PlanNet: A Generative Model for Component-Based Plan Synthesis

Qiang Fu*, Shuhan He, Xueming Li, Hongbo Fu

Abstract—We propose a novel generative model named as *PlanNet* for component-based plan synthesis. The proposed model consists of three modules, a wave function collapse algorithm to create large-scale wireframe patterns as the embryonic forms of floor plans, and two deep neural networks to outline the plausible boundary from each squared pattern, and meanwhile estimate the potential semantic labels for the components. In this manner, we use *PlanNet* to generate a large-scale component-based plan dataset with 10K examples. Given an input boundary, our method retrieves dataset plan examples with similar configurations to the input, and then transfers the space layout from a user-selected plan example to the input. Benefiting from our interactive workflow, users can recursively subdivide individual components of the plans to enrich the plan contents, thus designing more complex plans for larger scenes. Moreover, our method also adopts a random selection algorithm to make the variations on semantic labels of the plan components, aiming at enriching the 3D scenes that the output plans are suited for. To demonstrate the quality and versatility of our generative model, we conduct intensive experiments, including the analysis of plan examples and their evaluations, plan synthesis with both hard and soft boundary constraints, and 3D scenes designed with the plan subdivision on different scales. We also compare our results with the state-of-the-art floor plan synthesis methods to validate the feasibility and efficacy of the proposed generative model.

Index Terms—Generative model, wave function collapse, floor plan synthesis.

1 Introduction

Designing plausible plan layouts is an essential part of various design tasks like urban design [1], architectural design [2], and interior design [3], [4]. Even on different scales, these applications face the same problem that how to compartmentalize the space to guarantee functionality and diversity. Recently, AI-assisted techniques that automatically generate floor plans for residential buildings with input boundaries, become one of the hottest research [5], [6].

One way to tackle the floor plan generation problem is to straightforwardly divide inner walls given a hard outer room boundary. There have been recent efforts in this direction, with varying levels of success (e.g., [5], [6]). However, as an important step in architecture design, creating the floor plan configuration, i.e., an outer room boundary, also requires lot of manual workload for architects. In fact, architectural shape, floor plan boundary, and interior space division impact each other in an architect's design workflow. For example, when designing a building or a living quarter on a plot of land, the boundary of the land is not necessarily the boundary of the design result. Therefore, heuristic models that generate floor plans with differential configurations but are constrained inside the same outer boundary, what we call soft boundary constraints, are also useful for architectural design.

- * *Corresponding author (Email: fu.john.qiang@gmail.com)*
- *Q. Fu, S. He, and X. Li are with the School of Digital Media and Design Arts, Beijing University of Posts and Telecommunications, China.*
- *H. Fu is with the School of Creative Media, City University of Hong Kong.*

Another way of floor plan generation is example-driven. The example-driven idea is popular in research on 3D modeling, e.g., for man-made objects [7] and indoor scenes [8], [9]. The main challenge of employing the example-driven idea on floor plan generation is how to collect huge numbers of floor plans with both various shapes and plausible layouts. Since collecting a large amount of data of floor plans produced by human designers is expensive, especially for different scales and scene categories, automatic plan generation methods could mitigate this issue and make floor plan examples cheaper and more accessible. Besides, the generation model should be able to make variations of both inner and outer walls. For the generated floor plans, pre-room semantic labels should also be provided to assist architects in further design. Since floor plan generation is actually a way of space division, if we determine the proper semantic labels for the generated floor plans, they could also be used for larger outdoor scenes like flats or villas. It could enable the floor plan generation model to tackle scenes on multiple scales.

To address the above problems, we propose *PlanNet*, a generative model for component-based plan synthesis. Our method is based on two major insights: i) to make generated plans be more general, the generative model should be decoupled with specific scene categories, i.e., using no priors related to any scene categories; ii) the generative model should be able to automatically create large numbers of plan examples with plausible inner layouts and adequate variations, thus supporting example-driven floor plan design. To this end, we jointly use the wave function collapse algorithm (WFC for short) [10], which is efficient in automatic map generation for computer games, and two deep networks to design a generative model. Firstly, we hierarchically construct a set of binary pixel tiles, and use the WFC to

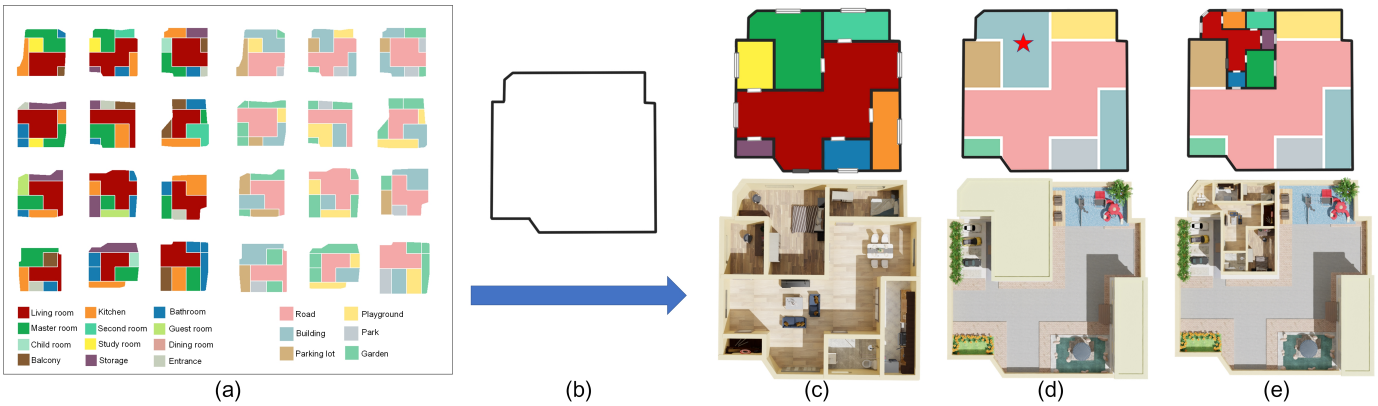


Fig. 1. Our PlanNet learns to generate a large-scale dataset of component-based floor plans (a). Given an input boundary (b), our approach retrieves dataset plan examples to guide the synthesis of indoor and even outdoor 3D scenes (c-d)-Bottom. We also have an interactive subdivision workflow to further subdivide the plan components (e.g., the one highlighted with a red star at (d)-Top.) to create more complex plans (e.g., (e)).

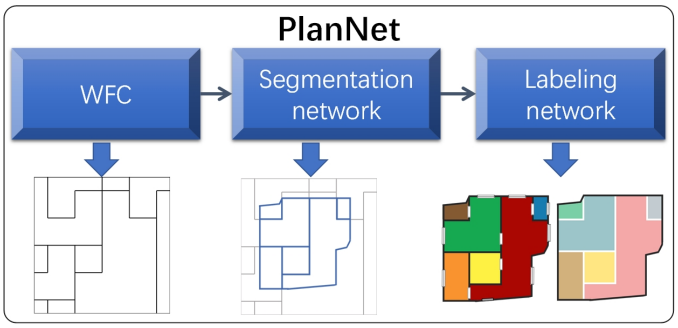


Fig. 2. Elements of *PlanNet*: a WFC module to generate various squared embryonic plans, a segmentation network module to extract a floor plan (highlighted in blue) from the embryonic plan, and a labeling network to suggest potential semantic labels for the floor plan components.

generate embryonic plans, i.e., squared wireframe patterns that have explicitly divided sub-spaces and can be further outlined with various boundaries. Then two deep neural networks are employed with the RPLAN dataset provided by [5] as the training set, aiming at retaining partial of the square patterns and suggesting plausible semantic labels for the divided sub-spaces (see the pipeline in Figure 2). In this manner, the embryonic plans can be converted to floor plan examples. The WFC and two networks jointly create a large-scale plan dataset with adequate variations. Specifically, our network architecture consists of a segmentation network based on the U-net to outline the boundary given a square pattern (i.e., embryonic plan), and a labeling network based on graph convolution to suggest potential semantic labels for the components of the generated floor plans. In this manner, we generate a large number of floor plan examples (10K) with adequate variations as well as the per-room semantic labels (Figure 1-(a)). For indoor scenes, we use adjacency relations to set doors on the inner walls and windows on the outer walls, once the floor plans are generated. We also make some attempts that establish semantic relations between indoor and outdoor scenes, so that the generated floor plans could be used for outdoor scenes such as flats or villas.

For the application of plan design under boundary constraints, we employ an example-driven method that transfers the space layout from the dataset floor plans to the hard or soft boundary constraints based on their configuration similarities. Given a boundary (Figure 1-(b)), we retrieve similar examples from the floor plan dataset and map the layout of a user-specified retrieved floor plan to the input. By setting different scale factors, designers can use our interface to recursively subdivide the plan, thus creating a series of floor plans, which are useful for not only residential buildings but also outdoor scenes (Figure 1-(c-e)).

We claim three main contributions in this work: 1) a joint framework to combine the wave function collapse algorithm and deep neural networks for automatic floor plan generation; 2) two networks, which tackle component-based plan synthesis and semantic estimation for each component; 3) an interactive workflow that assists users to create floor plans under hard or soft boundary constraints, and supports recursive subdivision as well. We demonstrate the effectiveness of our method through various experiments and comparisons with the state-of-the-art methods [5], [6], [11], [12], [13].

2 Related Work

In this section, we first review the existing studies on layout generation in various application fields. Then, we introduce some recent works that leverage deep learning to tackle the problem of layout and floor plan generation. Lastly, we discuss some related works on architectural and interior design to show how plan layouts facilitate 3D modeling tasks.

