Intelligent Home 3D: Automatic 3D-House Design from Linguistic Descriptions Only

Qi Chen[1,2*] Qi Wu[3*] Rui Tang[4] Yuhan Wang[4] Shuai Wang[4] Mingkui Tan[1†]

[1]South China University of Technology, [2]Guangzhou Laboratory, Guangzhou, China
[3]Australian Centre for Robotic Vision, University of Adelaide, Australia, [4]Kujiale, Inc.

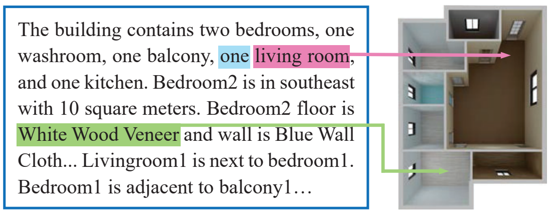
sechenqi@mail.scut.edu.cn qi.wu01@adelaide.edu.au
{ati,daishu,luorui}@qunhemail.com mingkuitan@scut.edu.cn

Abstract

Home design is a complex task that normally requires architects to finish with their professional skills and tools. It will be fascinating that if one can produce a house plan intuitively without knowing much knowledge about home design and experience of using complex designing tools, for example, via natural language. In this paper, we formulate it as a language conditioned visual content generation problem that is further divided into a floor plan generation and an interior texture (such as floor and wall) synthesis task. The only control signal of the generation process is the linguistic expression given by users that describe the house details. To this end, we propose a House Plan Generative Model (HPGM) that first translates the language input to a structural graph representation and then predicts the layout of rooms with a Graph Conditioned Layout Prediction Network (GC-LPN) and generates the interior texture with a Language Conditioned Texture GAN (LCT-GAN). With some post-processing, the final product of this task is a 3D house model. To train and evaluate our model, we build the first Text–to–3D House Model dataset, which will be released at: https://github.com/chenqi008/HPGM.

1. Introduction

Everyone wants a dream home, but not everyone can design home by themselves. Home design is a complex task that is normally done by certificated architects, who have to receive several years of training on designing, planning and using special designing tools. To design a home, they typically start by collecting a list of requirements for a building layout. Then, they use trial-and-error to generate layouts with a combination of intuition and prior experience. This usually takes from a couple of days to several weeks and has high requirements for professional knowledge.



(a) User linguistic requirements (b) 3D-House design

Figure 1: An example of generated 3D house with description using HPGM on the Text–to–3D House Model dataset. 3D-house generation from requirement seeks to design a 3D building automatically from given linguistic descriptions.

It will be fantastic if we can design our own home by ourselves. We may not have design knowledge and have no idea how to use those complicated professional designing tools, but we have strong linguistic ability to express our interests and desire. Thus, for time-saving and allowing people without expertise to participate in the design, we propose to use linguistic expressions as the guidance to generate home design plans (Figure 1). Thanks to the fast development of deep learning [6, 9, 11, 45, 46, 51, 52, 53, 55], especially Generative Adversarial Network (GAN) [3, 4, 8, 10] and vision-language research [13, 29, 38, 47, 50], we can turn this problem into a text-to-image generation problem, which has been studied in [22, 30, 32, 43, 44]. However, it is non-trivial to directly apply these methods on our new task because there exist two new technical challenges: **1)** A floor plan is a structured layout which pays more attention to the correctness of size, direction, and connection of different blocks, while the conventional text-to-image task focuses more on pixel-level generation accuracy. **2)** The interior texture such as floor and wall needs neater and more stable pixel generation than general images and should be

*Authors contributed equally.
†Corresponding author.

well aligned with the given descriptions.

To tackle the above issues, we propose a House Plan Generative Model (HPGM) to generate home plans from given linguistic descriptions. The HPGM first uses a Stanford Scene Graph Parser [33] to parse the language to a structural graph layout, where nodes represent room types associated with size, room floor (wall) colour and material. Edges between nodes indicate whether rooms are connected or not. We then divide the house plan generation process into two sub-tasks: *building layout generation* and *texture synthesis*. Both of them are conditioned on the above extracted structural graph. Specifically, we design a Graph Conditioned Layout Prediction Network (GC-LPN) which applies a Graph Convolutional Network [20] to encode the graph as a feature representation and predicts the room layouts via bounding box regressions. The predicted room layouts are sent to a floor plan post-processing step, which outputs a featured floor plan with doors, windows, walls, *etc*. To generate floor and wall textures, we design a Language Conditioned Texture GAN (LCT-GAN) that takes the encoded text representations as input and generates texture images with three designed adversarial, material-aware, and colour-aware losses. The generated floor plan and texture images are sent to an auto 3D rendering system to produce the final rendered 3D house plan.

For 3D house generation from linguistic description, we build the first Text–to–3D House Model dataset that contains a 2D floor plan and two texture (floor and wall) patches for each room in the house. We evaluate the room layout generation and texture generation ability of our model separately. The room layout accuracy is evaluated based on the IoU (Intersection over Union) between the predicted room bounding boxes and the ground-truth annotation. The generated interior textures are evaluated with popular image generation metrics such as *Fréchet Inception Distance (FID)* [12] and *Multi-scale Structural Similarity (MS-SSIM)* [39]. Our proposed GC-LPN and LCT-GAN outperform the baseline methods in a large margin. Besides, a generalisation ability evaluation of our LCT-GAN is carried out. We also perform a human evaluation on our final products – 3D house plans, which shows 39.41% pass it.

We highlight our principal contributions as follows:

- We propose a novel architecture, called House Plan Generative Model (HPGM), which is able to generate 3D house models with given linguistic expressions. To reduce the difficulty, we divide the generation task into two sub-tasks to generate floor plans and interior textures, separately.
- To achieve the goal of synthesising 3D building model from the text, we collect a new dataset consisting of the building layouts, texture images, and their corresponding natural language expressions.
- Extensive experiments show the effectiveness of our proposed method on both qualitative and quantitative metrics. We also study the generalisation ability of the proposed method by generating unseen data with the given new texts.

