Data-driven Interior Plan Generation for Residential Buildings

WENMING WU, University of Science and Technology of China, China
XIAO-MING FU*, University of Science and Technology of China, China
RUI TANG, Kujiale, China
YUHAN WANG, Kujiale, China
YU-HAO QI, University of Science and Technology of China, China
LIGANG LIU*, University of Science and Technology of China, China

We propose a novel data-driven technique for automatically and efficiently generating floor plans for residential buildings with given boundaries. Central to this method is a two-stage approach that imitates the human design process by locating rooms first and then walls while adapting to the input building boundary. Based on observations of the presence of the living room in almost all floor plans, our designed learning network begins with positioning a living room and continues by iteratively generating other rooms. Then, walls are first determined by an encoder-decoder network, and then they are refined to vector representations using dedicated rules. To effectively train our networks, we construct *RPLAN* - a manually collected large-scale densely annotated dataset of floor plans from real residential buildings. Intensive experiments, including formative user studies and comparisons, are conducted to illustrate the feasibility and efficacy of our proposed approach. By comparing the plausibility of different floor plans, we have observed that our method substantially outperforms existing methods, and in many cases our floor plans are comparable to human-created ones.

CCS Concepts: • **Computing methodologies → Computer graphics**; **Neural networks**; **Probabilistic reasoning**.

Additional Key Words and Phrases: floor plan generation, interior layout, data-driven approach, neural network, deep learning

ACM Reference Format:
Wenming Wu, Xiao-Ming Fu, Rui Tang, Yuhan Wang, Yu-Hao Qi, and Ligang Liu. 2019. Data-driven Interior Plan Generation for Residential Buildings. *ACM Trans. Graph.* 38, 6, Article 234 (November 2019), 12 pages. https://doi.org/10.1145/3355089.3356556

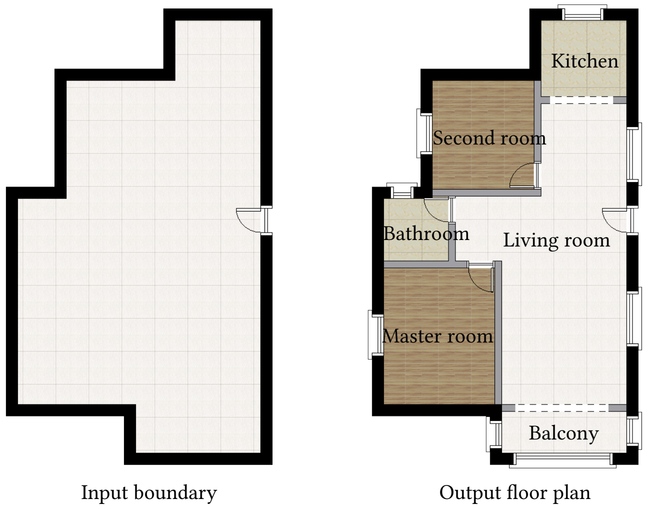


Fig. 1. Floor plan for one residential building generated by our approach, given a boundary as input.

1 INTRODUCTION

Designing a floor plan is an essential part of building a dwelling, as this plan indicates room connections, room types, room sizes, and wall lengths. Generally, floor plan design is an iterative trial-and-error and time-consuming process between interior designers, which requires significant expertise and experience, and home users.

In this paper, we consider a different problem of automatically designing floor plans for residential buildings, given the boundary as input only. In computer games and networked virtual environments, efficiently and automatically generating floor plans from a given boundary is practical and demanded. In interior design, providing an initial design or candidate option for house renovation with the existing boundary is very useful.

Some techniques have been proposed for automatically generating floor plans [Hua 2016; Liu et al. 2013; Merrell et al. 2010; Wu et al. 2018]. Most of them have considered floor plan generation as constraint-based systems with high-level constraints, such as room sizes, positions, and adjacencies. However, these constraints are highly dependent on the knowledge given by the individual designers and they may even be inconsistent in some cases. The work of [Merrell et al. 2010] cannot maintain the input boundary.

In order to avoid enumerating all high-level constraints and improve the plausibility of the generated floor plans, we aim to develop a learning based method for this task. That is, given a building boundary, its floor plan is automatically learned from a dataset of real floor plans without specifying any constraints. However, the challenges of achieving the goal are two fold. First, a large-scale dataset consisting of real floor plans with complete room annotations is expected.