Layout Generation. In general, layout generation is a task to arrange a group of components for certain purposes (e.g., [14], [15]). Such a task is essential to many applications, such as graphic design [16], [17], furniture design [18], urban or architecture design [2], [19], [20], [21], computer games [22], and indoor scene synthesis [3], [23], [24]. However, due to different scales and design purposes, these works focus on specific objects and design rules. For example, urban design methods consider more about the connectivity and layout of underlying networks (e.g., street networks [1]), and floor plan design for architecture focuses on generating a valid interior space division under a set of high-level constraints, e.g., room size, room position, etc. (e.g., [25]),

while indoor scene synthesis tackles the arrangement of indoor objects, aiming at understanding and representing the structure of scenes, and preserving the relations of co-existing indoor objects (e.g., [26]). Moreover, considering scenes with convertible layouts have also been studied [27], floor plans with various layouts become a rising demand. Recently, focusing on the space division of indoor scenes, a series of automatic layout generation models have been proposed (e.g., [12], [13], [28]). Most of them rely on the relation graphs of indoor scene elements (i.e., room categories). In our work, we expect to develop a more general model for scene layout generation on different scales (e.g., apartments, buildings, and living quarters). Hence, our method does not rely on any priors associated with scene categories. Even though our method is mainly for the floor plan generation of architecture, we can easily extend our method to plans on different scales, such as a living quarter with multiple buildings. We also provide an interactive workflow to support the subdivision of the generated plans for more complex designs.

Deep Networks for Interior Design. Recently, deep-learning-based methods have been successfully introduced to address the problems of interior design. Tasks such as object arrangement for indoor scenes have been tackled by deep learning methods. For example, Wang et al. [29] proposed a deep convolutional neural network to learn priors from a large-scale database for indoor scene synthesis. Li et al. [30] presented a recursive neural network to generate plausible 3D indoor scenes. Wang et al. [31] presented a framework for indoor scene synthesis by combining a high-level relation graph representation with spatial prior neural networks. Fu et al. [32] proposed to leverage the deep reinforcement learning method to address the layout generation problem. These works focus more on the object-level layouts of indoor scenes.

On the other hand, the problem of floor plan generation for residential buildings can also be addressed by deep learning. For instance, Wu et al. [5] introduced a deep network for the generation of floor plans of residential buildings. Hu et al. [6] introduced a learning framework for automated floor plan generation by combining a graph neural network and user-in-the-loop designs to set constraints. Our method is more related to the floor plan generation methods (e.g., [5], [6], [11]). However, since the training dataset often has floor plans with the same outer boundary but different inner layouts, using an outer contour as the input might make the convolutional network fail to converge. These methods thus used several strategies, e.g., starting the generation from the living room or using a relation graph as constraints. Instead, our generative model does not rely on boundary inputs as the constraints to create plans. It can generate a large-scale floor plan dataset with various configurations and space layouts as an alternative to data produced by human designers. For the scenario when certain boundary constraints are required, we employ an example-driven method to synthesize plans with the same or similar boundaries as the inputs.

Example-Driven Design. Example-driven modeling is popular in many computer graphics problems, such as man-made object synthesis and indoor scene modeling. The structure or style of examples can be used as references to decrease the difficulty of 3D modeling. For example, structure examples of man-made objects can guide assemble-based object synthesis and even across different categories (e.g., [33], [34]). Design examples can also provide priors such as component style and fabricable template to facilitate 3D modeling (e.g., [7], [35]). For indoor scenes, examples of object arrangement can directly guide layout generation (e.g., [8]). Some works leveraged indoor scene images [36] or floor plans [37] as examples to transfer the color styles or layouts from the examples to synthesized indoor scenes. Besides, plan examples can be combined or deformed to generate new ones. For instance, Aliaga et al. [38] extracted a street network and per-parcel aerial-view images from real-world urban layouts, and employed an example-based approach to synthesizing new urban layouts based on data from the example layout fragments. Peng et al. [39] proposed a method to generate layouts via tiling a domain with a set of deformable templates. Motivated by [38], [39], our method also employs the example-driven method for layout generation of floor plans. The difference is, we jointly use the WFC and two networks to generate plan examples as the dataset for example-driven plan synthesis. Moreover, the plans synthesized by our method can be applied to both indoor and even outdoor scenes.

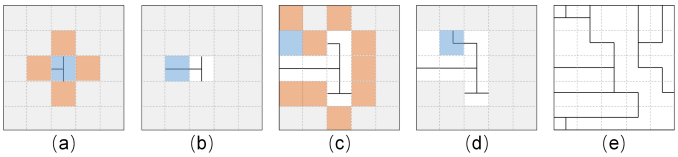


Fig. 3. An example of the WFC process. We illustrate the final embryonic plan (e), and two pairs of adjacent intermediate states, i.e., (a-b) and (c-d). Note that after the eigenstate of an element (blue) is determined, the WFC will randomly select one of the neighbors (orange) around all determined tiles in the next iteration.

3 WFC-based Embryonic Plan Generation

Wave function collapse is a concept in quantum mechanics. It occurs when a wave function initially in a superposition of several eigenstates reduces to a single eigenstate due to interaction (i.e., observation) from the external world. In computer graphics, an algorithm that imitates this phenomenon has been proposed to facilitate 2D and 3D virtual scene synthesis [10]. The WFC algorithm can randomly generate large patterns, since it addresses the problem of how to splice small tiles to form a large pattern, especially when certain tiles can be used for multiple times.

For the WFC algorithm, the scale of tiles should be restricted to an appropriate range to ensure plausible content of generated plans. If we use a boundary input to constrain an initial state of a plan and then start the WFC algorithm, the WFC would easily fail since only a few tiles with superposed eigenstates exist. Therefore, we leverage the WFC algorithm to first generate wireframe patterns as the embryonic plans (e.g., Figure 3), and then employ a convolution neural network to extract plausible floor plans from the embryonic plans. Aiming at improving the training efficiency of the neural network, we limit the resolution of the embryonic plan, which is the network input. Since we focus on indoor and simple outdoor scenes, our goal of

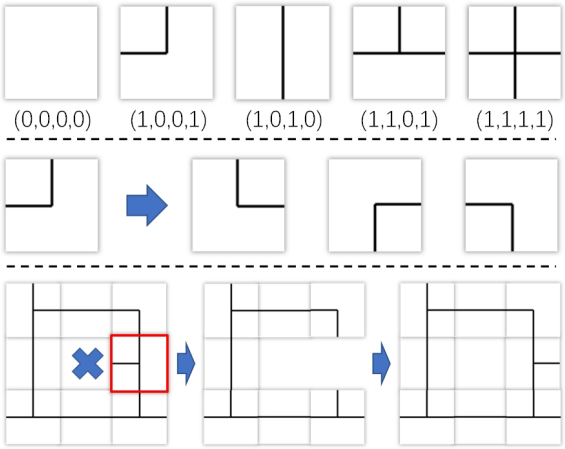


Fig. 4. **Top:** five kinds of tiles and the corresponding vectors. **Middle:** a tile can be rotated with $90°/180°/270°$ when performing the WFC algorithm. **Bottom:** when a tile (red box on the left) makes its neighbor be zero eigenstates (i.e., no available tile), backtracking can change the eigenstate of the previous tile to enable the WFC to continue.

plan synthesis is different from that for urban plans often involving very dense content. Therefore, we use a fixed proportion between the plan resolution and line width on the plan. For these reasons, we empirically set each tile to be a 70×70-pixel wireframe image with its line width set to 2 pixels. Totally, we choose 5 basic kinds of tiles to participate in the pattern generation, as illustrated in 4-(Top). We encode each kind of tile as a feature vector with four binary elements. Specifically, we denote the element as T_k^i for the k-th kind of tile about its i-th side. $T_k^i = 1$ indicates there is a line that crosses such a side of the tile, and $T_k^i = 0$ otherwise. In this manner, we can define the adjacency relation between two kinds of tiles as follows:

$$\mathcal{R}(T_k^i, T_m^j) = \begin{cases} 1, \text{ if } T_k^i = T_m^j \\ 0, \text{ otherwise} \end{cases}, \tag{1}$$

where $\mathcal{R}(T_k^i, T_m^j) = 1$ indicates the i-th side of the k-th tile kind can be combined with the j-th side of the m-th tile kind. Note that for two sides that are not possible to combine, e.g., combining the top side of one tile to the left side of the other one, we allow the tile to be rotated so that such two sides can be combined. This also means the basic tiles can be rotated to change their feature vectors, thus leading to more eigenstates (see Figure 4-(Middle)).

In our implementation, an embryonic plan consists of 5×5 tiles, i.e., 350×350 pixels. We represent each embryonic plan by a matrix with 5×5 elements, with each denoting a kind of tile at the associated position of the embryonic plan. In the beginning, each element can have all tile types representing all eigenstates. Therefore, once an element of the embryonic plan is determined as tile kind k, its eigenstates will reduce to a single one, and the eigenstates of its neighbor element about the i-th side will also reduce to a set of tile kinds $\{m\}$, where $\exists j, \mathcal{R}(T_k^i, T_m^j) = 1$, based on Equation 1.

More specifically, the WFC algorithm starts from a random element, and randomly reduces its eigenstates to a single tile kind. We randomly rotate the tile with $0°$, $90°$, $180°$, or $270°$, and then update the eigenstates of its four-side neighbors as aforementioned. In each iteration, the element with the fewest eigenstates is picked. We randomly choose a tile kind from all eigenstates of the picked element and then update its neighbors with more than one eigenstate. Note that the chosen tile might also need rotation to satisfy Equation 1.