2. Related Work

Building layout design. Several existing methods have been proposed for generating building layouts automatically [1, 5, 24, 28, 40]. However, most of these methods generate the building layouts by merely adjusting the interior edges in a given building outline. Specifically, Merrel *et al*. [24] generate residential building layouts using a Bayesian network trained in architectural programs. Based on an initial layout, Bao *et al*. [1] formulate a constrained optimisation to characterise the local shape spaces and then link them to a portal graph to obtain the objective layout. Peng *et al*. [28] devise a framework to yield the floor plan by tiling an arbitrarily shaped building outline with a set of deformable templates. Wu *et al*. [40] develop a framework that generates building interiors with high-level requirements. More recently, Wu *et al*. [41] propose a data-driven floor plan generating system by learning thousands of samples. However, the above methods require either a given building outline or a detailed structured representation as the input while we generate the room layouts with human verbal commands.

Texture synthesis. Many existing works in terms of texture generation focus on transferring a given image into a new texture style [16, 21, 36] or synthesising a new texture image based on the input texture [7, 35, 42]. Different from that, we aim to solve the problem that generates texture images with given linguistic expressions. The closest alternative to our task is texture generation from random noise [2, 15]. Specifically, Jetchev *et al*. [15] propose a texture synthesis method based on GANs, which can learn a generating process from the given example images. Recently, to obtain more impressive images, Bergmann *et al*. [2] incorporate the periodical information into the generative model, which makes the model have the ability to synthesise periodic texture seamlessly. Even if these methods have a strong ability to produce plausible images, they have limited real-world applications due to the uncontrollable and randomly generated results. We use natural language as the control signal for texture generation.

Text to image generation. For generating an image from text, many GAN-based methods [17, 23, 30, 32, 43, 44, 48, 49, 54] have been proposed in this area. Reed *et al*. [32] transform the given sentence into text embedding and then generate image conditioning on the extracted embedding. Furthermore, to yield more realistic images, Zhang *et al*. [49] propose a hierarchical network, called StackGAN, which generates images with different sizes (from coarse to

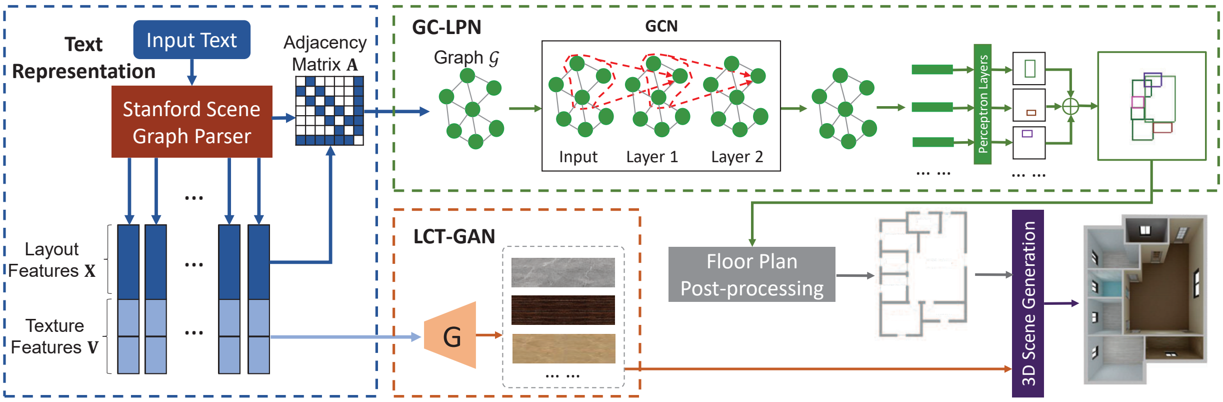


Figure 2: The overview of HPGM. We use Stanford Scene Graph Parser to parse given textual input and obtain the structural text representations $\mathbf{X}$, $\mathbf{V}$ and $\mathbf{A}$. Based on $\mathbf{X}$ and $\mathbf{A}$, GC-LPN yields a rough building layout using a graph convolutional network, followed by post-processing to refine the generated floor plan. LCT-GAN synthesises the corresponding textures for each room according to $\mathbf{V}$. Last, a 3D scene generation method is used to produce the objective 3D house plan.

fine). Meanwhile, they introduce a conditioning augmentation method to avoid the discontinuity in the latent manifold of text embedding. Based on StackGAN, Xu *et al*. [43] develop an attention mechanism, which ensures the alignment between generated fine-grained images and the corresponding word-level conditions. More recently, to preserve the semantic consistency, Qiao *et al*. [30] consider both text-to-image and image-to-text problems jointly.

3. Proposed Method

In this paper, we focus on 3D-house generation from requirements, which seeks to design a 3D building automatically conditioned on the given linguistic descriptions. Due to the intrinsic complexity of 3D-house design, we divide the generation process into two sub-tasks: *building layout generation* and *texture synthesis*, which produce floor plan and corresponding room features (*i.e*., textures of each room), respectively.

To complete the above two tasks, we propose a House Plan Generative Model (HPGM) to automatically generate a 3D home design conditioned on given descriptions. As shown in Figure 2, the proposed HPGM consists of five components: 1) text representation block, 2) graph conditioned layout prediction network (GC-LPN), 3) floor plan post-processing, 4) language conditioned texture GAN (LCT-GAN), and 5) 3D scene generation and rendering.

In Figure 2, the text representation is to capture the structural text information from given texts using a Stanford Scene Graph Parser [33]. Based on the text representations, GC-LPN is devised to produce a coarse building layout. To obtain a real-world 2D floor plan, we send the generated layout to a floor plan post-processing step to refine the coarse building layout to yield a floor plan with windows and doors. To synthesise the interior textures of each room, we further devise a Language Conditioned Texture GAN (LCT-GAN) to yield the controllable and neat images according to the semantic text representations. Last, we feed the generated floor plan with room features into a 3D rendering system for 3D scene generation and rendering. The details of each component are depicted below.

3.1. Text Representation

The linguistic descriptions of the building include the description of the number of rooms and room types, followed by the connections between rooms, and the designing patterns of each room. Although it follows a weakly structural format, directly using the template-based language parser is impractical due to the diversity of the linguistic descriptions. Instead, we employ the Stanford Scene Graph Parser [33] with some post-processing and merging to parse the linguistic descriptions to a structural graph format. For such a constructed graph, each node is a room with some properties (*e.g*., the room type, size, interior textures). The edge between nodes indicates the connectivity of two rooms. More details of the scene graph parser can be found in the supplementary materials.

