.03040

4 authors:

Xia Su
Tsinghua University
2 PUBLICATIONS **1** CITATION
SEE PROFILE

Chenglin Wu
Tsinghua University
28 PUBLICATIONS **717** CITATIONS
SEE PROFILE

Wen Gao
Tsinghua University
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Weixin Huang
Tsinghua University
30 PUBLICATIONS **86** CITATIONS
SEE PROFILE

Interior Layout Generation Based on Scene Graph and Graph Generation Model

Xia Su, Chenglin Wu, Wen Gao and Weixin Huang
Tsinghua University, China
su_xia1997@foxmail.com

Interior layout design is closely related to people's everyday life and is very widely demanded. As much workload of interior layout design being repetitive and categorized, automation and assistance could be applied with the help of most recent advancement of artificial intelligence. In this paper, we present an exploration work of automating interior layout design. We put forward a set of representation rules which turn interior scene photos into structuralized scene graphs. With representation rule containing both categorial and spatial information, we establish an interior scene graph dataset by annotating well-designed interior scene pictures downloaded from online photo sharing sites. Using the interior scene dataset which contains over 400 valid interior scene graphs, we train a graph generative model and further render its output as reconstructed scenes. The system could generate interior scene within short time and could potentially be applied in multiple related tasks.

Introduction

Interior layout design is one of the most related design categories to people. With the dramatically increasing population, urbanization rate and living standard around the world, more and more homes are built and the demand of interior layout design service is soaring. In this case, enterprises and researchers, like online interior design tool providers Planner5d, RoomSketcher, KooLab [1], [2] and Seehome, try to automate interior design in order to provide cheap, accessible and high-quality design service. In fact, we've seen countless work in this field contributed by researchers

Design Computing and Cognition DCC'20. J.S. Gero (ed),
pp. xx-yy.

from various disciplines. These works are made possible because of not only the wide demand of interior design, but also the relatively simple nature of interior design. Compared to other design categories like architectural design, industrial design or UI design, interior design is relatively simple in three ways. Firstly, interior layout design usually has a fixed design objective, especially when we focus on design of residential rooms. Secondly, interior design has relatively fixed component categories, usually limited kinds of furniture and item. Thirdly, the components of interior design have less intersection between each other compared with other design categories. In this case, simplifying interior design solution space as arrangement of interior items and applying state-of-the-art technology become feasible. An important observation is that interior layout can be well-represented as graphs, which could perfectly express the structure and inner relationships of interior scenes. Actually, searching through a space of labelled graphs has become a general way in the research field of 3D indoor synthesis [3] and multiple computational methods are conducted on the basis of graphs. Also, we could observe many researches following a 'simplify and compute' paradigm. Machine learning technologies, especially CNN [4], VAE [5], MCMC [6] and GNN [7] , has been adopted in related researches and yielded fruitful outcome. Specifically, the recent development of GNN (Graph Neural Network) and GCN (Graph Convolutional Network) has greatly helped the analysis, classification, prediction and generation tasks on graphs.

In this paper, we present an exploration work of automating interior layout design. We firstly put forward a set of representation rules which turn interior scene pictures into structuralized scene graphs. This representation process is partially bidirectional, since we could instantiate virtual indoor scenes with scene graphs, making it possible for us to simplify interior layout design task as the arrangement of scene graphs. Secondly, with the representation rule containing both categorial and spatial information, we establish an interior scene graph dataset by annotating over 1000 well-designed interior scene pictures (figure 1).The pictures are downloaded from picture sharing websites like Pinterest, making sure that they contain well-designed interior scenes. Finally, using the interior scene dataset which contains hundreds of valid interior scene graphs, we train a graph generative model to sample new graphs and further visualize its output as reconstructed scenes (figure 2).