*The corresponding authors

Authors' addresses: Wenming Wu, University of Science and Technology of China, China, wwming@mail.ustc.edu.cn; Xiao-Ming Fu, University of Science and Technology of China, China, fuxm@ustc.edu.cn; Rui Tang, Kujiale, China, ati@qunhemail.com; Yuhan Wang, Kujiale, China, daishu@qunhemail.com; Yu-Hao Qi, University of Science and Technology of China, China, qiyuhao7@mail.ustc.edu.cn; Ligang Liu, University of Science and Technology of China, China, lgliu@ustc.edu.cn.

There is no such dataset yet. Second, it is non-trivial to design a highly individualized learning approach that can quickly generate intuitive floor plans comparable to those designed by humans, given only the boundary as input.

To this end, we propose a novel data-driven method to automatically and efficiently generate floor plans for residential buildings with the given boundary. First, we have built *RPLAN*, a large-scale dataset containing more than 80K real floor plans from residential buildings. Each floor plan is represented as vector graphics composed of labeled rooms and walls. Then we develop a two-stage approach to learn the floor plan based on the observation that professionals design floor plans with two phases [Rengel 2011]: (i) determining room connections and positions and (ii) computing room sizes and wall positions. To predict the room locations, we use the iterative learning method of [Ritchie et al. 2019; Wang et al. 2018] with one novel modification - a living room first strategy to improve the plausibility of our generated floor plans. Then, given the computed locations of the rooms, we train an encoder-decoder network to predict the wall positions, and use customized rules to transform a pixelated representation into a vector representation.

From numerous experiments, our method is able to learn design rules from the dataset and has achieved better plausibility than existing methods. User studies have shown that it has achieved comparable floor plan results to those created by humans. Moreover, our approach achieves a satisfactory level of efficiency of generating one floor plan within an average of four seconds.

Our main contributions are as follows:

- We propose a novel learning method to automatically and efficiently generate floor plans for residential buildings given the building boundary as input only.
- To effectively train our networks, *RPLAN*, a large-scale dataset containing more than 80K real floor plans from residential buildings, is constructed with dense annotations.

To the best of our knowledge, this is the first time floor plan design has been automated using only the boundary as constraint without any pre-defined high-level constraints on rooms and walls. The constructed large-scale dataset *RPLAN* has pretty much potential to inspire more research.

2 RELATED WORK

Layout synthesis in computer graphics. Synthesizing layouts is an essential topic in computer graphics and plays an important role in many applications, such as architecture, computer games, and vitual/augmented reality [Liggett 2000; Smelik et al. 2014]. There are many types of layouts, such as urban layouts [Aliaga et al. 2008; Chen et al. 2008; Peng et al. 2016, 2014; Yang et al. 2013], game level layouts [Hendrikx et al. 2013; Ma et al. 2014], architectural layouts [Bao et al. 2013; Harada et al. 1995; Müller et al. 2006], interior layouts [Feng et al. 2016; Merrell et al. 2010; Wu et al. 2018], indoor scenes [Fisher et al. 2012; Merrell et al. 2011; Wang et al. 2018; Yu et al. 2011], and page layouts [Harada et al. 1995; Li et al. 2019; O'Donovan et al. 2014]. In this paper, we focus on floor plan generation for residential buildings.

Floor plan generation. Floor plan generation may be defined as the process of determining the position and size of several rooms. Due to the combinatorial complexity, evolutionary algorithms and data-driven methods have been developed. For rectangular single-story dwellings, Michalek *et al.* [2002] propose an optimization model that is solved by combining gradient-based algorithms with evolutionary algorithms. An enhanced hybrid evolutionary algorithm is developed to generate a set of floor plans in the early design stages of architectural practice [Rodrigues et al. 2013a,b,c]. Bahrehmand *et al.* [2017] use an evolutionary approach to design an interactive layout solver. Given a small dataset with 120 architectural programs [Merrell et al. 2010], they use a Bayesian network to learn attributes of rooms and synthesize the layout using the stochastic method without fixing the boundary. A data-driven method is proposed to estimate room dimensions and orientations [Rosser et al. 2017]. Some other methods have been proposed, such as a physically based space modeling method [Arvin and House 2002] and a mixed integer quadratic programming (MIQP) based method [Wu et al. 2018]. Wu *et al.* [2018] adopt high-level constraints as inputs and generate building interiors based on a MIQP formulation. An interactive system is presented to generate floor plans subject to design, user, and manufacturing constraints [Liu et al. 2013]. A lazy generation method is pretested to generate grid-like interior layouts [Hahn et al. 2006]. Floor plans with irregular rooms are automatically created [Hua 2016]. These methods do not generate floor plans from scratch using only the given boundary as input.