We use different representation as inputs in building layout generation and texture synthesis, since these two tasks require different semantic information. In building layout generation, we define input vectors as $\mathbf{X} \in \mathbb{R}^{N\times D}$, where N refers to the number of nodes (*i.e*., rooms) in each layout and D denotes the feature dimension. Each node feature $\mathbf{x}_i = \{\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i, \boldsymbol{\gamma}_i\} \in \mathbb{R}^D$ is a triplet, where $\boldsymbol{\alpha}_i$ is the type of room (*e.g*., bedroom), $\boldsymbol{\beta}_i$ is the size (*e.g*., 20 squares) and $\boldsymbol{\gamma}_i$ is the position (*e.g*., southwest). All features are encoded as one-hot vectors except the size is a real value. Moreover, to exploit the topological information elaborately, following [20], we convert the input features $\mathbf{X}$ to an undirected graph $\mathcal{G}$ via introducing an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$.

In the texture synthesis task, for a given text, we trans-

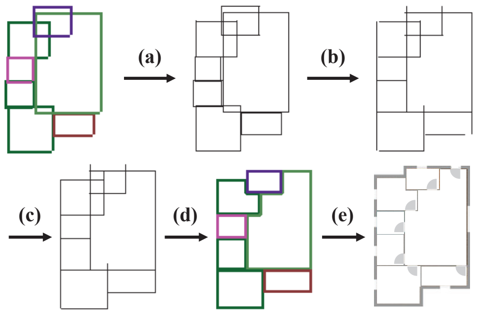


Figure 3: The procedures of floor plan post-processing.

form the linguistic expression to a collection of vectors $\mathbf{V} \in \mathbb{R}^{2N \times M}$, where $2N$ refers to the number of textures in each layout and M denotes the dimension of each feature vector. For $\mathbf{v}_i \in \mathbb{R}^M$, we design $\mathbf{v}_i = \{\mathbf{p}_i, \mathbf{q}_i\}$, where $\mathbf{p}_i$ indicates the material (*e.g.*, log, mosaic or stone brick) and $\mathbf{q}_i$ refers to the colour. We pre-build a material and colour word vocabulary from training data so that we can classify the parsed attributes into the material or colour set.

3.2. Graph Conditioned Layout Prediction Network

To generate the building layouts satisfying the requirements, we propose a Graph Conditioned Layout Prediction Network (GC-LPN). We incorporate the adjacent information into the extracted features via a GCN, which facilitates the performance when generating the objective layouts.

Graph convolutional network. In order to process the aforementioned graphs in an end-to-end manner, we use a graph convolutional network composed of two graph convolutional layers. Specifically, we take the feature matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ as inputs and produce a new feature matrix, where each output vector is an aggregation of a local neighbourhood of its corresponding input vector. In this way, we obtain a new feature matrix, which introduces the information across local neighbourhoods of the inputs. Note that, since we only focus on generating the layouts of resident building, the order and size of corresponding graph are small. Therefore, it is sufficient to leverage a two-layer GCN model (as shown in Figure 2) when introducing the information of adjacent rooms. Mathematically, we have

$$\mathbf{Y} = g(\mathbf{X}, \mathbf{A}) = \mathrm{Softmax}\left(\mathbf{A}\mathrm{ReLU}\left(\mathbf{A}\mathbf{X}\mathbf{W}_0\right)\mathbf{W}_1\right), \tag{1}$$

where $\mathbf{W}_0 \in \mathbb{R}^{D \times D}$ and $\mathbf{W}_1 \in \mathbb{R}^{D \times D}$ are the weights of two graph convolutional layers. Note that the adjacency matrix $\mathbf{A}$ only contains 1 and 0, which indicates whether pairs of nodes (rooms) are adjacent or not. $\mathbf{Y} \in \mathbb{R}^{N \times D}$ is the structured feature. Then, we add the extracted feature $\mathbf{Y}$ with the input feature $\mathbf{X}$ to get the feature $\mathbf{S} \in \mathbb{R}^{N \times D}$

$$\mathbf{S} = \mathbf{X} \oplus \mathbf{Y}, \tag{2}$$

where "$\oplus$" is the element-wise addition.

Bounding box regression. After reasoning on the graph with GCNs, we gain a set of embedding vectors, where each vector aggregates the information across the adjacent rooms. In order to produce the building layout, we must transform these vectors from the graph domain to the image domain. Thus, we define each room as a coarse 2D bounding box, which can be represented as $\mathbf{b}_i = (x_0, y_0, x_1, y_1)$. In this way, we cast the problem to that bounding box generation from given room embedding vectors.

In practice, we first feed the well-designed feature $\mathbf{S}$ into a two-layer perceptron network $h(\cdot)$ and predict the corresponding bounding box of each node $\hat{\mathbf{b}}_i = h(\mathbf{S}_i) = (\hat{x}_0, \hat{y}_0, \hat{x}_1, \hat{y}_1)$. Then, we integrate all the predicted boxes and obtain the corresponding building layout. For training the proposed model, we minimise the objective function

$$\mathcal{L}_B = \frac{1}{N}\sum_{i=1}^{N} \|\hat{\mathbf{b}}_i - \mathbf{b}_i\|_2^2, \tag{3}$$

where $\mathbf{b}_i$ is the ground-truth bounding box for i^{th} node (*i.e.*, the bounding box that covers the room).

3.3. Floor Plan Post-processing

To transform the bounding box layout to a real-world 2D floor plan, we propose a floor plan post-processing (shown in Figure 3), which consists of five steps, *i.e.*, (a)∼(e). To be specific, in Step **(a)**, we first extract boundary lines of all generated bounding boxes and then merge the adjacent segments together in Step **(b)**. In Step **(c)**, we further align the line segments with each other to obtain the closed polygon. In Step **(d)**, we judge the belonging of each closed polygon based on a weight function:

$$W_{ij} = \iint \frac{1}{w_i h_i} \exp\left(-\left(\frac{x_j - c_{x_i}}{w_i}\right)^2 - \left(\frac{y_j - c_{y_i}}{h_i}\right)^2\right) dx_j dy_j, \tag{4}$$

where $i = 1, 2, \ldots, n$ is the i^{th} original box (room) while $j = 1, 2, \ldots, m$ is the j^{th} aligned polygon. W_{ij} denotes the weight of j^{th} polygon belonging to room i. c_{x_i} and c_{y_i} indicate the central position while w_i and h_i are the half width and height of the i^{th} bounding box. x_j and y_j are the coordinates in the aligned polygon. We assign the j^{th} polygon with the room type, according to the corresponding original bounding box, which has maximum weight W.