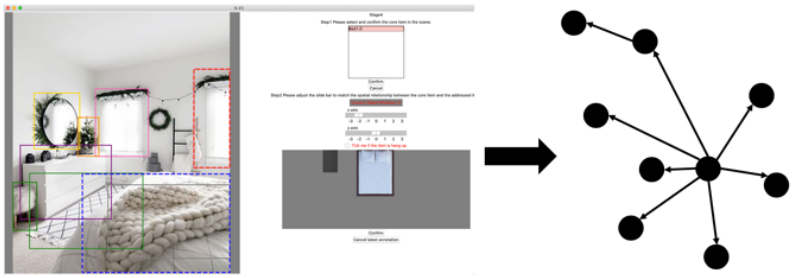

Fig1. Annotating tool interface and the output scene graph structure

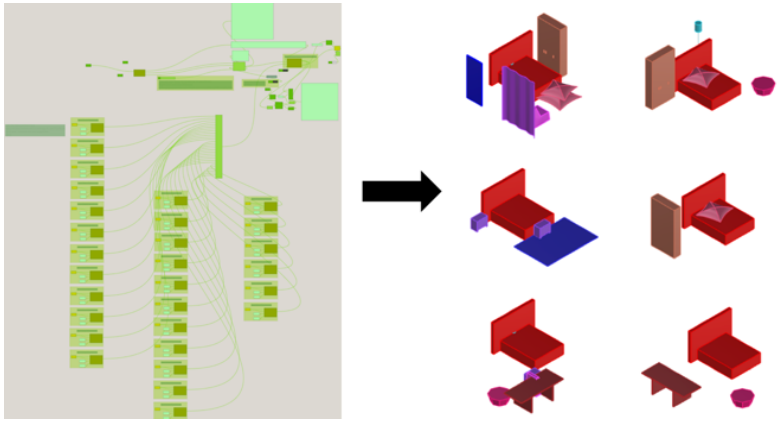

Fig2. Scene reconstruction program in Grasshopper and some example outputs

In summary, our contributions are:
1. Establishing a pipeline turning interior scene pictures into an annotated interior scene graph dataset.
2. Adjusting and extending graph generative model to produce richly-annotated interior scene graphs.
3. Instantiating virtual interior scenes based on scene graphs and rules using Rhinoceros and Grasshopper.

Background & related work

Interior Layout Generation

Interior layout generation receives much attention for its wide application and demand. Over the years, many researchers have tried to automatically or interactively generate interior layouts and their methods prominently

evolves with time. Early works usually adopt rule-based systems, trying to incorporate design field knowledge in the automatic systems. For example, Merrell et. al. [8] proposed a rule-based interactive interior layout system which suggests user with arrangements based on interior design guidelines. More recent works usually follow a 'simplify and compute' research paradigm, trying to reduce the size of search space and adopt state-of-the-art computational methods, especially machine learning methods. Xu W. et. al [9] present a wall-grid structure system to synthesize 3D interior scenes with given empty room shape. Qi et. al [10] synthesize 3D room layout using human-centric method. Kai Wang et. al. [4] used convolutional priors to automatically generate 3D model of indoor layout. Many related works are also categorized under the field of 3D indoor synthesis, and are systematically reviewed by Songhai Zhang et al. [3].

Scene graph

Representing scene as graph has been adopted as a common method in various domains, not only interior layout generation, but also image retrieval [11], image generation[12] and image captioning[13]. Scene graphs has been proved as an efficient and effective way of representing scenes with rich semantic relationships. Specifically, researches in interior layout generation have been widely using scene graphs and define the rules of representation based on their interest and need. For example, Chang et al. [14] turn text into graph with static support hierarchy and further generate virtual 3D interior scenes. They represent objects and rooms as nodes and spatial relationships as edges. Qi et al. [10] learn spatial And-Or graph and sample new interior scenes from overall graph. In this research, we also set up our unique rule of establishing scene graphs.

Graph Neural Network and graph generative models

Another important background of this work is the fast development of graph neural network (GNN) in recent years, which is thoroughly reviewed by Jie Zhou et al. [15] Since graph is a non-Euclidean data structure, conducting machine learning tasks like classification and clustering on graphs has been challenging for many years. Recently, the emergence of several key techniques has greatly boosted this domain. One of the key techniques introduced is GCN (graph convolution network), which is introduced by Kipf and Welling [16]. With graph convolution layers introduced, the non-uniform structured nature of graphs is overcome and overall analysis on graphs becomes easy. Another insightful work proves that GCN is actually a special form of Laplacian smoothing conducted on graph [17]. Based on GCN, many graph generative models are developed [18], [19], [20] and [21] with focus on different scale and feature. In this work, we adopt DGMG [21] as

the basis of our graph generative model and adjust the model architecture to better fit our generation task.