During this iterative process, a certain element might have zero eigenstate after updating. Namely, from all of its eigenstates, there is no tile that can be rotated by the same angle to be combined with all of its neighbors. If so, we perform backtracking on the matrix and then choose another eigenstate to continue the iteration (see Figure 4-(Bottom)). Finally, all elements of the matrix have been reduced to a single kind of tile, and thus an embryonic plan is created. Since this is a random process, we can use the WFC algorithm to generate embryonic plans to construct our dataset. In our implementation, we generate 10K embryonic plans in total. On average, it takes 49 steps per embryonic plan construction. For 10K embryonic plans, the zero eigenstate occurs 623 times. All of these zero eigenstate can be addressed with only single-step backtracking.

4 Segmentation and Labeling Networks

Aiming at collecting the plan examples with adequate shape and layout variations, we propose to perform two deep neural networks on the embryonic plans in two stages. The first one focuses on the task of dividing a region with a plausible boundary from a squared embryonic plan, while the second is proposed to estimate the semantic labels for each part of a floor plan.

4.1 Network Architecture

As illustrated in Figure 5, we use a U-Net architecture [40] as the segmentation network in the first stage. It has a contracting path (left side) and an expansive path (right side) to convert an input image to an output mask. In our implementation, the contracting path consists of repeated application of two 3×3 convolutions, a rectified linear unit (ReLU), and a 2×2 max pooling operation with stride 2 for downsampling. The expansive path has a symmetrical structure but with upsampling operations. Note that the number of feature channels should also be doubled at each downsampling step, while halved at each upsampling step. Since the U-net employs an overlap-tile strategy to tackle the image segmentation in local regions, there is no need to resize the input image to the same size as the first layer. The purpose of the segmentation network is to outline a plausible boundary inside the input squared embryonic plan. Namely, the network input is a pattern, while the output is a two-channel feature map encoding the probabilities of both the foreground and background per pixel. In our implementation, the foreground provides a plausible boundary for the given embryonic plan.

Prior to the second stage, we perform a plan refinement operation, which will be discussed later. An adjacency graph will be extracted from the refined floor plan. Each node of the graph represents a certain part of the plan, while the edge between two nodes indicates that such two parts are adjacent. Then we feed it to the network of the second stage.

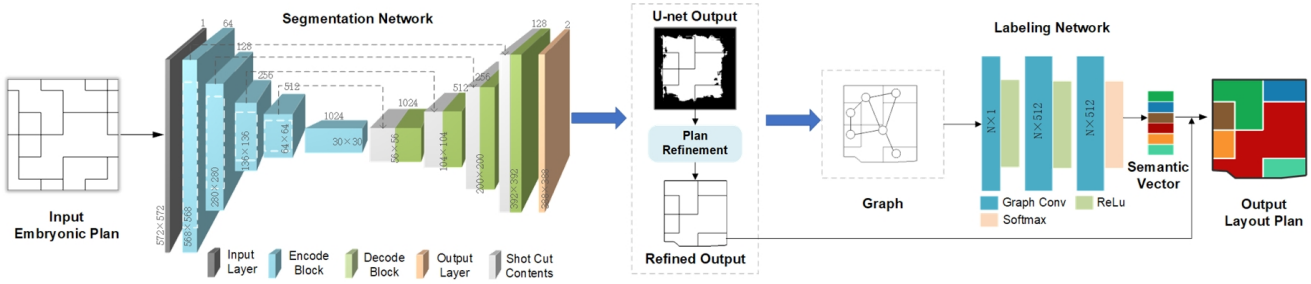


Fig. 5. The architectures of the segmentation and labeling networks. The input is a squared embryonic plan generated by WFC, and the network output is the associated boundary and the semantic labels for individual components.

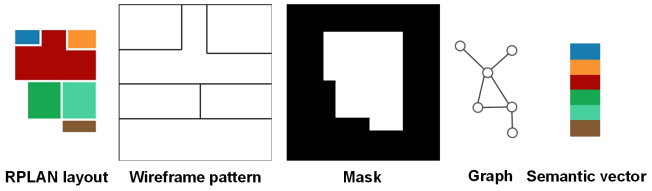


Fig. 6. An example of the training data extracted from the floor plans of the RPLAN dataset (Left), including the wireframe pattern, binary mask, adjacency graph, and semantic vector.

Here we adopt the graph convolution network [41] as the labeling network, which takes the adjacency graph of a plan as input and outputs a vector of semantic labels according to each node of the graph (Figure 5). More specifically, each training data provides an adjacency graph with N nodes represented as a vector with N elements as the network input. We first initially set up the vector by using the area ratio of each component to the whole plan for each element. After three layers of graph convolution and a soft-max layer, the network outputs an $N \times 12$ matrix, where each row is a 1×12 vector denoting the probabilities of the node with respect to 12 kinds of semantic labels. We use cross-entropy as the loss function.

4.2 Training of Two Networks

We use the RPLAN dataset [5], a large-scale dataset containing more than 80K real floor plans with dense annotations, to prepare our training data. Since two networks of our method are independent, we train them separately. For each RPLAN data (e.g., Figure 6), we extend it to a square wireframe image to make it the same as our network input, where black represents the wall and white represents the room. The original boundary of the floor plan is also recorded as a binary mask. Meanwhile, we extract an unlabeled adjacency graph from each RPLAN layout, and encode the corresponding annotations to a vector of semantic labels.

In total, we have 80,788 square layout images with masks to train the segmentation network. We randomly pick 60,591 images as the training set, 13,455 images as the test set, and 6,742 images as the validation set. Since dividing a boundary from the input is a binary classification problem, we use binary cross-entropy as the loss function for training. The segmentation network is trained through a deep learning library PaddlePaddle [42] with batch size = 64, epoch number = 10, learning rate = 0.001, and Adam optimizer.

On the other hand, for 80,877 graph-label pairs extracted from the RPLAN dataset, we use the unlabeled graph as the network input and the vector of semantic labels as the output to complete supervision training. Hence, we randomly pick 70,000 and 10,877 graph-label pairs as the training set and test set, respectively. Such a graph convolution network is trained through the PaddlePaddle framework with batch size = 512, epoch number = 100, learning rate = 0.001, and Adam optimizer.

4.3 Plan Example Dataset

We construct the plan example dataset by jointly employing the WFC and the two networks. Aiming at balancing the diversity of the plans and the retrieval efficiency, we limit the size of the dataset to 10K. Specifically, we first generate 12,150 embryonic plans via the WFC. Then we use the trained segmentation network to delimit the outline for each embryonic plan. Note that we also have a plan refinement operation to improve the quality of the network-generated boundaries, and filter out the undesired low-quality plan results from the dataset. After that, the adjacency graphs of the generated floor plans can be calculated, and then used for semantic estimation by the trained labeling network. Each floor plan can have two sets of semantic labels corresponding to indoor and outdoor scenes, respectively. For better usage on indoor scenes, we also employ an appendage suggestion operation to determine the potential positions of windows and doors in the plan examples. Totally, we collected 10,028 plan examples in our dataset for plan synthesis. To improve the efficiency of the next retrieval stage, when constructing the dataset, we extract the shape features, i.e., vectors consisting of the *area* and *perimeter* of the plan examples, and count the number of their components. Such a simple shape feature can be used to explore a small set of candidate floor plans from a large-scale dataset in a short time, then we can employ a time-consuming fine comparison between the input and the small candidate set to retrieve floor plans that suit the input.

Plan Refinement. The main quality issues of the generated plan examples exist in the connectivity and completeness of the region, and smoothness of the boundary. We propose two refinement mechanisms to fill holes and fit the boundary to a polygon (Figure 7-(b-c)). Specifically, we first perform morphological open and close operations on the network-output plans, in order to eliminate the noise and fill the unexpected holes. Then, we employ the Douglas–Peucker algorithm [43] to generate the polygon

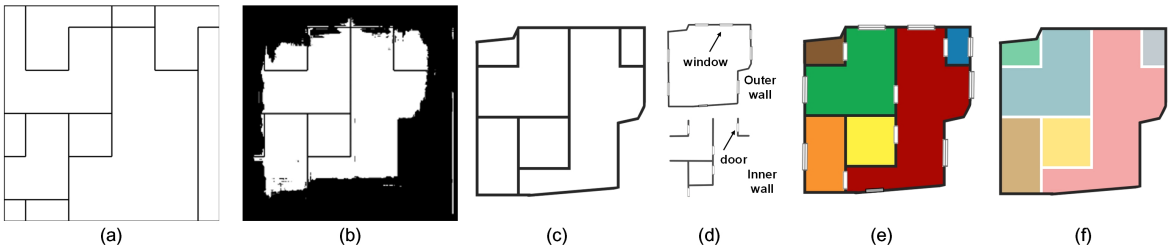


Fig. 7. After the segmentation network outlines a rough boundary (b) for a given embryonic plan (a), the plan refinement operation then fills the holes, and fits the plan with a polygon boundary (c). For indoor scenes, we also estimate the positions of windows and doors (d) on the outer and inner boundaries of the plan (e). For outdoor scenes, the refined plans can be directly used with no appendages (f).

boundary of each plan. In addition, since the shapes of certain floor plans might be over-changed after the refinement, we leverage the IOU ("intersection over union") of the areas between the floor plans before and after refinement to filter out the over-changed plans. In our implementation, we set a threshold to 70% to filter out the plans whose IOU values are below the threshold.