Finally, in Step **(e)**, we apply a simple rule-based method to add doors and windows in rooms. Specifically, a door or open wall is added between the living room and any other room. We set the window on the longest wall of each room and set the entrance on the wall of the biggest living room. We find these rules work in most cases and good enough to set reasonable positions, but a learning-based method may improve this process and we leave it as the future work.

3.4. Language Conditioned Texture GAN

For better controlling the details of textures, we consider the texture images in terms of two fields, *i.e.*, material and

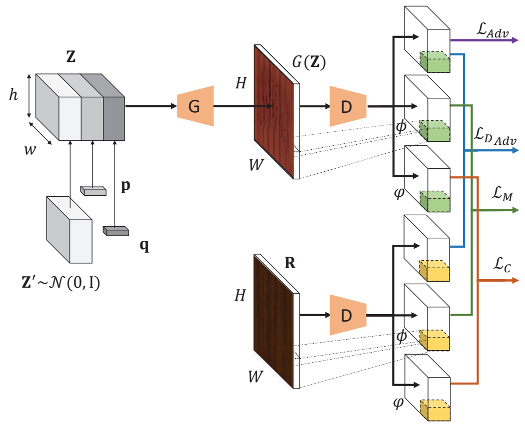


Figure 4: Architecture of LCT-GAN. Generator G transforms a conditional noise $\mathbf{Z}$ into an RGB image $G(\mathbf{Z})$ with fully convolutional layers. Discriminator D, which is used to distinguish fake data from real ones, is fed either fake image $G(\mathbf{Z})$ or real image $\mathbf{R}$. Two classifiers (ϕ and φ) have been added on top of D, which are used to impose the image into the right material and colour categories, respectively.

colour. In this way, we design a Language Conditioned Texture GAN (LCT-GAN) (in Figure 4), which can generate texture images that align with the given expressions.

Texture generator. We first obtain the input noise $\mathbf{Z}' \in \mathbb{R}^{w \times h \times d_1}$ from Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. After that, to incorporate the conditional information, we extend the aforementioned material and colour vectors $\mathbf{p} \in \mathbb{R}^{1 \times 1 \times d_2}$ and $\mathbf{q} \in \mathbb{R}^{1 \times 1 \times d_3}$ as the same size with the noise $\mathbf{Z}'$ and then concatenate them together to obtain the objective input $\mathbf{Z} \in \mathbb{R}^{w \times h \times (d_1 + d_2 + d_3)}$.

Conditioning on the input tensor $\mathbf{Z}$, we generate the corresponding texture image by $G(\mathbf{Z}) \in \mathbb{R}^{W \times H \times 3}$, where W and H denote the width and height of the generated image, respectively. Note that, in order to generate arbitrary size of texture, we design the generator G with a fully convolutional network (FCN), which allows input $\mathbf{Z}$ with various sizes when inferring. In practice, we establish our FCN model with only five blocks, where each block consists of a $2\times$ upsampling interpolation, a convolutional layer, a batch normalisation [14] and an activation function.

On the other hand, to generate texture from an expression, the generator G must: 1) ensure the generated images are natural and realistic ; and 2) preserve the semantic alignment between given texts and texture images. To satisfy the above requirements, we propose an optimisation mechanism consisting of three losses $\mathcal{L}_{Adv}$, $\mathcal{L}_M$ and $\mathcal{L}_C$, which indicate the adversarial loss, material-aware loss and colour-aware loss, respectively. Overall, the final objective function of the texture generator G is

$$\mathcal{L}_G = \mathcal{L}_{Adv} + \lambda_1 \mathcal{L}_M + \lambda_2 \mathcal{L}_C, \tag{5}$$

where λ_1 and λ_2 are trade-off parameters. In experiments, we set λ_1 and λ_2 to 1 by default. We will elaborate on the modules that lead to these losses in the following sections.

Adversarial loss. To synthesise the natural images, we follow the traditional GAN [8], where the generator G and discriminator D compete in a two-player minimax game. Specifically, the generator G tries to fool the discriminator D while D tries to distinguish whether the given image is real or fake/generated. Based on that, for our task, when optimising the discriminator D, we minimise the loss

$$\mathcal{L}_{D_{Adv}} = -\mathbb{E}_{\mathbf{R} \sim P_r}[\log D(\mathbf{R})] - \mathbb{E}_{\mathbf{Z}' \sim P_z}[\log(1 - D(G(\mathbf{Z})))], \tag{6}$$

where P_r and P_z denote the distributions of real samples and noise, respectively. $\mathbf{Z}$ refers to the input of G, as mentioned before, consisting of noise $\mathbf{Z}'$ and conditions $\mathbf{p}$ and $\mathbf{q}$. On the other hand, when optimising network G, we use

$$\mathcal{L}_{Adv} = -\mathbb{E}_{\mathbf{Z}' \sim P_z}[\log D(G(\mathbf{Z}))]. \tag{7}$$

Material-aware loss. To preserve the semantic alignment between generated textures and given texts, we propose a material-aware loss, which is sensitive to fine-grained material categories. To be specific, as mentioned in Section 3.1, we transform the linguistic descriptions to a structural format, which includes a label for each node to indicate its floor/wall material categories. We then add a material classifier on top of D, called ϕ, which imposes the generated texture into the right category. In this way, we obtain the posterior probability $\phi(\mathbf{c}_m|\cdot)$ of each entry image, where $\mathbf{c}_m$ refers to the category of material. Thus, we minimise the training loss of G and D as

$$\mathcal{L}_M = -\mathbb{E}_{\mathbf{R} \sim P_r}[\log \phi(\mathbf{c}_m|\mathbf{R})] - \mathbb{E}_{\mathbf{Z}' \sim P_z}[\log \phi(\mathbf{c}_m|G(\mathbf{Z}))]. \tag{8}$$

Colour-aware loss. Similar to the above material-aware loss, instead of focusing on materials, colour-aware loss pays more attention to colour categories. Based on the given expressions of texture colour, we cast the colour alignment as a classification problem. Specifically, we reuse the discriminator D as the feature extractor and replace the last layer to be a colour classifier φ. Then, in both G and D, we try to minimise the loss

$$\mathcal{L}_C = -\mathbb{E}_{\mathbf{R} \sim P_r}[\log \varphi(\mathbf{c}_c|\mathbf{R})] - \mathbb{E}_{\mathbf{Z}' \sim P_z}[\log \varphi(\mathbf{c}_c|G(\mathbf{Z}))], \tag{9}$$

where $\varphi(\mathbf{c}_c|\mathbf{R})$ is the posterior probability conditioning on the given texture image $\mathbf{R}$.