Deep layout generation. Deep learning architecture holds some promise in addressing the layout generation problem. Deep convolutional neural networks [Ritchie et al. 2019; Wang et al. 2018] are used for indoor scene synthesis. LayoutGAN [Li et al. 2019] is a novel Generative Adversarial Network that synthesizes layouts for graphic design. A room layout is estimated from a single RGB panorama using a deep learning framework [Yang et al. 2019; Zou et al. 2018]. A floor plan is reconstructed from a rasterized floor plan image [Liu et al. 2017]. RGBD streams are used to automatically reconstruct a floor plan using a novel neural architecture, called FloorNet [Liu et al. 2018]. We also use the deep learning approach to generate floor plans for residential buildings, but only with the boundary as input. Aydemir *et al.* [2012] provide a dataset which contains 940 floors. There are only 870 floor plan images in [Liu et al. 2017] and 5,000 samples in [Kalervo et al. 2019]. Large-scale floor plans can be extracted from SUNCG [Song et al. 2017] consisting of large-scale synthetic indoor scenes; however, they are synthetic and cannot model the complexity of real floor plans or replace real floor plans. To this end, we propose a large-scale dataset with more than 80K real floor plans from residential buildings.

3 OVERVIEW

Problem. Our method receives as input a building's outer boundary defined as the geometry of the exterior walls with an entrance (Fig. 2 (a)). Our goal is to generate a desired floor plan, as a layout of rooms and walls with room annotations (Fig. 2 (c) and (d)).

Challenges. However, there are two challenges. First, since only the boundary is given, it is non-trivial to design a highly individualized learning approach to automatically and efficiently generate intuitive floor plans that are comparable to the human-created floor

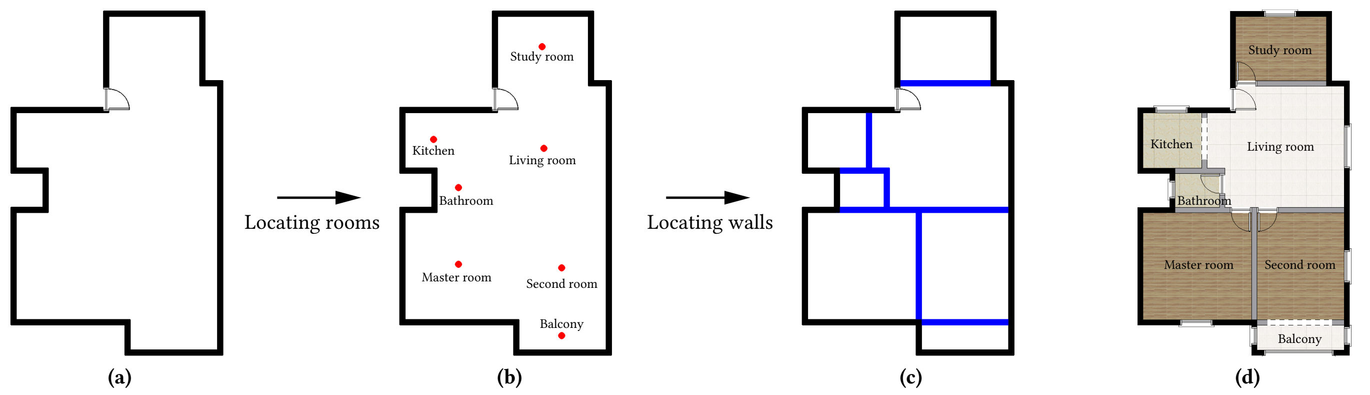


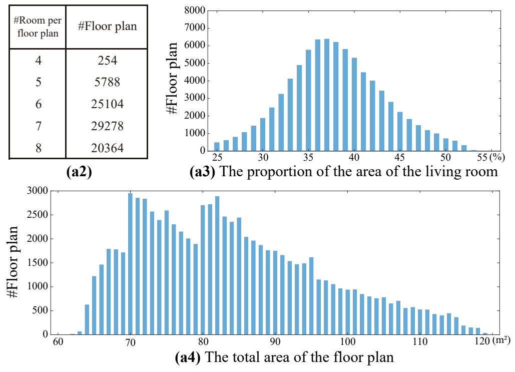
Fig. 2. Overview of our method. Given an input boundary (a), our method first uses an iterative prediction model to obtain room locations (red dots in b). Based on the this, our method further locates walls (shown in blue) to obtain a vectorized floor plan (c). In this step, our method uses an encoder-decoder network to predict wall locations and then a post-processing step converts the floor plan into the final vector format. Additional details, including doors and windows, are added to visualize the floor plan (d).