Representation of Interior Scene

Source of data

One of our starting points of this work is one key observation: among all online contents, only photos on certain websites contain interior designs of designer-level taste. Image sharing sites, like Pinterest and Huaban (which is the Chinese version of Pinterest), collects and manually select pictures to maintain the high quality of feeding. We could easily search 'Bedroom' on Pinterest and find hundreds of high-quality bedroom scene pictures. Compared to other source of data which are widely used in previous interior scene generation work, like the SUNCG dataset [22] or 3D Warehouse, the quality and design standard of online pictures from Pinterest is prominently better (Figure 3). Trying to utilize the best designed interior scenes, we collect over 1000 photos from Pinterest and Huaban and utilize them to generate new layouts with machine learning techniques. Our choice is novel because almost every other existing work of interior layout generation utilize 3D interior model datasets instead due to the convenience of extracting information, although these 3D models aren't considered well-designed from our perspective. Since most interior scene photos are about one certain room, we only focus on bedroom scene for now and make sure our method could be transferred to other residential rooms with minimal effort.

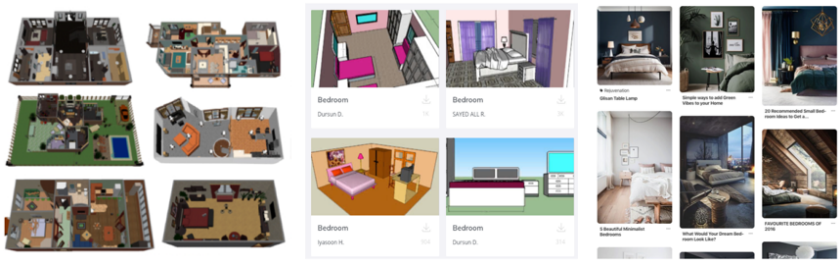

Fig3. Difference between three sources of interior scenes. Left: SUNCG dataset; middle: 3dwarehouse; right: Pinterest.

Representing scene with graph

However, utilizing interior scene images is usually tough. Common computer vision models used in tasks like object recognition and semantic segmentation usually cannot produce output precise enough to be further

utilized in indoor synthesis work, due to the fact that 2D image cannot embody all 3D information we need [23]. Our solution is to manually represent scene photos as graphs. Firstly, we limit the total number of item categories to 25, based on observation of the collected photo dataset and our design experience. The 25 categories of items include common bedroom furniture like bed, nightstand, cabinet and chair; functional devices like lamp, TV, laptop and pillow; as well as decorative items like portrait and craft. The full list of categories is shown in Table 1. Here we treat room elements like doors and windows as furniture items, instead of higher-level room components, for this stage of research in order to simplify the problem, due to the fact that many, if not most, pictures in our dataset don't show doors and windows.

Table 1 Relationships between author s and definitions

Furniture Categories	**Functional Devices**	**Decorative Items**
Bed, Window, Curtain, Door, Nightstand, Cabinet, Cupboard or Closet, Desk, Table, Chair, Stool	Lamp, Basket or Bin or Dirty Hamper, Mirror, TV, PC or Laptop, Clock, Mug or Cup, Pillow or Cushion, Blanket	Carpet or Mat, Vase or Pottery, Craft, Painting or Photo or Poster, Books,

With certain object categories, we could further represent relationship between objects by setting up an object hierarchy, which is also a common practice in related work. Based on previous work and our observation of interior design activity, we set up a 3-level hierarchy to express bedroom scenes: core item, independent item and supported item.

Core items are the most important items in our scene graphs, which is always bed since we only work on bedrooms in this work. Core items always exist and strongly influence how every other item is arranged. We set the core item as the highest-level node in our scene graph. Every other node is either directly or indirectly linked to the core item and the core item also serves as the origin of coordinates to help locate other items.

Independent items are mainly furniture or functional devices which are independently arranged in the scene, either on the floor or hang up to ceiling or wall. These items are directly linked to core items and the edge between them incorporate spatial information as a two-dimensional discrete vector. Some item categories, especially small decorative items like vase or pottery, cannot serve as independent items.

Supported items are at the lowest place in our 3-level hierarchy. Since objects being put upon other items is very common in interior scenes, including

some of the supported items in the scene graph is necessary. However, the supported items are usually small and movable, they don't influence the overall arrangement of interior scene. In this case, we only link the supported items to their supporter, which have to be the core item or independent item. The edges between them mark the support relation.