3.5. 3D Scene Generation and Rendering

For the better visualisation of the generated floor plan with textures, we introduce a 3D scene generator followed with a photo-realistic rendering process. Given generated floor plan and textures as shown in Figure 5, we generate walls from boundaries of rooms with fixed height and thickness. We set the height of walls to 2.85m and the thickness of interior walls to 120mm. The thickness of the exterior

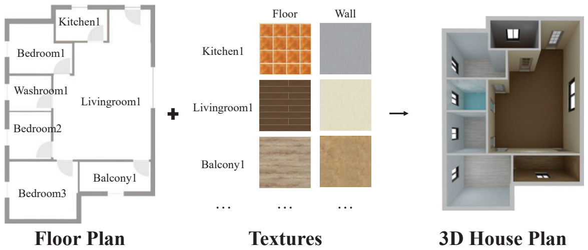


Figure 5: 3D house plan generation and rendering.

wall is set to 240mm while the length of the door is 900mm and the height is 2000mm. We simply set the length of the window to thirty percent of the length of the wall it belongs to. Besides, we develop a photo-realistic rendering based on Intel Embree [37], an open-source collection of high-performance ray tracing kernels for x86 CPUs. Photo-realistic renderer is implemented with Monte Carlo path tracing. By following the render equation [18], the path tracer simulates real-world effects such as realistic material appearance, soft shadows, indirect lighting, ambient occlusion and global illumination. In order to visualise the synthetic scenes, we deploy a virtual camera on the front top of each scene and capture a top-view render image.

4. Experiments

4.1. Experimental Settings

Dataset. To generate 3D building models from natural language descriptions, we collect a new dataset, which contains $2,000$ houses, $13,478$ rooms and 873[1] texture images with corresponding natural language descriptions. These descriptions are firstly generated from some pre-defined templates and then refined by human workers. The average length of the description is 173.73 and there are 193 unique words. In our experiments, we use $1,600$ pairs for training while 400 for testing in the building layout generation. For texture synthesis, we use 503 data for training and 370 data for testing. We put more dataset analysis in supplementary.

Evaluation metrics. We quantitatively evaluate our model and compare it with other models in threefold: layout generation accuracy, texture synthesis performance, and final 3D house floor plans. We measure the precision of the generated layout by *Intersection-over-Union (IoU)*, which indicates the overlap between the generated box and ground-truth one, where the value is from 0 to 1. For the evaluation of textures, we use *Fréchet Inception Distance (FID)* [12]. In general, the smaller this value is, the better performance the method will have. Besides, to test the pair-wise similarity of generated images and identify mode collapses reliably [26], we use *Multi-scale Structural Similarity (MS-SSIM)* [39] for further validation. A lower score indicates a higher diversity of generated images (*i.e.*, fewer model collapses). Note that, following the settings in [48], for a fair comparison, we resize all the images to 64×64 before computing FID and MS-SSIM. For the 3D house floor plans, which are our final products, we run a human study to evaluate them.

Implementation details. In practice, we set input $\mathbf{Z} \in \mathbb{R}^{w \times h \times (d_1+d_2+d_3)}$ of LCT-GAN with $h = 5$, $w = 5$, $d_1 = 100$, $d_2 = 19$ and $d_3 = 12$. All the weights of models (GC-LPN and LCT-GAN) are initialised from a normal distribution with zero-mean and standard deviation of 0.02. In training, we use Adam [19] with $\beta_1 = 0.5$ to update the model parameters of both GC-LPN and LCT-GAN. We optimise our LCT-GAN to generate texture images of size 160×160 with mini-batch size 24 and learning rate 0.0002.

4.2. Building Layout Generation Results

Compared methods. We evaluate the generated layout and compare the results with baseline methods. However, there is no existing work on our proposed text-guided layout generation task, which focuses on generating building layouts directly from given linguistic descriptions. Therefore, our comparisons are mainly to ablated versions of our proposed network. The compared methods are:

MLG: In "Manually Layout Generation" (MLG), we draw the building layouts directly using a program with the predefined rules, according to the given input attributes, such as type, position and size of the rooms. Specifically, we first roughly locate the central coordinates of each room conditioning on the positions. After that, we randomly pick the aspect ratio $\rho \in (\frac{2}{3}, \frac{3}{2})$ for different rooms, and then get the exact height and width by considering the size of each room. Finally, we draw the building layouts with such centre, height, width and type of each room.

C-LPN: In "Conditional Layout Prediction Network" (C-LPN), we simply remove the GCN in our proposed model. That means, when generating building layouts, the simplified model can only consider the input descriptions and ignore the information from neighbourhood nodes.

RC-LPN: In "Recurrent Conditional Layout Prediction Network" (RC-LPN), we yield the outline box of rooms sequentially like [34]. To be specific, we replace GCN with an LSTM and predict the building layout by tracking the history of what has been generated so far.

Quantitative evaluation. We evaluate the performance of our proposed GC-LPN by calculating the average IoU value of the generated building layouts. From Table 1, compared with the baseline methods, GC-LPN obtains higher value in IoU, which implies that the GC-LPN has the capacity to locate the outline of layout more precisely than other approaches. Models without our graph-based representation,

[1] Some rooms have same textures so this number is smaller than the total number of rooms.

	MLG	C-LPN	RC-LPN	GC-LPN (ours)
IoU	0.7208	0.8037	0.7918	**0.8348**

Table 1: IoU results on Text-to-3D House Model dataset.

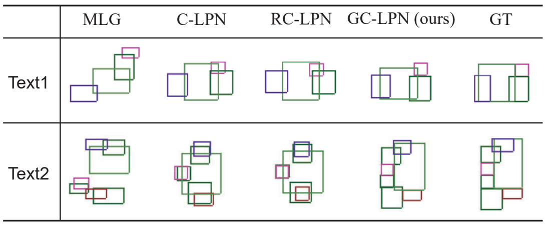


Figure 6: Visual comparisons between GC-LPN and baselines. "Text1" and "Text2" are the input descriptions, where "Text1" is relatively simple while "Text2" is more complex.