Appendage Suggestion. For the application of floor plan design, we attempt to suggest plausible appendages, i.e., windows and doors, on both the outer and inner boundaries of the plan examples (e.g., Figure 7-(d)). We always choose the central positions on the walls for the appendages. For simplicity, we assume that windows only exist on the outer boundary, but we randomly choose one position on the outer boundary of the largest component (which could be the living room) to set the entrance door. Afterward, we go through the inner boundaries between components, to set interior doors. Since the positions or even the numbers of the appendages might require further refinement, in our user interface, the windows and doors can be interactively modified to satisfy the design purposes.

5 Example-Based Plan Synthesis

The constructed plan example dataset can be used to facilitate component-based plan synthesis. We develop a user interface to assist users in designing plan layouts given a hard or soft boundary constraint. After users input a polygon to our system or specify a certain component for subdivision, our method employs an example-driven approach to retrieving proper plan examples from the dataset with component-level semantic labels to guide the plan synthesis.

User Interface. Figure 8-(a) illustrates our web-based user interface developed through the Flask framework. We choose a web-based interface so that we can deploy the plan example dataset on the server side and make the client side lightweight. The input panel of the UI allows users to create a polygon by setting the vertex positions, and choose it as a hard or soft boundary constraint. Our system also employs third-order Bezier curves to allow users to create curved inputs (e.g., Figure 8-(b)). Users can edit the position of the two control points between two endpoints to change such a curve. After that, a series of plan examples from the dataset are retrieved and presented in the example panel for users to choose from. The order of the listed plan examples depends on their configuration similarities to the input boundary. In addition, to enable users to control the synthesized plans,

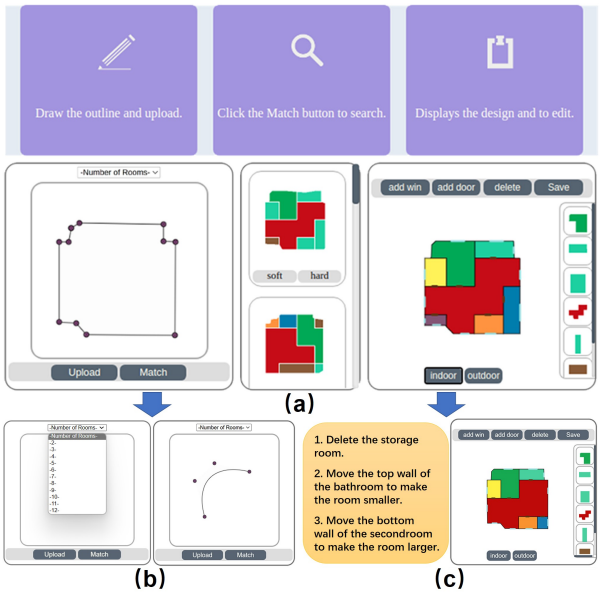


Fig. 8. **(a)** Our user interface consists of a function menu (Top), an input panel to draw a polygon (Bottom-Left), an example panel to display the retrieved plan examples (Bottom-Middle), and an output panel to show the generated plan for further editing (Bottom-Right). **(b)** Users can specify the number of components and input curved boundary through the input panel. **(c)** An example that shows how users follow the suggestion (Left) to modify the floor plan (Right).

our system UI allows users to specify the amount of the plan components as a constraint input. Since the number of components of all plan examples in the dataset have been recorded, such input can ensure the retrieval plan examples meet the user demands.

Our system has a modification suggestion mechanism for certain low-quality rooms to assist users to refine their designs. In the output panel, our system provides two ways for users to refine the designed plans. First, users can manually edit certain inner walls to change the component shapes or merge two components (e.g., Figure 8-(c)). Other elements, such as the windows and doors, are also editable by users. Note that the user-edited plans can be uploaded to our dataset as new examples for enriching the diversity of our dataset. Besides, if users choose one component on the generated plan, its boundary can be used as the input polygon for further subdivision. More complex plans on different scales can be generated via recursively subdividing components into smaller ones (e.g., Figures 1 and 17).

Plan Retrieval and Synthesis. Each plan example in our

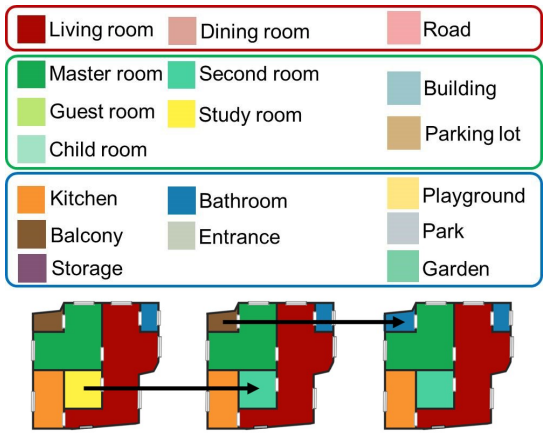


Fig. 9. **Top:** We gather the semantic labels of both indoor and outdoor scenes into three groups, illustrated in the red, green, and blue boxes, respectively. **Bottom:** an example to show how the random selection algorithm works.

dataset has boundary information, i.e., a binary image in which the white region is inside while the black is outside, hence we convert the user-input polygons or inputs with curved boundaries to the same kind of binary image for retrieval. We first normalize the white regions of both the masks of the dataset plans and the mask created by the input boundary into the same scale. Since these white regions represent the configurations of the plans, we directly use the IOU value between the white regions of the dataset plan and input boundary as the metric to measure the configuration similarity. Note that since the pair-wise IOU-based comparison is time-consuming, we separate the plan retrieval into two phases: a primary retrieval that leverages the distances of the shape features between the input boundary and the dataset floor plans to explore Top-1000 plan candidates with smaller distances, and a fine retrieval that leverages the IOU-based comparison to explore floor plans from the 1000 candidates similar to the input. This could speed up the retrieval stage and avoid the huge computational effort as a result of the dataset size increasing. Considering most of the plans are axis-aligned, we also rotate the input boundary by $0°$, $90°$, $180°$, and $270°$, respectively, to obtain the max IOU value as the comparison result. The retrieved dataset plan examples are ranked based on their IOU values compared to the input from high to low and return Top-10 results on the UI for users to select.

Moreover, if the polygon input is a hard constraint, we need to map the layout of the chosen plan example to the input. This is implemented by a simple reshaping approach. More specifically, we first align the input polygon and the user-selected retrieved plan example by the associated rotation angle. For each corner point of a plan example, we find the closest pixel on the input polygon to establish the correspondence. To preserve the shape of the plan components, we only seek for the pixels closest to the corner points in the horizontal and vertical directions. Then, we set the positions of the corners to their corresponding boundary pixels to reshape the plan example and fill potential holes to make the shape of the plan fit the input polygon well. If the polygon input is used as the soft boundary constraint, we use the input boundary to clip the retrieved dataset plan example. Namely, the plan example regions outside the input polygon would be clipped, thus changing the configuration of the dataset plan example to meet the user demand. Since the network-output plan examples might have too many corners even after plan refinement, our system highlights the non-right angle corners and allows users to remove certain of them to optimize the plan boundary.

Lastly, we are to leverage the pre-stored semantic labels of each plan example to determine the semantics of each component for the final plan result. To ensure the diversity and expandability of the generated plans, we propose a random selection algorithm to make variations on the semantic labels of the components. More specifically, we focus on the interchangeability among different scene types, namely, groups of semantic labels that all suit the same part of a plan. For example, a study room and a guest room are interchangeable when considering the function of a certain room. In Figure 9-Top, we show how we define the groups and their members.

For the output semantic labels of our labeling network, the selection algorithm randomly replaces a certain component label with another one from the same group in turns (e.g., Figure 9-Bottom highlighted by the black arrows). We set the probabilities for the labels of certain components following these criteria: i) the probabilities of semantic labels in the same group are associated with the rank of component size on the plan, e.g., larger components would be more likely to be master rooms rather than child rooms; ii) labels such as master room, kitchen, and bathroom should each have at least one component present, while labels such as study room and kitchen should be at most one component. Moreover, our user interface also allows users to set the scenario as outdoor scenes. For this purpose, we set the correspondence between the semantic labels of indoor and outdoor scenes based on their functional similarity. Since there are few types of semantic labels for outdoor scenes, if the neighbors of a component are already labeled as buildings, we just reduce the probability of building labels for that component.

Modification Suggestion. To help users enhance their floor plans, we develop a mechanism for modification suggestion based on established design guidelines in the building sector, such as ISO 9836:2017 and GB/T 50362-2022. These guidelines consider a room's aspect ratio, smallest area, and shortest edge length when determining its quality. In other words, a high-quality room should have enough space and illumination so that it should not be too narrow or small. We use three metrics to evaluate a floor plan's rooms: aspect ratio (R_a), component area ratio (R_c), and relevant edge length (R_e). To calculate these metrics, we first find the short and long edge lengths for each room and the entire floor plan. R_a is the ratio of a room's short to long edge lengths. R_c is the ratio of a room's area to the entire floor plan's area. R_e is the ratio of a room's edge length to the corresponding edge length of the entire floor plan. Note that we can use the minimum values of these factors for all rooms in a floor plan as the lower limits to evaluate the quality of a floor plan.