With the hierarchy as the rule of establishing interior scene graphs, we could turn certain pictures into fully attributed graphs. The graph not only record all items in the scene, but also mark their spatial relationships. One simple example is shown in Figure 4.

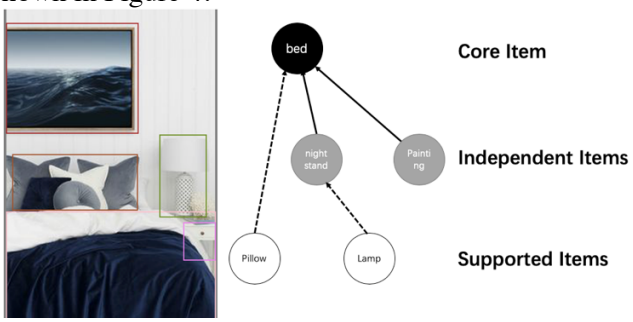

Fig4. Example of interior scene graph hierarchy

Annotation process

However, turning interior scene photo into scene graph could obviously be tough. Since establishing scene graph requires precise information of all object instances in scene and their mutual relationship. Although object detection and 3D reconstruction have been two well-discussed research tasks in computer vision, the existing tools and models cannot fulfill our need here because of two major reasons. Firstly, our dataset contains mostly well-designed bedroom scenes, which usually feature boutique and uncommon interior items, making it hard for pretrained models to be transferred on our dataset. Secondly, inferring 3D spatial relationship of complicated scene using single image is usually very hard when dealing with diversified scenes. Since our work is not focused on these computer vision tasks, we adopt manual annotation to avoid these workloads. At current stage we carry out a 4-step annotation process to manually mark all information we need and yield the scene graphs. We also developed a software tool to assist the annotating process. The interface is shown in Figure 4. The four steps are: (1) quality control; (2) bounding box labeling; (3) support relation pairing and (4) spatial relation numbering. All steps are generally too complicated problem to be automatic fulfilled by computer vision tools. For instance, the first quality control step marks whether the picture that are used for graph

building to be a valid bedroom scene and is decently designed. This judgement depends on annotator's professional design experience. Similarly, telling spatial relationship between objects on picture is also a task without reliable solution in computer vision research. Therefore, we recruit volunteers with design experience to finish these annotating tasks. The details of annotating steps are shown below.

Step 1: Quality Control

This step eliminates pictures that are not well-designed bedroom scene. The annotating tool provides annotators with a simple yes-or-no choice to make. If marked negative, the picture would be removed from dataset immediately.

Step 2: Bounding Box Labeling

In this step, annotators are required to draw bounding boxes for all object instances of 25 category. The categories are listed on the right part of interface and annotator click categories before dragging bounding boxes of the same category on the picture. The bounding boxes, one for every single object instance, will be in same color as the chosen category. The process is shown in Figure 6. After this step, all object instances are marked and recorded.

Step 3: Support Relation Pairing

After objects been fully labeled, we could confirm their support relations. Since only some of the item categories could possibly be supported, we could list the possible supported objects in a listbox and list their possible supporter in another listbox. Annotator could click the supported item and its supporter, and then click the pairing button to confirm the relation. Then the support relation is saved and the supported item is removed from the upper listbox. For example, in the example interface, the 'lamp' item is supported by the item of 'cupboard or closet' in the picture. The users choose these two items from two listboxes, see the dashed bounding box marking these objects, and confirm by clicking the 'pair' button. (Figure 5, 3rd interface).

Step 4: Spatial Relation Numbering

Spatial relationships between core item and other non-supported items are illustrated in step 4. Since the pictures are all bedroom scene, bed is set to be the layout center while other items' positions are all defined relatively to the bed. Specifically, the layout position is defined by x and y axis ranging from discrete integer -3 to 3. Setting bed as center (0,0), each of the rest item highlighted in dotted line in the picture could be set a coordinate describing its relative approximate position (not actual measurement) to the bed within the integer range on x and y axis. A layout illustration showing the relative position will assist this process in the lower part of the tool window.