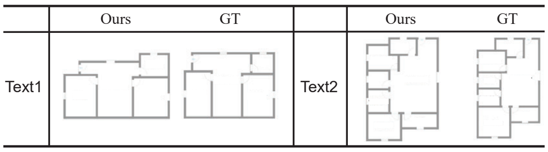


Figure 7: Examples of generated 2d floor plans and ground-truth counterparts with "Text1" and "Text2", respectively.

such as C-LPN and RC-LPN, have lower performance.

Qualitative evaluation. Moreover, we investigate the performance of our GC-LPN by visual comparison. From Figure 6, we provide two layout samples corresponding to "Text1[2]" and "Text2[3]" respectively. The results show that compared with the baseline methods, GC-LPN obtains more accurate layouts, whether simple or complex. We also present the generated 2D floor plans after post-processing, and the corresponding ground-truths in Figure 7.

4.3. Texture Synthesis Results

Compared methods. For the conditional texture generation task, we compare the performance of our proposed method with several baselines, including ACGAN [26], StackGAN-v2 [48] and PSGAN [2]. Note that PSGAN can only generate image from random noise. Thus, to generate images in a controlled way, we design a variant of PSGAN, which introduces the conditional information when synthesising the objective texture like [25].

Quantitative evaluation. In this part, we compare the performance of different methods on our proposed dataset in terms of FID and MS-SSIM. In Table 2, our LCT-GAN achieves the best performance in FID, which implies that our method is able to yield more photo-realistic images than others. Moreover, for MS-SSIM, our LCT-GAN obtains the competitive result compared with PSGAN, which is also designed specifically for texture generation. It suggests that our method has the ability to ensure the diversity of synthesised images when preserving realism.

Methods	Train Set		Test Set	
	FID	MS-SSIM	FID	MS-SSIM
ACGAN [26]	198.07	0.4584	220.18	0.4601
StackGAN-v2 [48]	182.96	0.6356	188.15	0.6225
PSGAN [2]	195.29	0.4162	217.12	0.4187
LCT-GAN (ours)	**119.33**	**0.3944**	**145.16**	**0.3859**

Table 2: FID and MS-SSIM results of generated textures.

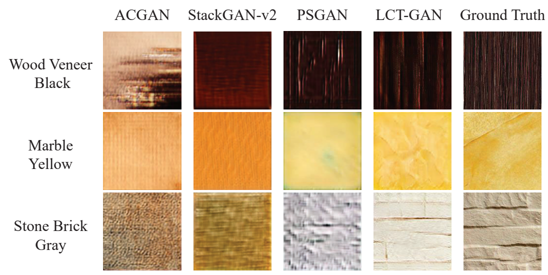


Figure 8: Visual results of LCT-GAN and baselines.

Qualitative evaluation. For further evaluating the performance of LCT-GAN, we provide several visual results of the generated textures. From Figure 8, compared with the baselines, our synthesised images contain more details while preserving the semantic alignment with the conditional descriptions. The results demonstrate LCT-GAN has the ability to semantically align the given texts and capture more detailed information than other approaches.

Ablation studies. To test the effect of each proposed loss, we conduct an ablation study to compare the generated results by removing some losses and show the quantitative results in Table 3. Note that the model only using adversarial loss $\mathcal{L}_{Adv}$ can not yield controllable images. Thus, we combine $\mathcal{L}_{Adv}$ with the other two losses (*i.e.*, $\mathcal{L}_M$ and $\mathcal{L}_C$) to investigate the performance. The results show that based on $\mathcal{L}_{Adv}$, $\mathcal{L}_M$ and $\mathcal{L}_C$ are able to improve the performance very well. When using all the three losses into our model, we obtain the best results on both FID and MS-SSIM.

Generalisation ability. In this section, we conduct two experiments to verify the generalisation ability of our proposed method. We first investigate the landscape of the latent space. Following the setting in [31], we conduct the linear interpolations between two input embeddings and feed them into the generator G. As shown in Figure 9, the generated textures change smoothly when the input semantics (*i.e.*, material or colour) vary. On the other hand, to further evaluate the generalisation ability of our LCT-GAN, we feed some novel descriptions, which are not likely to be seen in the real world, into the generator G. From Fig-

[2] *The building contains one washroom, one bedroom, one livingroom, and one kitchen. Specifically, washroom1 has 5 squares in northeast. bedroom1 has 14 square meters in east. Besides, livingroom1 covers 25 square meters located in center. kitchen1 has 12 squares in west. bedroom1, kitchen1, washroom1 and livingroom1 are connected. bedroom1 is next to washroom1.*

[3] Due to page limit, we put the content of "Text2" in the supplementary.

	$\mathcal{L}_{Adv}$	$\mathcal{L}_{M}$	$\mathcal{L}_{C}$	Train Set		Test Set	
				FID	MS-SSIM	FID	MS-SSIM
LCT-GAN	√	√		134.06	0.4189	157.01	0.4191
	√		√	134.61	0.4310	158.20	0.4263
	√	√	√	**119.33**	**0.3944**	**145.16**	**0.3859**

Table 3: Impact of losses in conditional texture generation.

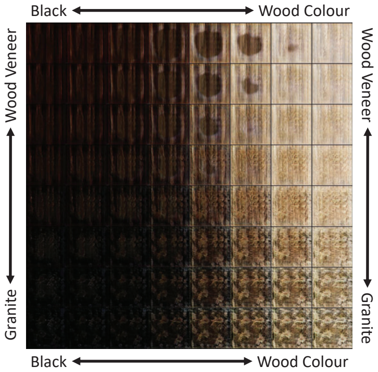


Figure 9: Interpolation results of generated texture images.

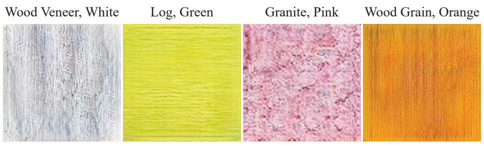


Figure 10: Generated textures with novel material-colour scenarios, which are impossible existing in the real world.

ure 10, even with such challenging semantic setting, our proposed method is still able to generate meaningful texture images. Both the two experiments suggest that LCT-GAN generalises well to novel/unseen images rather than simply remembering the existing data in the training set.

4.4. 3D House Design

Qualitative results. For quality evaluation, we show the 3D house plans (in Figure 11) generated by our HPGM and the ground-truth counterparts with conditional text[4], where the floor plan and corresponding room textures are drawn by architects. Our method has ability to produce competitive visual results, even compared with the human-made plans.