Using these room metrics, we created a new mechanism to suggest changes for rooms that may need adjustments. We set the lower limit thresholds for R_a, R_c, and R_e to

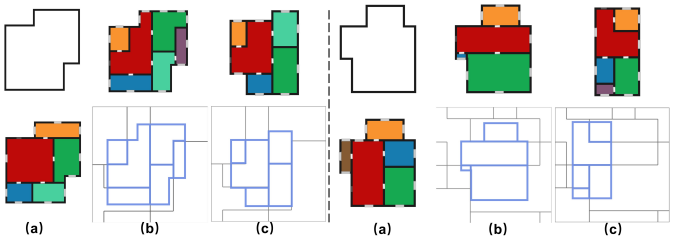


Fig. 10. We compare our method with direct online generation in two cases (Left and Right). In each case, we show the input boundary (a-Top), our result (a-Bottom), the online-generated floor plan and its segmentation from the embryonic plan (b), and a more proper outer boundary for the same embryonic plan suggested by our segmentation network (c). Note that we constrain the outer walls of plans in (c) to be straight for comparability.

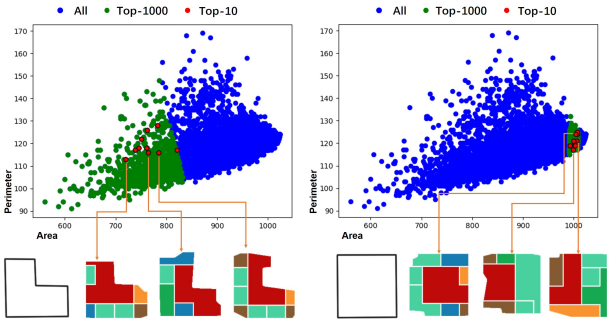


Fig. 11. Two examples of the floor plan retrieval process. We illustrate the feature space of all dataset floor plans, highlight the Top-1000 primary retrieval results in green, and highlight the Top-10 fine retrieval results in red. We also show certain according floor plans in the bottom, next to each input boundary.

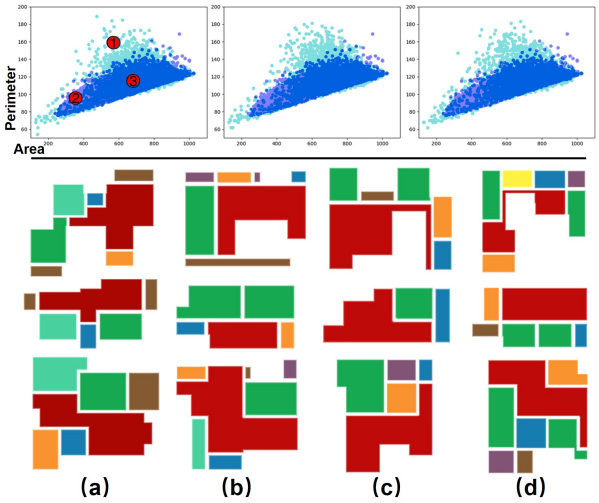


Fig. 12. **Top:** Comparisons between the RPLAN dataset (indigo) and three new datasets of 10K plans generated by our method, respectively (cyan). Note the overlapping regions are blue. **Bottom:** We show the plan examples are closest to position #1, #2, and #3, respectively from the top to bottom. In each row, we show the RPLAN floor plan (a) and our results (b-d) in the same order as their datasets in the top figure.

TABLE 1
Dataset comparison based on the layout statistics.

Dataset	Avg. R	Avg. C	Avg. A
RPLAN	6.79	5.31	0.48
Ours	6.37	4.28	0.44

0.5, 0.1, and 0.2, respectively. Our system can suggest three types of modifications: deleting a room (by merging it with a neighboring room), or moving a wall (top, bottom, left, or right) to make the room larger or smaller. If a room's metric falls below the threshold, our system will suggest moving certain walls or even deleting the room. To prevent changes from affecting other rooms besides the living room, our system will only suggest making a room larger if its neighbor that shares the wall is the living room.

6 Experimental Results

In this section, we first report the quantitative analysis on two networks of the proposed *PlanNet* and show some retrieval results, to evaluate the constructed plan example dataset. Then we show several plan synthesis results under both hard and soft boundary constraints, and present some 3D scene modeling examples guided by the designed plans. We also compare our method with the state-of-the-art plan generation methods [5], [6] and conduct a user study to evaluate these methods.

6.1 System Evaluation

Firstly, we compare our retrieval-based synthesis method with online floor plan generation. Specifically, we use embryonic plans randomly generated by the WFC algorithm to create floor plans by aligning the input boundary with the center of the embryonic plan and segmenting it. This eliminates the need for a plan dataset. However, due to the randomness of WFC-generated embryonic plans, the input boundary may not always fit the online-generated embryonic plan, resulting in unstable floor plan quality. For instance, Figure 10 shows two cases where an unsuitable embryonic plan (e.g., the right case) results in an improper inner layout. In contrast, our segmentation network, which is trained by RPLAN and leverages interior design priors to determine more proper boundaries for each embryonic plan, ensures high-quality floor plan designs. Since the WFC algorithm may not always produce an appropriate embryonic plan for the given boundary, instead of using the WFC algorithm to directly generate a floor plan for the given boundary, we use WFC to create a dataset of floor plans and retrieve suitable ones. Then, We evaluate the proposed generative model through quantitative analysis in two stages. For the segmentation network, we achieve an average IOU of 84.6% on the test set for the task of boundary segmentation. The reason the IOU result is not very high is mainly that there exist similar embryonic plans but different layouts in the RPLAN dataset. Namely, multi-solvability exists in the task of delineating the boundaries of floor plans. However, such accuracy is sufficient for the trained segmentation network to obtain plausible plan boundaries. For the labeling network, we achieve average accuracy 51%, precision 48%, and recall 49% on the test set with respect to 12 semantic labels. Note that since some kinds of semantic labels are interchangeable, their deep features extracted from the graph convolution operation

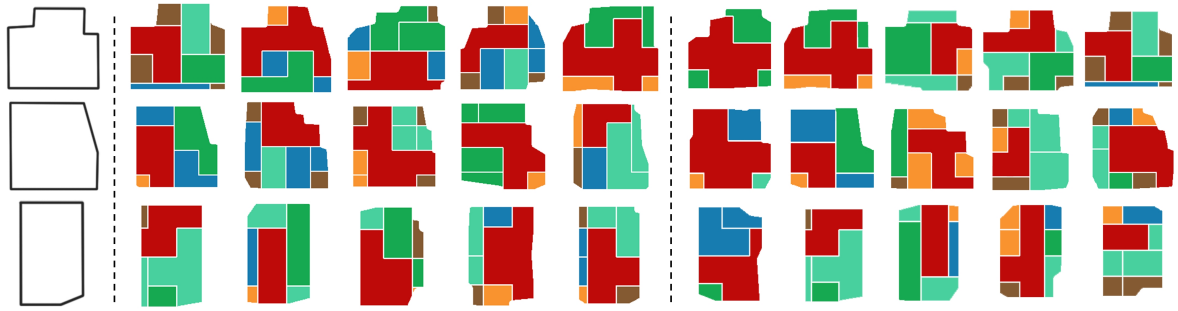


Fig. 13. **Left:** polygon boundary inputs. **Middle** retrieved plan examples ranked by their IOU values associated with the input (from left to right). **Right:** retrieved plan examples with the user-specified component amounts ranging from 4 (left) to 8 (right).

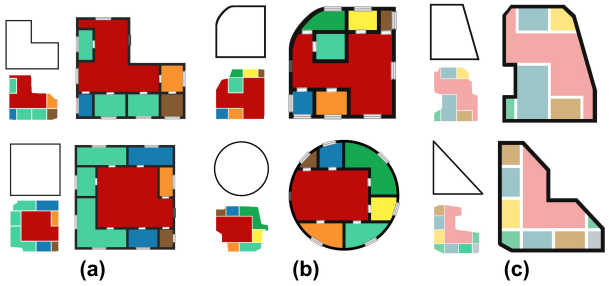


Fig. 14. Plan synthesis results with hard boundary constraints (a & b) and soft boundary constraints (c).

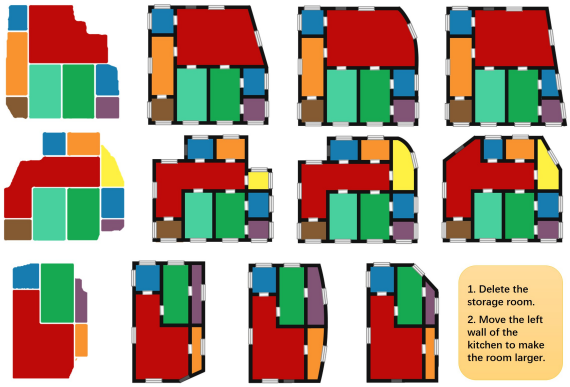


Fig. 15. Plan synthesis results by using the same dataset plan examples (Left) with different input boundaries. We show the modification scheme suggested by our system for the bottom case.