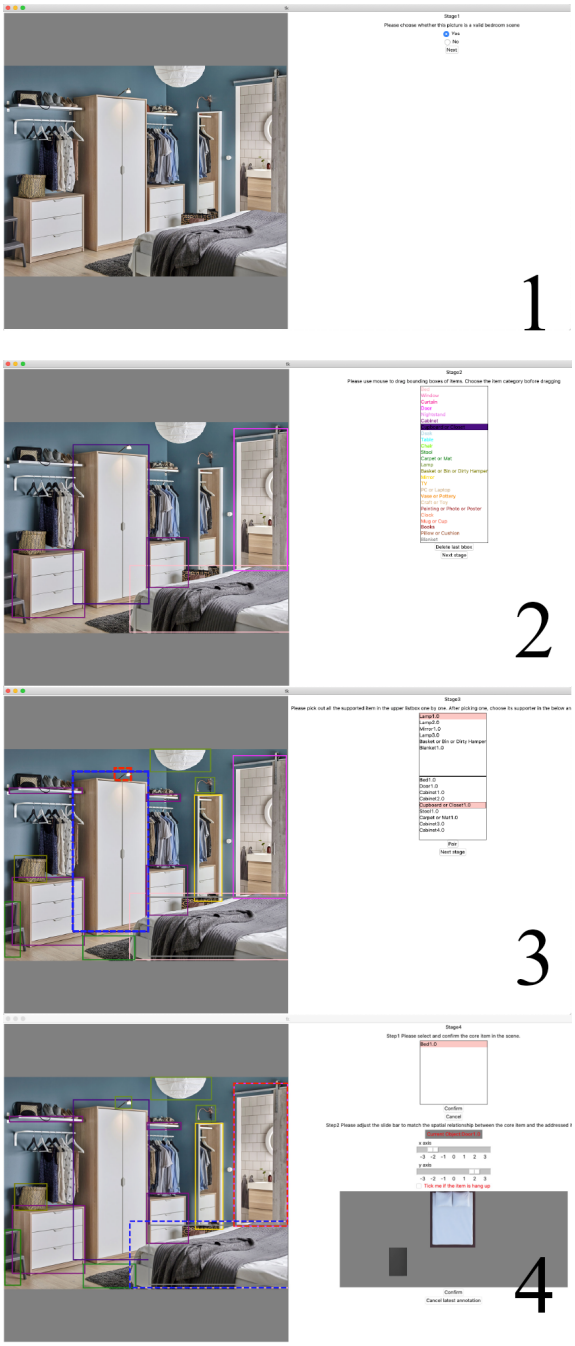

Fig5. 4 steps of our annotating tool

With four steps of annotation, the scene graphs are fully established and saved in xml format to serve as the FancyInterior dataset. We recruited 8 architecture students to annotate over 1000 interior scene pictures. Since many of the collected pictures don't contain a valid bedroom scene, the final dataset contains 492 valid bedroom scene graphs. We visualize the structure of some of these graphs with Networkx.(Figure 6)

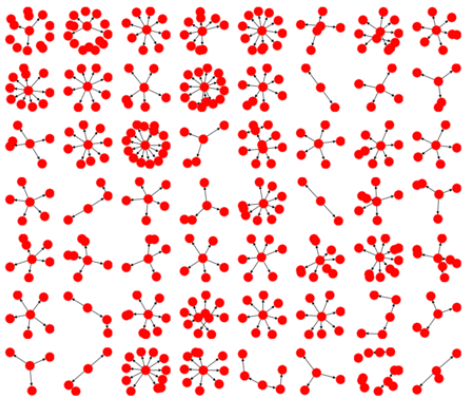

Fig6. Visualization of annotated scene graphs

Training of Graph Generation Model

With dataset collected and annotated, we could then train data-driven machine learning models to generate new graphs based on the feature distribution of the dataset. The generation of scene graph are conducted by training graph generation models. Although the graph generative model domain is still relatively new, several successful models have been proposed. Most of these models have limited range of application, some more adapted to small graphs and some to bigger ones. Comparing multiple models, we adopt DGMG (Li et al. 2018) as the basis of the graph generation model architecture in this work. DGMG (Deep Generative Model of Graphs) generate graphs by serially conduct three kinds of decisions: *whether to add node*, *whether to add edge* and *to which node the new edge be linked*. The decisions are separately generated by three linear layers with graph convolution layer output as their input. Since the graph convolution layer, here also called message passing graph convolution, is the crucial mechanism in graph generation, we introduce how it work in the following steps.

Initialization of hidden vectors. Since the graph convolution layer have to use all graph properties, we need to initialize hidden vectors to incorporate attributes. Firstly, we use two linear layers to map node and edge features x_v and x_e to latent representations h_v and h_e. The process goes as $h_v = f_{init-v}(x_v)$ and $h_e = f_{init-e}(x_e)$.