Room type	#Floor plan
Bathroom	97113
SecondRoom	99987
Balcony	86545
LivingRoom	80788
MasterRoom	80466
Kitchen	77768
StudyRoom	14985
Storage	3351
Wall-in	1043
GuestRoom	860
DiningRoom	1312
ChildRoom	3928
Entrance	292

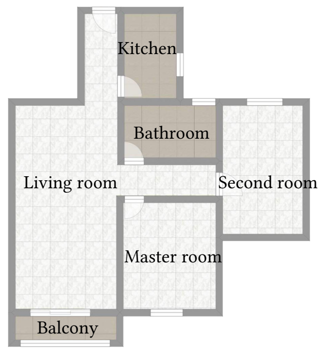
(a1)

#Room per floor plan	#Floor plan
4	254
5	5788
6	25104
7	29278
8	20364

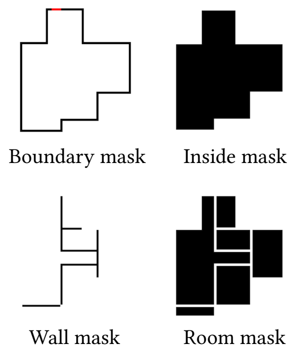
(a2)



(a) Statistics of our dataset



(b) Floor plan



(c) Ground truth

Fig. 3. Our dataset *RPLAN*. (a) Statistics on the occurrence of each room type (a1), room number per floor plan (a2), the proportion of the area of the living room (a3), and the number of floor plan according to the total area (a4). (b) One typical floor plan in our collected dataset. (c) For each floor plan, we abstract the necessary information used in our method, including the boundary mask (the entrance is shown in red), inside mask, (interior) wall mask, and room mask.

plans. The second challenge is that the data-driven method requires a large amount of training data. However, there is no such a large-scale database of floor plans from real residential buildings.

Methodology. To solve the data problem, we propose RPLAN: a large-scale dataset with more than 80K floor plans from real residential buildings with labeled rooms and walls (Section 4). Inspired by human artists' creative processes, we propose a two-stage approach to first locate rooms and then walls (Section 5). Fig. 2 shows the workflow of our proposed two-stage method. Given the building boundary (Fig. 2 (a)), the first stage is to predict room types and locations with an iterative learning scheme, shown as the red dots in Fig. 2 (b). The second is to locate walls using an encoder-decoder network and some dedicated rules (Fig. 2 (c) and (d)).

4 DATASET OF RPLAN

We have built *RPLAN*, a large-scale dataset of floor plans from residential buildings with semantic annotations at the pixel level (Fig. 3). To inspire more research, the dataset will be published later.

Data collection. We have collected more than 120K floor plans from real-world residential buildings in the estate market in Asia. The dataset is collected at our own expense with the user and floor plan privacy eliminated. Hence, all floor plans in the dataset have no copyright issue. Each floor plan has a vector-graphics representation within a squared region of $18m \times 18m$, including the geometric and semantic information as shown in Fig. 3 (b). For the sake of applying learning scheme, we convert each floor plan into a 256×256 image.

Filtering. Real-world residential buildings often have some small areas for the flue, elevator and equipment platform. These are not our target since these areas are too small and may be randomly set. To avoid the interference of these factors and enhance the reliability of the dataset, we filter out some non-standard data for training and testing. Therefore, we first remove floor plans that contain undefined room types or rooms with very low frequency. In our dataset, we have 13 kinds of rooms after filtering. Then, we only keep floor plans that satisfy all the following requirements:

(1) The total area of the floor plan is larger than 60 square meters and less than 120 square meters.
(2) The number of rooms in the floor plan is larger than 3 and less than 9.
(3) The floor plan has a living room.