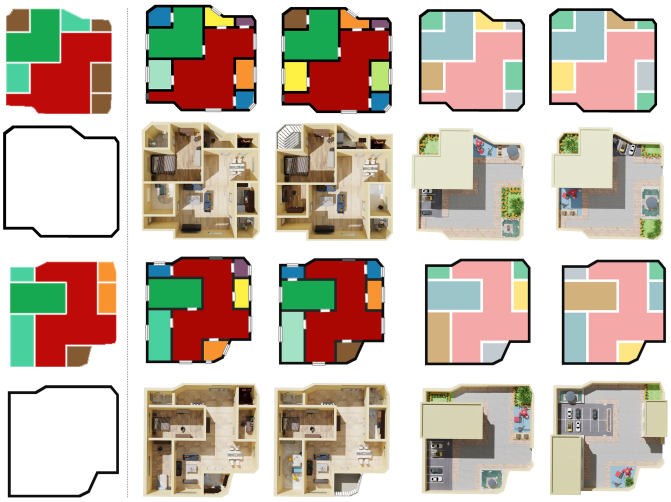


Fig. 16. By using the random selection algorithm, the given boundaries and the retrieved plan examples (Left) can lead to more various plans for both indoor and outdoor scenes (Right).

are very similar. Therefore, using over-refined semantic labels for the evaluation is too strict for the plan synthesis application. In fact, for the labeling task with three label groups, we achieve average accuracy 76%, precision 80%, and recall 80%. Consequently, in our plan example dataset, the network-generated semantic labels associated with each example plan are meaningful. On average, it takes 0.8s, 0.07s, 1.5s, and 0.06s for performing WFC, segmentation network, plan refinement, and labeling network respectively. The training times for segmentation and labeling network are 8h and 1h, respectively. Besides, the plan retrieval takes about 9.6s. The experiments were run on a PC with Intel Core i9-10900K CPU, 64GB RAM, and a TITAN RTX GPU.

In Figure 11, we illustrate our dataset based on its 2D shape features, i.e., area-perimeter vectors. The area of a floor plan is calculated by counting the number of pixels within the plan. The perimeter is determined by counting the number of pixels along the outer boundary of the entire floor plan. Note that both the left and right of Figure 11 show the feature space of the same dataset, in which all floor plans are normalized to 32×32 pixels. Each point represents a floor plan in the dataset created by our PlanNet. We highlight the retrieval results of two different input boundaries on the left and right, respectively. We can see that the plans of the near points have similar boundaries, while the far points always have different shapes. Since the shape feature is mainly used to encode the shape of the floor plan instead of their inner space layouts, we can also see that even the plans with similar boundaries could have different inner space layouts. This ensures the inner space diversity of the synthesized floor plans given a boundary input. However, the simple area-perimeter shape features can hardly measure the exact shape of a floor plan, e.g. point distribution of the Top-1000 plans is not gathered in the left case, we still need to perform fine retrieval in the 1000 floor plan candidates. On the other hand, lots of RPLAN data have small balconies, bathrooms, and storerooms on the boundary. The created plan examples in our dataset thus also have small rooms on the boundary or corner due to the segmented network trained by RPLAN.

In Table 1, we compare the RPLAN dataset and ours by

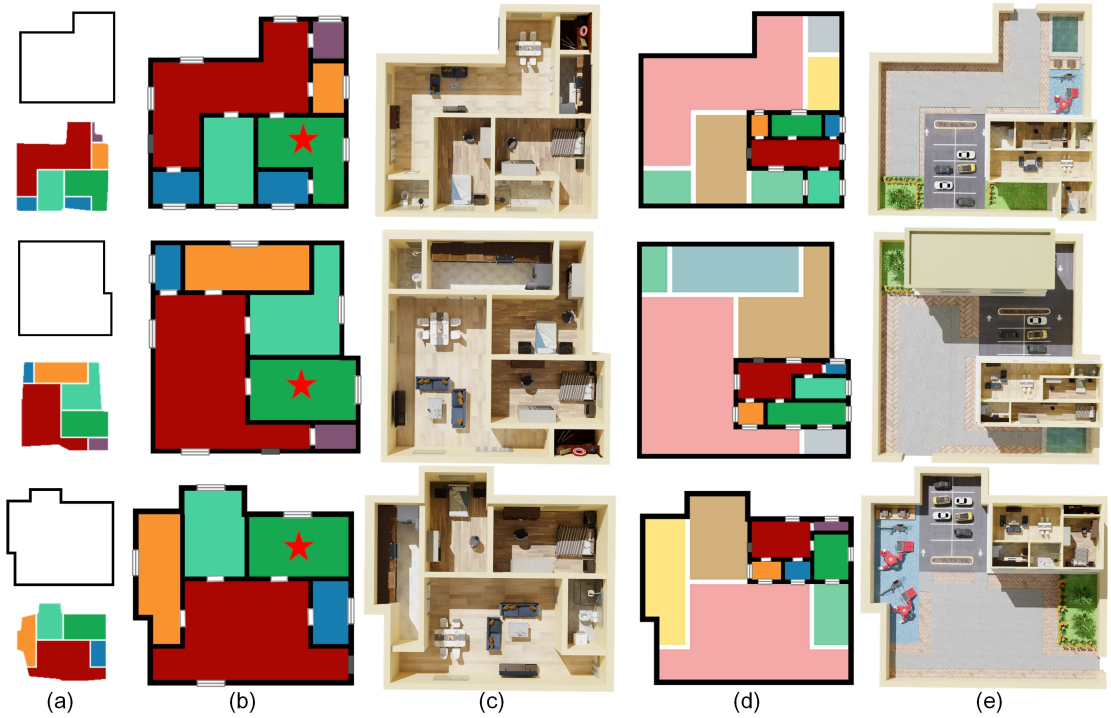


Fig. 17. Three groups of plan-driven 3D scene synthesis results. In each row, we show the input boundary (a)-Top, the retrieved plan example (a)-Bottom, the generated floor plan (b), and the associated 3D scene (c). We also show further subdivision results in (d) about the components highlighted with red stars in (b). These plans are used to design 3D scenes on larger scales (e).

using the layout statistics, including the number of rooms (R), the number of rooms connected to the living room (C), and the area ratio of the living room (A). It shows that our generation model can automatically create floor plans with similar layout statistics as data produced by human designers. To test the stability of the distributions of floor plans generated by our method, we produce three additional 10K datasets. We then use area-perimeter vectors to compare their feature space with that of the RPLAN dataset (Figure 12-Top). To make our plans comparable with RPLAN, we constrain their boundaries to be either horizontal or vertical. The distributions show that our generative models can produce plans with greater configuration variation. The similarity of the distributions of the three new datasets also demonstrates the stability and consistency of our method. In Figure 12-Bottom, we display three groups of plans closest to three specified positions (red points in Figure 12-Top). It can be seen that plans with similar features can still vary in shape and inner layout. This underscores the importance of using IOU-based retrieval and increasing the number of datasets to achieve greater diversity in plan synthesis of inner layouts. On the other hand, our method might generate some uncommon floor plan shapes (e.g., those closest to point #1), and some rooms may have low-quality shapes. In our retrieval-based plan synthesis method, plan examples would only be explored when given unnatural boundary inputs. To enhance the floor plans' quality, users can manually modify the walls in this scenario.

6.2 Plan Retrieval and Synthesis

We first test our dataset plan examples through plan retrieval to demonstrate their ability to support plan synthesis. In Figure 13, we validate the feasibility of our plan dataset through three groups of retrieval results. Given a polygon boundary (Left), proper plan examples from the dataset can be retrieved in terms of configuration similarities (Middle). These plan examples are ranked from left to right illustrating the similarity in descending order. In Figure 13-Right, we show the retrieval results subject to both the user input boundaries and the user-specified component amounts.

In Figure 14, we first show the plan synthesis results with hard boundary constraints (columns a & b). Column (a) shows the synthesized plans with polygon inputs, while the middle column shows the results with input curved boundaries. Like the polygon inputs, the curved boundary inputs have binary maps for plan retrieval and pixels for plan reshaping. Hence, our plan synthesis method for polygon inputs can also be used for curved boundary inputs. Then, we show the plan synthesis results with soft boundary constraints in column (c). We use the semantic labels of outdoor scenes to show the design of living quarters with two plots of lands. Note that the retrieved plan examples are clipped by the input boundaries. In Figure 15-Left, we choose three low-quality plan examples from the retrieval results in Figure 13. For the three results on the right, the first one use the same boundary in Figure 13-Left. We use two boundaries similar to the first one to create the other two results. These results demonstrate that, by using the inputs as hard boundary constraints, even the low-quality plan examples can be reshaped to be plausible floor plans. Note that the design can be further improved by following the modification suggestions made by our system (e.g., in the bottom of Figure 15).

In addition, We provide several examples of 3D indoor

TABLE 2
Layout statistics comparisons between our method and Wu et al. 2019 [5] (a), Hu et al. 2020 [6] (b), Sun et al. 2022 [11] (c), respectively. Under each metric, the left value of each vertical bar belongs to the compared method while the right one is ours. Note that for the last three metrics, our results consist of the metric value of the floor plan before applying the suggested modifications, plus the increase in value after the modifications.

Figure 18	Avg. C	Avg. A	V	AVG. Min. R_a	AVG. Min. R_c	AVG. Min. R_e
(a)	$4.67 \mid 4.33$	$0.47 \mid 0.47$	$2.20 \mid 1.45$	$0.76 \mid 0.73 + 0.04$	$0.15 \mid 0.15 + 0.01$	$0.29 \mid 0.32 + 0.01$
(b)	$4.50 \mid 5.33$	$0.44 \mid 0.40$	$2.00 \mid 1.33$	$0.76 \mid 0.68 + 0.03$	$0.15 \mid 0.15 + 0.01$	$0.28 \mid 0.31 + 0.02$
(c)	$5.83 \mid 5.00$	$0.44 \mid 0.38$	$1.57 \mid 1.08$	$0.65 \mid 0.67 + 0.06$	$0.13 \mid 0.13 + 0.02$	$0.26 \mid 0.28 + 0.04$

and outdoor scene synthesis guided by our designed plans. In Figure 16, we show the input boundaries and the retrieved dataset plan examples on the left, along with the 3D scenes (two for indoor and two for outdoor) constructed by the synthesized plans on the right. Note that the random selection algorithm adjusts the component semantic labels thus impacting the constructed scenes. In Figure 17, we illustrate how our subdivision workflow works in progressive 3D scene construction. The users can set up the plan scale and further subdivide certain component(s) to obtain more complex designs on different scales. These plan synthesis results and constructed 3D scenes demonstrate the feasibility of our generative model in component-based plan synthesis. Note the 3D scenes in both Figure 16 and 17 are created manually, but their floor plans that guided the manual scene modeling are generated by our system. These results show how our system facilitates 3D scene modeling at multiple scales.