Human study. Since the automatic metrics can not fully evaluate the performance of our method, we perform a human study on the house plans. Inspired by [23, 27], we conduct a pairwise comparison between HPGM and human beings, using 100 house plans pairs with their corresponding descriptions. Then, we ask 20 human subjects (university students) to distinguish which is designed by human beings. Finally, we calculate the ratio of choice and obtain the final metrics. From Table 4, 39.41% generated samples pass the exam, which implies that compared with the manually designed samples, the machine-generated ones are exquisite enough to confuse the evaluators.

[4] *Building layout contains one washroom, one study, one livingroom, and one bedroom. To be specific, washroom1 has Blue Marble floor, and wall is Wall_Cloth and White. washroom1 is in southeast with 11 square meters. Additionally, study1 has Wood_color Log floor as well as has Yellow Wall_Cloth wall. study1 has 8 squares in west. livingroom1 is in center with 21 square meters. livingroom1 wall is Earth_color Wall_Cloth while uses Black Log for floor. Besides, bedroom1 covers 10 square meters located in northwest. bedroom1 floor is Wood_color Log, and has Orange Pure_Color_Wood wall. livingroom1 is adjacent to washroom1, bedroom1, study1. bedroom1 is next to study1.*

	HPGM (ours)	Human	Tie
Choice (%)	39.41	47.94	12.65

Table 4: Results of HPGM v.s. human. "Tie" refers to the confusing results, which can not be clearly distinguished.

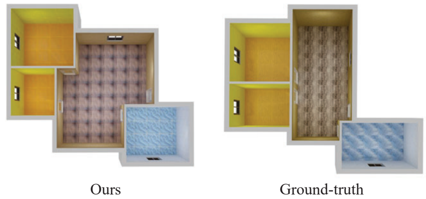


Figure 11: Comparison of our generated 3D house plans with ground-truth (human-made) counterparts.

5. Conclusion

3D house generation from linguistic descriptions is non-trivial due to the intrinsic complexity. In this paper, we propose a novel House Plan Generative Model (HPGM), dividing the generation process into two sub-task: *building layout generation* and *texture synthesis*. To tackle these problems, we propose two modules (*i.e.*, GC-LPN and LCT-GAN), which focus on producing floor plan and corresponding interior textures from given descriptions. To verify the effectiveness of our method, we conduct a series of experiments, including quantitative and qualitative evaluations, ablation study, human study, *etc*. The results show that our method performs better than the competitors, which indicates the value of our approach. We believe this will be a practical application with further polish.

Acknowledgments

This work was partially supported by Guangdong Provincial Scientific and Technological Funds under Grant 2018B010107001 and Grant 2019B010155002, National Natural Science Foundation of China (NSFC) 61836003 (key project), Fundamental Research Funds for the Central Universities under Grant D2191240, Program for Guangdong Introducing Innovative and Enterpreneurial Teams under Grant 2017ZT07X183, Tencent AI Lab Rhino-Bird Focused Research Program under Grant JR201902, Guangdong Special Branch Plans Young Talent with Scientific and Technological Innovation under Grant 2016TQ03X445, Guangzhou Science and Technology Planning Project under Grant 201904010197, Natural Science Foundation of Guangdong Province under Grant 2016A030313437.

References

[1] Fan Bao, Dong-Ming Yan, Niloy J Mitra, and Peter Wonka. Generating and exploring good building layouts. *ACM Transactions on Graphics (TOG)*, 32(4):122, 2013. 2

[2] Urs Bergmann, Nikolay Jetchev, and Roland Vollgraf. Learning texture manifolds with the periodic spatial gan. In *Proc. Int. Conf. Mach. Learn.*, pages 469–477, 2017. 2, 7

[3] Jiezhang Cao, Yong Guo, Qingyao Wu, Chunhua Shen, Junzhou Huang, and Mingkui Tan. Adversarial learning with local coordinate coding. In *Proc. Int. Conf. Mach. Learn.*, 2018. 1

[4] Jiezhang Cao, Langyuan Mo, Yifan Zhang, Kui Jia, Chunhua Shen, and Mingkui Tan. Multi-marginal wasserstein gan. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 1774–1784, 2019. 1

[5] Stanislas Chaillou. Archigan: a generative stack for apartment building design. 2019. 2

[6] Peihao Chen, Chuang Gan, Guangyao Shen, Wenbing Huang, Runhao Zeng, and Mingkui Tan. Relation attention for temporal action localization. *IEEE Trans. Multimedia*, 2019. 1

[7] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 262–270, 2015. 2

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 2672–2680, 2014. 1, 5

[9] Yong Guo, Jian Chen, Jingdong Wang, Qi Chen, Jiezhang Cao, Zeshuai Deng, Yanwu Xu, and Mingkui Tan. Closed-loop matters: Dual regression networks for single image super-resolution. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2020. 1

[10] Yong Guo, Qi Chen, Jian Chen, Qingyao Wu, Qinfeng Shi, and Mingkui Tan. Auto-embedding generative adversarial networks for high resolution image synthesis. *IEEE Trans. Multimedia*, 2019. 1

[11] Yong Guo, Yin Zheng, Mingkui Tan, Qi Chen, Jian Chen, Peilin Zhao, and Junzhou Huang. Nat: Neural architecture transformer for accurate and compact architectures. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 735–747, 2019. 1

[12] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 6626–6637, 2017. 2, 6

[13] Deng Huang, Peihao Chen, Runhao Zeng, Qing Du, Mingkui Tan, and Chuang Gan. Location-aware graph convolutional networks for video question answering. In *Proc. Conf. AAAI*, 2020. 1

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Proc. Int. Conf. Mach. Learn.*, 2015. 5

[15] Nikolay Jetchev, Urs Bergmann, and Roland Vollgraf. Texture synthesis with spatial generative adversarial networks. *arXiv preprint arXiv:1611.08207*, 2016. 2

[16] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Proc. Eur. Conf. Comp. Vis.*, pages 694–711. Springer, 2016. 2

[17] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 1219–1228, 2018. 2

[18] James T Kajiya. The rendering equation. In *ACM SIGGRAPH computer graphics*, volume 20, pages 143–150. ACM, 1986. 6

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Proc. Int. Conf. Learn. Representations*, 2015. 6

[20] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *Proc. Int. Conf. Learn. Representations*, 2017. 2, 3