Message propagation along edges. With latent representation incorporating node and edge features, we could further pass these vectors along edges. Similar to the previous step, we use linear layer $f_{message}$ to compute the propagated message $m_{uv} = f_{message}(h_u, h_v, h_e)$. The messages are passed to end nodes of edges and get stored in message box.

Aggregation and updating. Since one node can be the destination of multiple edges, there might be many messages in its message box. Here we use the simplest method to map the variant degree to fixed dimension: we sum the messages up and feed the sum to a linear layer to produce node activation vector. Then the node latent vector is updated as $h_v^{t+1} = f_{update}^t(h_{activation}^t, h_v^t)$, here t represent t rounds of propagation. This propagate, aggregate and update process is repeated for certain rounds and the number of rounds is set to 2 in this work.

Gating and summing. In this work, what we need from graph convolution module is an overall representation of graph, called graph embedding vector h_G. It is calculated by applying gating functions on all nodes and sum the outputs. $h_G = \sum f_{activation}(h_v)$.

After the four steps, the message of nodes and edges are propagated around the graph, and a graph embedding vector is yielded to feed the following decision modules. Then serial decisions are serially made to generate graphs. Unlike the original DGMG setting, we have 4 decision modules in our model, which are Add Node, Node Feature, Edge Source, Edge Feature. The redesign of modules is based on the unique grammar of our scene graphs: every new node must have exactly one edge linked to one of the existing nodes. So the *whether to add edge* module in the original setting is skipped. The details of four decision modules are listed below.

Add Node: Every round of decision starts with adding a new node to graph. Since the node in our scene graph could be in 25 possible categories, we set up a linear layer with a 26-dimension softmax output. The first 25 dimensions are used as the probability of 25 categories while the last dimension refers to the probability of ending.

Node Feature: After adding a node, we use a linear layer to predict whether the object is hang up or not. At this stage there are only one node feature to determine in this module but any upcoming features can be easily added to this module.

Edge Source: Every new node has one edge to link to previous nodes, the problem is to determine the edge source. In this module all possible nodes are listed, and their latent vector is fed to a linear layer to calculate probability of each node.
Edge Feature: The last step is to predict the features of newly added edges. In this work, there are two edge features: support and spatial vector. They are both sampled from distributions generated by linear layers.
With all the layers and modules, the process of training and sampling is established, which is shown in Figure 7. The loss function is calculated as the sum of all sampling log-probability in decision modules. By minimizing the loss using gradient descent and back propagation, we could maximize the likelihood of generating graphs in dataset.

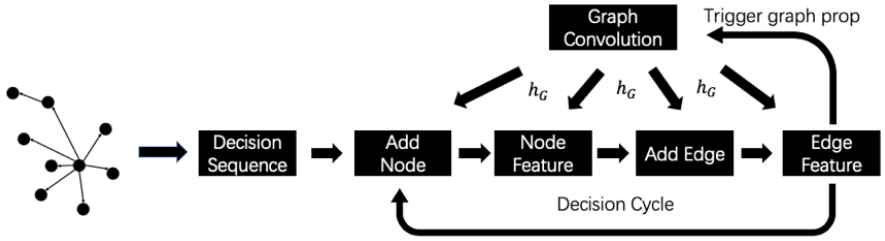

Fig7. Graph generative model pipeline

Here we list several key parameters of training in Table 2. These parameters are adjusted to fit our dataset.

Table 2 Training settings

Parameter name	**value**	**explanation**
hidden size	32	The dimension of node or edge embedded vector.
Prop round	2	How many rounds of propagation are done when a new edge is added to graph.
Optimizer	Adam	
Initial learning rate	1e-4	Initial learning rate of optimizer
Lr_dacay_rate	0.5/50	Learning rate times 0.5 every 50 epochs.
Clip gradient	0.25	constraint of gradient norm for gradient clipping

By adjusting and adding features based on initial framework, we trained the graph generation model with our scene graph dataset. The training is conducted on a 2018 Mac Mini with Intel i7 3.2GHz CPU and is finished very fast. 200 epochs of training only take around 2 hours and sampling of 1000 graphs takes extra 5 minutes to finish. The process is fast especially considering the fact that no GPU is involved. The following figure shows the loss of training process.