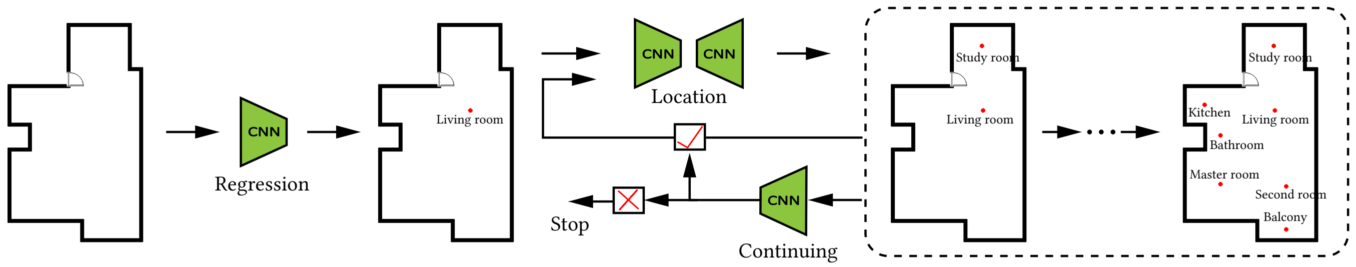


Fig. 4. An iterative prediction model to predict room locations. Our prediction model consists of three deep networks. Given the boundary as input, our model first adopts a regression network to choose a location for the living room. Based on this, a location network and a continuing network are iteratively used to predict the types and locations of other rooms in a step-by-step manner. The iterative process stops when the continuing network decides not to add rooms.

(4) The proportion of the area of the living room to the total area of the floor plan is larger than 0.25 and less than 0.55.
(5) The average area of each room is larger than 10 square meters and less than 20 square meters.

After filtering, we are left with more than 80K floor plans in our dataset, whose statistics are shown in Fig. 3 (a). 75K of them are used for our network training, while half of the remaining data is used as the test set, and the other half is used as the verification set.

Representation. We represent each floor plan in our dataset as a four-channel image (Fig. 3 (c)). Specifically, we store the inside mask in the first channel, the boundary information in the second channel, and the semantics of rooms and walls in the third channel. We use the specific integers (e.g., 0 for the living room) to represent different masks. In the fourth channel, we store extra information to distinguish between different rooms with the same labels. Different integers are used to distinguish different rooms with the same labels.

5 TWO-STAGE APPROACH

5.1 Locating rooms

5.1.1 Living room first strategy.

Key observations. The living room is an indispensable part in the modern residence and there are two key features: (1) it is usually located in the central area of the floor plan and (2) connected to most other rooms. Based on these observations, we develop a living room first strategy, which predicts the location of the living room first (Fig. 4). Once the living room is determined, the connectivity can be obtained by detecting the adjacencies between the living room and other rooms. A separated prediction model for the living room helps to improve the predictive accuracy and the overall rationality of the floor plan, as shown in Fig. 5.

Our iterative strategy. Inspired by [Ritchie et al. 2019; Wang et al. 2018], we propose the following iterative strategy (Fig. 4):

(1) Computing the location of the living room.
(2) Deciding what type of room to add and where.
(3) Deciding whether to add another room. If yes, go to Step (2); otherwise stop the algorithm.

5.1.2 Living room regression. We determine the location of the living room through a regression network.

Training dataset. We build a training dataset for the regression network with a multi-channel image as the input and the location of the living room as the regression target. Since the shape of the living room is a polygon, we represent the location of the living room as the centroid of the polygon. The multi-channel input includes the following information at each pixel, which defaults to 0:

- *Inside mask*: taking a value of 1 for the interior.
- *Boundary mask*: taking a value of 1 for the exterior walls and 0.5 for the front door.
- *Entrance mask*: taking a value of 1 for the front door.

Network architecture. We use a modified Resnet-34 [He et al. 2016] to extract the spatial features from the multi-channel input. The Resnet-34 architecture is modified to use 256×256 multi-channel images as inputs. We drop the last average pooling layer and fully connected (FC) layer, and append two convolution layers. We then use batch normalization (BN) and leaky rectified linear unit (leaky ReLU) between two convolution layers. We add an average pooling layer at the end of the network to obtain two-dimensional coordinates. We train the regression network using the robust smooth L_1 loss [Girshick 2015], which is less sensitive than the L_2 loss.