6.3 Comparisons

We compare our results with the state-of-the-art floor plan synthesis methods [5], [6], [11], [12], [13]. We directly collect high-quality floor plan results from these papers and use the same boundaries to create plans using our system. The dataset plan examples used for our results are manually selected from the Top-10 retrieval results explored by our system.

In Figure 18, we compare our method with Wu et al. 2019 [5] (a), Hu et al. 2020 [6] (b), and Sun et al. 2022 [11] (c), respectively. For a fair comparison, we specify the same number of rooms for our system as the compared methods. The number of rooms (R) thus cannot be used as a metric in this comparison. Therefore, we only compare the number of rooms connected to the living room (C), the area ratio of the living room (A), and the variance of A (denoted as V), between our results and those by the compared methods. Besides, our results are slightly adjusted following the modification suggestion. We also compare the average minimum room metrics for these floor plans. For values of AVG. Min. R_a, R_c, and R_e, a larger value indicates a better floor plan. The quantitative comparison results are summarized in Table 2. From the first two metrics, we can see that the results of the compared methods all have plausible inner layouts based on the ground truth (RPLAN) in Table 1. The variance V shows that the area ratio of the living room of our results is more consistent than that of the other methods. The last three room metrics show that the modification schemes suggested by our system can enhance the quality of the synthesized floor plans. Additionally, our results outperform those of other methods in terms of these

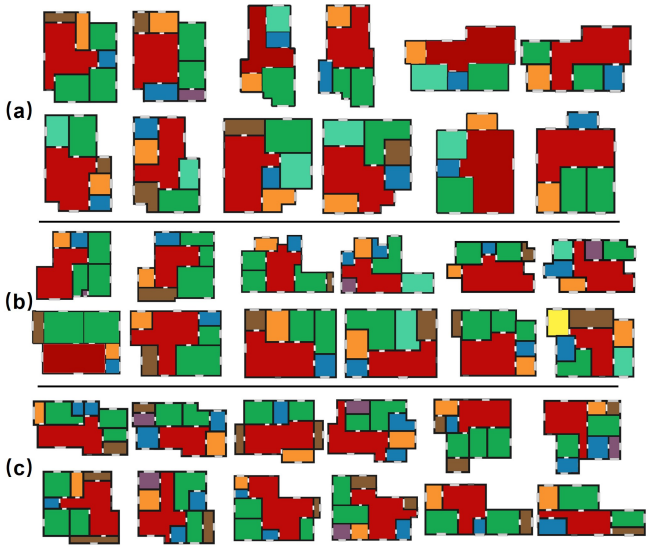


Fig. 18. Visual comparisons with Wu et al. 2019 [5] (a), Hu et al. 2020 [6] (b), and Sun et al. 2022 [11] (c). In each plan pair, we show the result of the compared method (Left) and ours (Right).

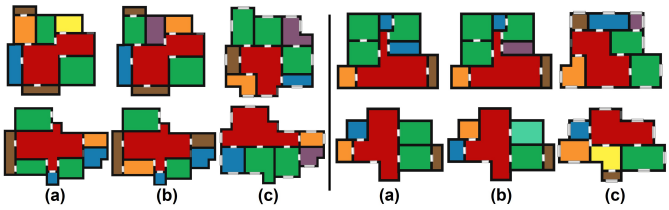


Fig. 19. Comparisons with Nauata et al. 2021 [13] (Left) and Para et al. 2021 [12] (Right). On each side, we show the plan generated by the compared method (a), our semantic prediction result for the same plan (b), and our result, given the same out boundary (c).

room metrics, indicating that our method generates floor plans that are well-suited for indoor scenes.

In addition, we also conducted a user study to quantitatively compare these results. 30 postgraduate students who majored in art & design were invited to this study. All pairs of floor plans in Figure 18 were used in the user study via a forced choice preference approach, i.e., we asked the participants to choose one plan from each presented plan pair. Note that the floor plans in each pair are in a random order. The user study results show that our results got 98, 101, and 90 votes from a total of 180 votes, when compared with [5], [6], and [11] respectively.

We also qualitatively compare our method with the layout generation methods (Nauata et al. 2021 [13] and Para et al. 2021 [12]). These methods leverage the relation

graphs of indoor scenes to generate floor plans and make variations. In Figure 19, we select the high-quality floor plans generated by the compared methods (column (a) on both sides). Considering that their results are guided by scene graphs, we first remove the plan labels and then use our labeling network to predict the component semantic labels (column (b) on both sides). The results show that, like [12], [13], our method also can establish correct relations between the semantic labels and plan components. The column (c) on both sides is generated by our system, given the same outer boundaries as column (a). Our system gives different but equally reasonable results, demonstrating the multi-solvability of indoor scene space division.

These comparisons demonstrate that our work, even relying on an example-driven method for plan synthesis, can still achieve competitive results compared with state-of-the-art methods. This also validates the usability of the proposed *PlanNet* in plan creation with adequate variations. Moreover, our method can be extended to outdoor scenes much easier than the other two methods, and is available for soft boundary inputs to assist the prototype design of architecture.

7 Conclusion

In this paper, we have introduced a novel generative model *PlanNet* for component-based plan synthesis. The proposed model consists of a WFC-based module for embryonic plan generation, and two deep neural networks to outline the plausible boundary from each embryonic plan and estimate the semantic labels of the components on each created plan. We use PlanNet to construct a dataset with 10K plan examples. The plan example dataset can facilitate the component-based plan synthesis given a polygon constraint. Our user interface provides a recursive subdivision workflow to enrich the designed plans on different scales and a random selection algorithm to generate plausible per-component semantic labels for the floor plan. We show several experimental results and comparisons to demonstrate the feasibility and efficacy of our method.

Limitations. Our current method still has several limitations. First, our method cannot directly generate plans with regular layouts such as plans with aligned rooms of equal size, or symmetrical layouts. For these demands, extra rules are required to splice the generated plans of our method into a larger one with a regular or symmetrical layout. Second, although we try to establish a correspondence between the pre-component semantic labels of indoor and outdoor scenes to extend our method to outdoor scenes, our method is still not very suitable for designing outdoor scenes. This is because we train our segmentation network using only an indoor scene dataset (RPLAN), so the components of the floor plan have more characteristics of indoor scenes. Additionally, simple correspondences are not always accurate. For example, we label the living room as corresponding to the road label because both of them connect other components. However, the shape of a living room is not similar to that of a road in outdoor scenes, which can result in generated outdoor scenes with too wide and spacious roads. Third, due to the discretized tiles adopted in the WFC process, the designed floor plans only have straight inner walls and right-angle corners, and the relevant distances of their inner walls would be limited to some constant values. This would limit the style variation of the floor plans generated by our method. Lastly, the current plan retrieval employs a time-consuming pair-wise comparison, making our system not efficient enough. We find that decreasing the number of candidates explored from the feature-based ranking could speed up the plan retrieval. For example, if we use Top-100 candidates for pair-wise comparison, the time cost of plan retrieval decreases from 9.6s to 1.0s. However, this would also decrease the retrieval accuracy.

In the future, we plan to extend our method to 3D space division, which means plans on different floors in a house can be jointly synthesized, and the layout of decoration on the wall can also be considered. Besides, we are also interested in exploring common rules from the domain knowledge between urban design and architectural design to enhance our generative model and improve the quality of synthesized plans. We believe that the automatic plan synthesis method would inspire more applications in various fields.

Acknowledgements

This work was supported by a grant from the NSFC (No.61902032).

References

[1] C.-H. Peng, Y.-L. Yang, F. Bao, D. Fink, D.-M. Yan, P. Wonka, and N. J. Mitra, "Computational network design from functional specifications," *ACM Transactions on Graphics*, vol. 35, no. 4, 2016.

[2] F. Bao, D.-M. Yan, N. J. Mitra, and P. Wonka, "Generating and exploring good building layouts," *ACM Transactions on Graphics*, vol. 32, no. 4, 2013.

[3] P. Merrell, E. Schkufza, Z. Li, M. Agrawala, and V. Koltun, "Interactive furniture layout using interior design guidelines," *ACM Transactions on Graphics*, vol. 30, no. 4, pp. 87:1–87:10, 2011.

[4] S. Zhang, Y. Li, Y. He, Y. Yang, and S. Zhang, "Mageadd: Real-time interaction simulation for scene synthesis," in *ACM International Conference on Multimedia*, 2021, pp. 965–973.

[5] W. Wu, X.-M. Fu, R. Tang, Y. Wang, Y.-H. Qi, and L. Liu, "Data-driven interior plan generation for residential buildings," *ACM Transactions on Graphics*, vol. 38, no. 6, 2019.