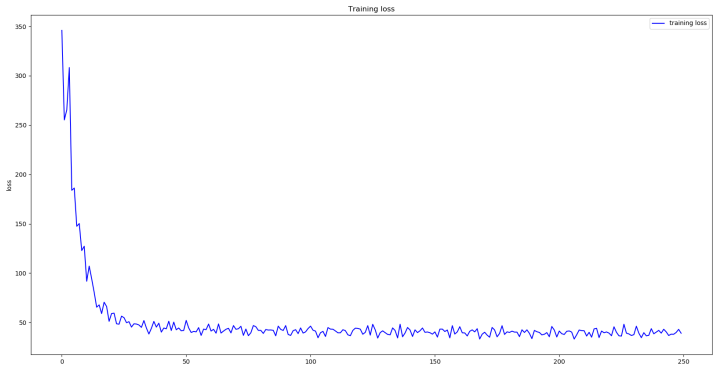

Fig8. Training loss

Generation of Interior Layout Based on Graph

Although sampling graphs with the generative model is highly efficient, some of the outcome might be invalid. To filter away the invalid outputs, we check the graphs with 3 tests:

- The graph should be a Directed Acyclic Graph (DAG), meaning no cycle exists in graph.
- The graph should have no more than 3 hierarchy counting from core item node.
- The graph should not have isolated nodes.

The tests can only exclude graphs with wrong topology structure. To further validate the quality of outcome, we need to visualize the graphs as reconstructed scenes. With category and spatial data represented in graph, we could render the graphs into 3D scenes with Rhinoceros and Grasshopper (shown in Figure 2). Since the visual features of interior items are not included in our current work, we use a fixed set of furniture 3d model to render the scene. Minimal texture and shape complexity are adopted to enable real-time output.

Arranging furniture into interior layouts is also a challenging task. Multiple computational methods are adopted in related works to help optimize the

scene outcome. Since our work is not mainly about this part, we come up a simple rule-based process to visualize our generated graphs. There are four steps in the visualization process: Step 1: place core item in space. Step 2: place independent items according to their spatial relationship with core item. Make sure they face the core item. If two or more objects take up same spot, move both of them to avoid overlapping. Step 3: place the supported items on their supporters. Step 4: add floor, wall and ceiling based on the scale of instantiated scene. The examples of generated virtual scenes are shown in figure 8.

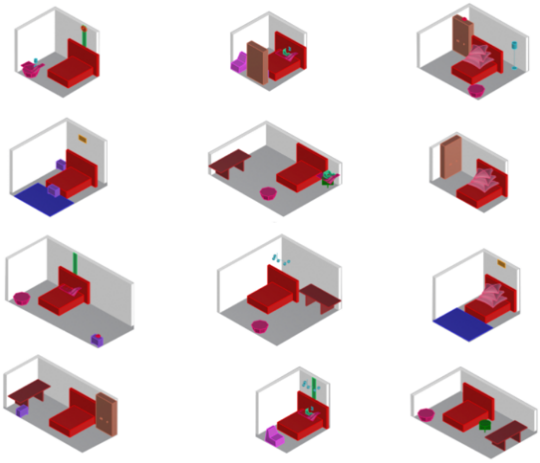

Fig9. Examples of reconstructed 3D bedroom scenes

Conclusion and Discussion

In this paper, we present an indoor layout generation method based on interior scene photo and graph generation model. By conducting annotating, training and rendering, dataset of interior scene is established and generative model is trained. We further validate the quality of output by conducting visualization using Rhinoceros and Grasshopper. From all outcome and performances, we could conclude that our interior layout generation method is highly functional and efficient.

There are several major limitations in this paper. Firstly, we only focus on bedroom scene in this work. Applying the method on other interior scenes, like living room and kitchen, would require much extra work since extra object category and dataset are needed. Secondly, the annotating process of scene pictures is relatively time-consuming and tiring. Partially involve computer vision models to alleviate workload might be inevitable in future work. Thirdly, the current reconstruction of 3D scene based on scene graph

is relatively simple and coarse. More detailed rules and optimization process is needed.

In our future work, we are going to further explore this method in many aspects including applying this generation pipeline on multiple interior scenes, automation of dataset generation, including visual feature in graph generating process and improving current graph generation model for richly-attributed graphs. We see great potential in this work since generation of multiple rooms could contribute to an all-purpose interior design tool. Also, the great flexibility and real-time-output nature of this graph generative model could be used to drive interactive recommender systems incorporated in design tools, which could greatly help novice users in interior layout design tasks.