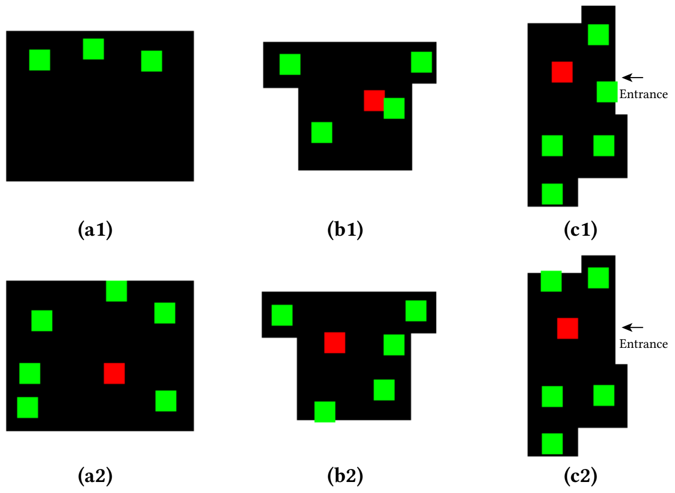


Fig. 5. Illustration of the importance of our living room first strategy. Top row: examples generated by our method without the living room regression network. Bottom row: examples generated by our method. The living room is shown as a red block. (a1) lacks a living room which is necessary for a residential building. (b1) chooses a location for another room that is very close to the centroid of the living room. In (c1), the connection between the living room and the entrance is blocked by another room.

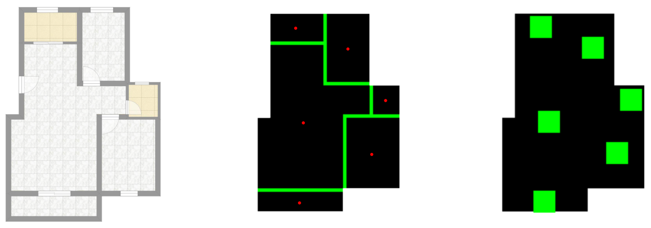

(a) Floor plan **(b)** Ground truth **(c)** Simplification

Fig. 6. Simplified representation. Given a floor plan (a), the ground truth (b) contains a set of rooms which are in the shape of a polygon. For the convenience of training, room shapes are simplified into small squares (green squares in (c)) centered on room centroids (red dots in (b)).

5.1.3 Room type and location. Given the previously generated room types and locations, we use an encoder-decoder network to determine a new room type and location.

Training dataset. To make the dataset amenable to train our prediction network, we simplify the representation of each room in the floor plan (Fig. 6). We use a small square (18 × 18) centered on the centroid of a room to represent this room. We then build a training dataset for the prediction network by randomly removing rooms from the floor plan of the dataset. Since our prediction network is based on the living room regression, we ensure that all samples in the training dataset contain a living room. The input for our location network is also a multi-channel image. Except for the channels in living room regression, we add additional room masks (i.e., the 18 × 18 squares at the room centroids) for each individual existing room in the floor plan. Finally, we add another channel that contains all existing room masks to allow the prediction network to have a global grasp of the overall existing room distribution. The target of our prediction network is a labeled image that is the same size as the input image. Aside from room types, we add three more labels: EXISTING (i.e., a pixel belonging to existing rooms), NOTHING (i.e., a pixel belonging to the interior but not occupied by any rooms), and OUTSIDE (i.e., a pixel belonging to the exterior).

Network architecture. For the network architecture, we borrow the atrous spatial pyramid pooling (ASPP) from [Chen et al. 2018]. We modify the ASPP architecture based on Resnet-34, and drop their last upsampling layer. Instead, we append an atrous convolution layer and a deconvolution layer at the end of the network. The deconvolution layer is used to upsample to the target size. We find this modification improves the accuracy. Since our location network performs a pixel-classification task, we use averaged pixel-wise cross entropy loss.

Sampling. Our prediction map is generated with noise, which is very common for large generative models. To reduce the impact of the noise, we adopt a more direct sampling method to obtain the type and location of the new room at the same time, as shown in Fig. 7. Our sampling process contains the following steps:

(1) For each pixel p with a room label in the prediction map, we compute the number of pixels (denoted as N_p), which have the same label as p and are in the 18 × 18 neighborhood of p.
(2) Choose the location of the pixel that has the greatest N_p as the center of the newly added room.