[6] R. Hu, Z. Huang, Y. Tang, O. Van Kaick, H. Zhang, and H. Huang, "Graph2plan: Learning floorplan generation from layout graphs," *ACM Transactions on Graphics*, vol. 39, no. 4, Jul. 2020.

[7] A. Schulz, A. Shamir, D. I. W. Levin, P. Sitthi-amorn, and W. Matusik, "Design and fabrication by example," *ACM Transactions on Graphics*, vol. 33, no. 4, 2014.

[8] M. Fisher, D. Ritchie, M. Savva, T. Funkhouser, and P. Hanrahan, "Example-based synthesis of 3d object arrangements," *ACM Transactions on Graphics*, vol. 31, no. 6, pp. 135:1–135:11, 2012.

[9] S.-H. Zhang, S.-K. Zhang, W.-Y. Xie, C.-Y. Luo, Y. Yang, and H. Fu, "Fast 3d indoor scene synthesis by learning spatial relation priors of objects," *IEEE Transactions on Visualization and Computer Graphics*, vol. Early Access, pp. 1–1, 2021.

[10] M. Gumin, "Wavefunction collapse," https://github.com/mxgmn/WaveFunctionCollapse, 2016.

[11] J. Sun, W. Wu, L. Liu, W. Min, G. Zhang, and L. Zheng, "Wallplan: Synthesizing floorplans by learning to generate wall graphs," *ACM Transactions on Graphics*, vol. 41, no. 4, jul 2022.

[12] W. Para, P. Guerrero, T. Kelly, L. Guibas, and P. Wonka, "Generative layout modeling using constraint graphs," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 6670–6680.

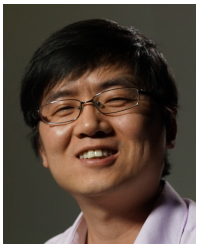
[13] N. Nauata, S. Hosseini, K.-H. Chang, H. Chu, C.-Y. Cheng, and Y. Furukawa, "House-gan++: Generative adversarial layout refinement network towards intelligent computational agent for professional architects," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 13 627–13 636.

[14] P. Xu, H. Fu, T. Igarashi, and C.-L. Tai, "Global beautification of layouts with interactive ambiguity resolution," in *UIST '14*, 2014.
[15] P. Xu, H. Fu, C.-L. Tai, and T. Igarashi, "GACA: Group-aware command-based arrangement of graphic elements," in *Proceedings of CHI '15*, 2015.
[16] C. Zou, J. Cao, W. Ranaweera, I. Alhashim, P. Tan, A. Sheffer, and H. Zhang, "Legible compact calligrams," *ACM Transactions on Graphics*, vol. 35, no. 4, p. Article 122, 2016.
[17] J. Li, J. Yang, J. Zhang, C. Liu, and T. Xu, "Attribute-conditioned layout gan for automatic graphic design," *IEEE Transactions on Visualization and Computer Graphics*, vol. PP, no. 99, pp. 1–1, 2020.
[18] Q. Fu, F. Zhang, X. Li, and H. Fu, "Magic furniture: Design paradigm of multi-function assembly," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–12, 2023.
[19] C. A. Vanegas, I. Garcia-Dorado, D. G. Aliaga, B. Benes, and P. Waddell, "Inverse design of urban procedural models," *ACM Transactions on Graphics*, vol. 31, no. 6, 2012.
[20] Y.-L. Yang, J. Wang, E. Vouga, and P. Wonka, "Urban pattern: Layout design by hierarchical domain splitting," *ACM Transactions on Graphics*, vol. 32, no. 6, 2013.
[21] J. Lin, D. Cohen-Or, H. Zhang, C. Liang, A. Sharf, O. Deussen, and B. Chen, "Structure-preserving retargeting of irregular 3d architecture," *ACM Transactions on Graphics*, vol. 30, p. 183, 12 2011.
[22] M. Hendrikx, S. Meijer, J. Velden, and A. Iosup, "Procedural content generation for games: A survey," *ACM Transactions on Multimedia Computing, Communications, and Applications*, 2013.
[23] L.-F. Yu, S. K. Yeung, C.-K. Tang, D. Terzopoulos, T. F. Chan, and S. Osher, "Make it home: automatic optimization of furniture arrangement," *ACM Transactions on Graphics*, vol. 30, no. 4, p. 86, 2011.
[24] M. Fisher, M. Savva, and P. Hanrahan, "Characterizing structural relationships in scenes using graph kernels," *ACM Transactions on Graphics*, vol. 30, no. 4, pp. 34:1–34:12, 2011.
[25] W. Wu, L. Fan, L. Liu, and P. Wonka, "Miqp-based layout design for building interiors," *Computer Graphics Forum*, vol. 37, no. 2, pp. 511–521, 2018.
[26] K. Xu, R. Ma, H. Zhang, C. Zhu, A. Shamir, D. Cohen-Or, and H. Huang, "Organizing heterogeneous scene collections through contextual focal points," *ACM Transactions on Graphics*, vol. 33, no. 4, pp. 35:1–35:12, 2014.
[27] G. Xiong, Q. Fu, H. Fu, B. Zhou, and Z. Deng, "Motion planning for convertible indoor scene layout design," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 12, pp. 4413–4424, 2021.
[28] N. Nauata, K.-H. Chang, C.-Y. Cheng, G. Mori, and Y. Furukawa, "House-gan: Relational generative adversarial networks for graph-constrained house layout generation," in *2020 European Conference on Computer Vision (ECCV)*, 2020, pp. 162–177.
[29] K. Wang, M. Savva, A. X. Chang, and D. Ritchie, "Deep convolutional priors for indoor scene synthesis," *ACM Transactions on Graphics*, vol. 37, no. 4, p. 70, 2018.
[30] M. Li, A. Gadi Patil, K. Xu, S. Chaudhuri, O. Khan, A. Shamir, C. Tu, B. Chen, D. Cohen-Or, and H. Zhang, "Grains: Generative recursive autoencoders for indoor scenes," *ACM Transactions on Graphics*, vol. 38, no. 2, 2019.
[31] K. Wang, Y.-A. Lin, B. Weissmann, M. Savva, A. X. Chang, and D. Ritchie, "Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks," *ACM Transactions on Graphics*, vol. 38, no. 4, Jul. 2019.
[32] Q. Fu, H. Fu, Z. Deng, and X. Li, "Indoor layout programming via virtual navigation detectors," *Science China Information Sciences*, vol. 65, pp. 189 101:1–189 101:2, 2022.
[33] Thomas, Funkhouser, Michael, Kazhdan, Philip, Shilane, Patrick, Min, William, and K. and, "Modeling by example," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 652–663, 2004.
[34] X. Su, X. Chen, Q. Fu, and H. Fu, "Cross-class 3d object synthesis guided by reference examples," *Computers & Graphics*, vol. 54, no. C, p. 145–153, 2016.
[35] C. H. Shen, H. Fu, K. Chen, and S. M. Hu, "Structure recovery by part assembly," *ACM Transactions on Graphics*, vol. 31, no. 6, 2012.
[36] X. Chen, J. Li, Q. Li, B. Gao, D. Zou, and Q. Zhao, "Image2scene: Transforming style of 3d room," in *Proceedings of the 23rd Annual ACM Conference on Multimedia Conference*. ACM, 2015, pp. 321–330.
[37] Q. Fu, X. Chen, X. Wang, S. Wen, B. Zhou, and H. Fu, "Adaptive synthesis of indoor scenes via activity-associated object relation graphs," *ACM Transactions on Graphics*, vol. 36, no. 6, p. 201, 2017.
[38] D. Aliaga, C. Vanegas, and B. Benes, "Interactive example-based urban layout synthesis," *ACM Transactions on Graphics*, vol. 27, p. 160, 12 2008.
[39] C.-H. Peng, Y.-L. Yang, and P. Wonka, "Computing layouts with deformable templates," *ACM Transactions on Graphics*, vol. 33, no. 4, 2014.
[40] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015, pp. 234–241.
[41] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2017.
[42] Y. Ma, D. Yu, T. Wu, and H. Wang, "Paddlepaddle: An open-source deep learning platform from industrial practice," *Frontiers of Data and Computing*, vol. 1, no. 1, pp. 105–115, 2019.
[43] E. Douglas, "Algorithms for the reduction of the number of points required to represent a digitised line or its caricature," *Canadian Cartographer*, vol. 10, 1973.

Qiang Fu is an Associate Professor in the School of Digital Media and Design Arts, Beijing University of Posts and Telecommunications. He received a Ph.D. in computer science from Beihang University, China, in 2018 and a B.E. in automation from Beihang University in 2011. His research interests include computer graphics and virtual reality.

Shuhan He received a B.E. in digital media technology from Beijing Forestry University. She is a postgraduate student in the School of Digital Media and Design Arts, Beijing University of Posts and Telecommunications. Her research interests include computer graphics and machine learning.

Xueming Li received a B.E. in electronics engineering from the University of Science and Technology of China, in 1992, and a Ph.D. in electronics engineering from Beijing University of Posts and Telecommunications in 1997. He is a professor in the School of Digital Media and Design Arts, Beijing University of Posts and Telecommunications. His research interests include digital image processing, video coding, and multimedia telecommunication.



Hongbo Fu is a Professor in the School of Creative Media, City University of Hong Kong. He received a Ph.D. in computer science from Hong Kong University of Science and Technology in 2007 and a B.S. in information sciences from Peking University, China, in 2002. His primary research interests fall in the fields of computer graphics and human computer interaction. He has served as an associate editor of The Visual Computer, Computers & Graphics, and Computer Graphics Forum.