References

1. Wu, W., Fu, X. M., Tang, R., Wang, Y., Qi, Y. H., & Liu, L. (2019). Data-driven interior plan generation for residential buildings. *ACM Transactions on Graphics (TOG)*, *38*(6), 1-12.
2. Li, W., Saeedi, S., McCormac, J., Clark, R., Tzoumanikas, D., Ye, Q., ... & Leutenegger, S. (2018). InteriorNet: Mega-scale multi-sensor photo-realistic indoor scenes dataset. *arXiv preprint arXiv:1809.00716*.
3. Zhang, S. H., Zhang, S. K., Liang, Y., & Hall, P. (2019). A Survey of 3D Indoor Scene Synthesis. Journal of Computer Science and Technology, 34(3), 594-608.
4. Wang, K., Savva, M., Chang, A. X., & Ritchie, D. (2018). Deep convolutional priors for indoor scene synthesis. ACM Transactions on Graphics (TOG), 37(4), 70.
5. Li, M., Patil, A. G., Xu, K., Chaudhuri, S., Khan, O., Shamir, A., ... & Zhang, H. (2019). GRAINS: Generative recursive autoencoders for indoor scenes. *ACM Transactions on Graphics (TOG)*, *38*(2), 12.
6. Huang, S. S., Fu, H., & Hu, S. M. (2016). Structure guided interior scene synthesis via graph matching. *Graphical Models*, *85*, 46-55.
7. Wang, K., Lin, Y. A., Weissmann, B., Savva, M., Chang, A. X., & Ritchie, D. (2019). Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Transactions on Graphics (TOG)*, *38*(4), 1-15.
8. Merrell, P., Schkufza, E., Li, Z., Agrawala, M., & Koltun, V. (2011, August). Interactive furniture layout using interior design guidelines. In ACM transactions on graphics (TOG) (Vol. 30, No. 4, p. 87). ACM.
9. Xu, W., Wang, B., & Yan, D. M. (2015). Wall grid structure for interior scene synthesis. *Computers & Graphics*, *46*, 231-243.

10. Qi, S., Zhu, Y., Huang, S., Jiang, C., & Zhu, S. C. (2018). Human-centric indoor scene synthesis using stochastic grammar. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5899-5908).
11. Johnson, J., Krishna, R., Stark, M., Li, L. J., Shamma, D., Bernstein, M., & Fei-Fei, L. (2015). Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3668-3678).
12. Johnson, J., Gupta, A., & Fei-Fei, L. (2018). Image generation from scene graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1219-1228).
13. Anderson, P., Fernando, B., Johnson, M., & Gould, S. (2016, October). Spice: Semantic propositional image caption evaluation. In *European Conference on Computer Vision* (pp. 382-398). Springer, Cham.
14. Chang, A., Savva, M., & Manning, C. D. (2014, October). Learning spatial knowledge for text to 3D scene generation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 2028-2038).
15. Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., & Sun, M. (2018). Graph neural networks: A review of methods and applications. arXiv preprint arXiv:1812.08434.
16. Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.
17. Li, Q., Han, Z., & Wu, X. M. (2018, April). Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
18. Brockschmidt, M., Allamanis, M., Gaunt, A. L., & Polozov, O. (2018). Generative code modeling with graphs. *arXiv preprint arXiv:1805.08490*.
19. De Cao, N., & Kipf, T. (2018). MolGAN: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*.
20. You, J., Liu, B., Ying, Z., Pande, V., & Leskovec, J. (2018). Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in Neural Information Processing Systems* (pp. 6410-6421).
21. Li, Y., Vinyals, O., Dyer, C., Pascanu, R., & Battaglia, P. (2018). Learning deep generative models of graphs. arXiv preprint arXiv:1803.03324.
22. Song, S., Yu, F., Zeng, A., Chang, A. X., Savva, M., & Funkhouser, T. (2017). Semantic scene completion from a single depth image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1746-1754).
23. Liu, M., Zhang, K., Zhu, J., Wang, J., Guo, J., & Guo, Y. (2018). Data-driven Indoor Scene Modeling from a Single Color Image with Iterative Object Segmentation and Model Retrieval. *IEEE transactions on visualization and computer graphics*.