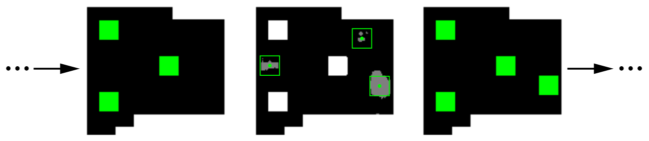

(a) Current scene **(b)** Sampling **(c)** Next scene

Fig. 7. Illustration of the process of location sampling. The type and location of the new room are chosen together when sampling. For the current scene (a), our network generates a prediction map (b). The white represents existing rooms (green squares in (a)) in the current scene, and the gray represents locations of candidate rooms that may be added. For each location, we consider a square area centered on this location (the hollow green square in (b)) and count the number of pixels having the same room label within the square area. The location with the most coverage pixel is sampled as the new room to obtain the next scene (c).

(3) The type of the new room is the same as the label of the chosen pixel and the new room is represented as the corresponding 18 × 18 square.

If more than one candidate pixel has the greatest N_p, we pick one randomly when sampling.

Discussions. We use an 18 × 18 square to improve the stability of the learning process. Since most pixels are with non-room labels, learning an 18×18 square is much easier than learning a single point with X-Y coordinates. It also weakens the category imbalances. Since we filter out the floor plans in the training dataset with very small rooms, the 18 × 18 square will not cause that squares of different rooms overlap with each other.

5.1.4 Continuing network. Given a set of generated room types and locations, we develop a continuing network to determine whether to terminate the algorithm.

To build the training dataset for the continuing network, we first adopt representation simplification for each floor plan in our dataset. Then we randomly choose half of the floor plans in our dataset as negative samples with label TRUE (i.e., continue) and the rest as positive samples with the label FALSE (i.e., terminate). For each room in one negative sample, we discard it with a 50% chance. The input for the continuing network is the same as the location network, except that we add a count vector, whose dimension is the number of room types, for the existing rooms to the input.

We adopt a similar network architecture to [Wang et al. 2018], but we modify it with Resnet-34. For the last three fully-connected layers, we only use leaky ReLU activation between these layers. The continuing network performs a binary classification task, so we use the standard binary cross entropy loss.

5.2 Locating walls

Prediction-based locating strategy. The next step is to locate walls to allocate space for each room. Previous constraint-based methods can be directly applied by formulating room locations into constrains. However, to generate a plausible floor plan, additional constraints, such as geometric constraints and topology constraints, should be provided. On the one hand, these inputs complicate the design process for users, since constraint design requires increased consideration. On the other hand, a system that includes too many constraints may have no solutions due to the contradictions between

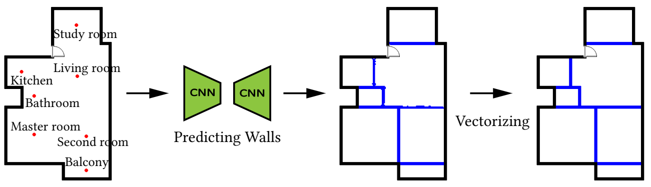


Fig. 8. Given the boundary and predicted rooms, the encoder-decoder network predicts walls on the pixel level. We use a post-processing step to convert the predicted wall map into the vector representation.

constraints. Therefore, we propose a prediction-based locating strategy, as shown in Fig. 8. Specifically, we first use an encoder-decoder network to predict walls in discrete pixels given the boundary and room locations. Then, a post-processing step is applied to convert the predicted walls into a vector representation.

5.2.1 Encoder-decoder prediction. After locating rooms, we should now have a series of room types and locations. The next step is to build walls based on this information. Given the boundary and predicted rooms, we use an encoder-decoder network to predict the locations of walls.

To build a training dataset for our wall locating network, as we did before, we first adopt a representation simplification for each floor plan in our dataset. The input for the network is the same as the room locating network. Our training target is a labeled image that is the same size as the input image. We have three kinds of labels for each pixel: WALL (i.e., a pixel belonging to walls), NOTHING (i.e., a pixel belonging to the interior but not part of any walls) and OUTSIDE (i.e., a pixel belonging to the exterior). We treat room doors as parts of walls, though we can predict locations for room doors at the same time. We will pursue this further in future work.

We use the same network architecture as in the room locating network due to the similar pixel-classification tasks. We then use averaged pixel-wise cross entropy loss to train our network.

5.2.2 Vectorization. Our network generates a wall map with discrete pixels representing walls. Although we obtain the approximate outlines of walls from the wall map, there are still a few issues remaining. We use a post-processing step to convert the wall map into a vector representation. We implement the vectorization using four steps, as shown in Fig. 9.

Step (1). Decompose and fit the noise predicted walls into rectangular parts. For a noise wall map (Fig. 9 (a)), we first perform the morphological closing operation. Then, the walls are decomposed into vertical and horizontal wall blocks, which are represented as their bounding boxes. Finally, these wall blocks are transformed with a fixed wall width, as shown in Fig. 9 (b).