[21] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *Proc. Eur. Conf. Comp. Vis.*, pages 702–716. Springer, 2016. 2

[22] Wenbo Li, Pengchuan Zhang, Lei Zhang, Qiuyuan Huang, Xiaodong He, Siwei Lyu, and Jianfeng Gao. Object-driven text-to-image synthesis via adversarial training. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 12174–12182, 2019. 1

[23] Yitong Li, Zhe Gan, Yelong Shen, Jingjing Liu, Yu Cheng, Yuexin Wu, Lawrence Carin, David Carlson, and Jianfeng Gao. Storygan: A sequential conditional gan for story visualization. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 6329–6338, 2019. 2, 8

[24] Paul Merrell, Eric Schkufza, and Vladlen Koltun. Computer-generated residential building layouts. In *ACM Transactions on Graphics (TOG)*, volume 29, page 181. ACM, 2010. 2

[25] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 7

[26] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *Proc. Int. Conf. Mach. Learn.*, pages 2642–2651, 2017. 6, 7

[27] Yingwei Pan, Zhaofan Qiu, Ting Yao, Houqiang Li, and Tao Mei. To create what you tell: Generating videos from captions. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1789–1798. ACM, 2017. 8

[28] Chi-Han Peng, Yong-Liang Yang, and Peter Wonka. Computing layouts with deformable templates. *ACM Transactions on Graphics (TOG)*, 33(4):99, 2014. 2

[29] Yuankai Qi, Qi Wu, Peter Anderson, Marco Liu, Chunhua Shen, and Anton van den Hengel. Rerere: remote embodied referring expressions in real indoor environments. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2020. 1

[30] Tingting Qiao, Jing Zhang, Duanqing Xu, and Dacheng Tao. Mirrorgan: Learning text-to-image generation by redescription. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 1505–1514, 2019. 1, 2, 3

[31] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *Proc. Int. Conf. Learn. Representations*, 2016. 7

[32] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *Proc. Int. Conf. Mach. Learn.*, 2016. 1, 2

[33] Sebastian Schuster, Ranjay Krishna, Angel Chang, Li Fei-Fei, and Christopher D Manning. Generating semantically precise scene graphs from textual descriptions for improved image retrieval. In *Proceedings of the fourth workshop on vision and language*, pages 70–80, 2015. 2, 3

[34] Fuwen Tan, Song Feng, and Vicente Ordonez. Text2scene: Generating compositional scenes from textual descriptions. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 6710–6719, 2019. 6

[35] Matthew Tesfaldet, Marcus A Brubaker, and Konstantinos G Derpanis. Two-stream convolutional networks for dynamic texture synthesis. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 6703–6712, 2018. 2

[36] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 6924–6932, 2017. 2

[37] Ingo Wald, Sven Woop, Carsten Benthin, Gregory S Johnson, and Manfred Ernst. Embree: a kernel framework for efficient cpu ray tracing. *ACM Transactions on Graphics (TOG)*, 33(4):143, 2014. 6

[38] Peng Wang, Qi Wu, Jiewei Cao, Chunhua Shen, Lianli Gao, and Anton van den Hengel. Neighbourhood watch: Referring expression comprehension via language-guided graph attention networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 1960–1968, 2019. 1

[39] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003. 2, 6

[40] Wenming Wu, Lubin Fan, Ligang Liu, and Peter Wonka. Miqp-based layout design for building interiors. In *Computer Graphics Forum*, volume 37, pages 511–521. Wiley Online Library, 2018. 2

[41] Wenming Wu, Xiao-Ming Fu, Rui Tang, Yuhan Wang, Yu-Hao Qi, and Ligang Liu. Data-driven interior plan generation for residential buildings. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 38(6), 2019. 2

[42] Wenqi Xian, Patsorn Sangkloy, Varun Agrawal, Amit Raj, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Texturegan: Controlling deep image synthesis with texture patches. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 8456–8465, 2018. 2

[43] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018. 1, 2, 3

[44] Guojun Yin, Bin Liu, Lu Sheng, Nenghai Yu, Xiaogang Wang, and Jing Shao. Semantics disentangling for text-to-image generation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 2327–2336, 2019. 1, 2

[45] Runhao Zeng, Chuang Gan, Peihao Chen, Wenbing Huang, Qingyao Wu, and Mingkui Tan. Breaking winner-takes-all: Iterative-winners-out networks for weakly supervised temporal action localization. *IEEE Trans. Image Process.*, 28(12):5797–5808, 2019. 1

[46] Runhao Zeng, Wenbing Huang, Mingkui Tan, Yu Rong, Peilin Zhao, Junzhou Huang, and Chuang Gan. Graph convolutional networks for temporal action localization. In *Proc. IEEE Int. Conf. Comp. Vis.*, Oct 2019. 1

[47] Runhao Zeng, Haoming Xu, Wenbing Huang, Peihao Chen, Mingkui Tan, and Chuang Gan. Dense regression network for video grounding. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, June 2020. 1

[48] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2018. 2, 6, 7

[49] 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 *Proc. IEEE Int. Conf. Comp. Vis.*, 2017. 2

[50] Junjie Zhang, Qi Wu, Jian Zhang, Chunhua Shen, and Jianfeng Lu. Mind your neighbours: Image annotation with metadata neighbourhood graph co-attention networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 2956–2964, 2019. 1

[51] Yifan Zhang, Hanbo Chen, Ying Wei, Peilin Zhao, Jiezhang Cao, et al. From whole slide imaging to microscopy: Deep microscopy adaptation network for histopathology cancer image classification. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 360–368. Springer, 2019. 1

[52] Yifan Zhang, Ying Wei, Peilin Zhao, Shuaicheng Niu, Qingyao Wu, Mingkui Tan, and Junzhou Huang. Collaborative unsupervised domain adaptation for medical image diagnosis. In *Medical Imaging meets NeurIPS*, 2019. 1

[53] Yifan Zhang, Peilin Zhao, Qingyao Wu, Bing Li, Junzhou Huang, and Mingkui Tan. Cost-sensitive portfolio selection via deep reinforcement learning. *IEEE Trans. Knowl. Data Eng.*, 2020. 1

[54] Minfeng Zhu, Pingbo Pan, Wei Chen, and Yi Yang. Dm-gan: Dynamic memory generative adversarial networks for text-to-image synthesis. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 5802–5810, 2019. 2

[55] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 875–886, 2018. 1