Step (2). Connect and align the wall blocks to recover the complete walls (Fig. 9 (c)). Separated blocks are connected by computing the intersection of horizontal and vertical wall blocks. For further optimization, we adjust the wall blocks locally: (1) close wall blocks within a certain threshold are merged together; and (2) wall blocks are moved to align with other wall blocks or the exterior walls.

Step (3). Obtain the label for each pixel based on the predicted rooms and recovered walls. To derive room geometry from the normalized wall map, the predicted room locations are used to generate pixel-wise semantics according to the connection between pixels. We also compute the semantics of exterior walls, entrance, and interior walls. In Fig. 9 (d), we show the semantics.

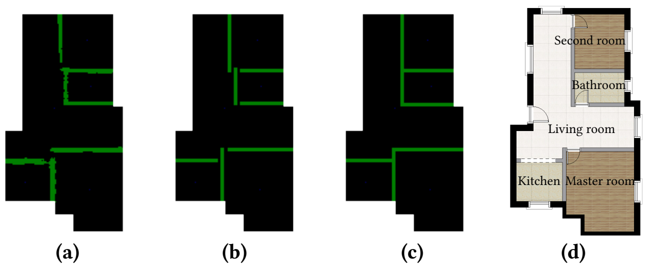


Fig. 9. Illustration of the process of vectorization. (a) Input noise wall map. (b) Decomposed wall blocks with a fixed width. (c) Complete walls. (d) Final result with room geometry, doors, and windows.

Step (4). Set doors and windows. The connectivity is determined based on the key observation that most doors are connected to the living room. We use this prior information to add passageways between the living room and other rooms. In addition, a passageway between any other connected rooms is added. Two empirical rules are proposed to place the passageways:

(1) Open-walls are placed in public rooms (i.e., kitchen and balcony); otherwise, we place ordinary doors.
(2) We place the door in the wall that minimizes the distance from the door to the front door.

The windows are placed based on two empirical rules:

(1) We set the French windows for the living room and small windows for the bathroom in the consideration of privacy; otherwise, we set the ordinary windows.
(2) Except for the bathroom, windows are laid out along the longest exterior wall segments within each room. We set at most one window at the center of the wall segments.

These heuristics are simple but available. Fig. 9 (d) shows the final result of the floor plan. A learning based method is expected to improve this process.

6 EXPERIMENTS

6.1 Implementation details

We use PyTorch to implement and train our networks. All models are trained and tested on an NVIDIA GeForce GTX 980 GPU. It takes around two days to train the regression network for the living room as well as the continuing network. Training takes around five days for both the room locating network and the wall locating network. The details of network architectures and training are available in the supplementary material.

Our regression network performs well with an average error of 0.82 meters (i.e. the actual distance from the predicted location to the ground truth) in the validation dataset. The continuing network reaches around 99% validation accuracy. In the pixel-classification task, we have a class imbalance problem, since most of the target labels are auxiliary labels (i.e. NOTHING and OUTSIDE). To fix this issue and improve the accuracy, we use weighted cross entropy loss with a weight of 1.25 for the room class or the wall class and 1 for auxiliary classes.

At synthesis time, it takes around four seconds to generate a vectorized floor plan given the building boundary as input.

6.2 User study

We perform comparisons to other methods or human-created floor plans through user studies.

Competitors. We select state-of-the-art methods as the competitors. The networks for indoor scene synthesis [Wang et al. 2018] can be used to locate rooms. We denote the network of [Wang et al. 2018] as ISSNet. Given the room locations, the MIQP-based method [Wu et al. 2018] can be used to determine the wall positions. Then, the method first uses ISSNet to locate rooms and then uses MIQP to locate walls is the first competitor (denoted as ISSNet+MIQP). Replacing the ISSNet of ISSNet+MIQP with our first stage approach for locating rooms is the second competitor (denoted as Stage1+MIQP). We substitute the MIQP of ISSNet+MIQP with our second stage approach for locating walls to defne the third competitor (denoted as ISSNet+Stage2). The human-created is the fourth competitor (denoted as Human). We provide more details for ISSNet and MIQP in the supplementary material.

User studies. Given a pair of floor plans that share the same boundary, the forced-choice comparison task is designed, similar to [Wang et al. 2018]. In each task, each participant should choose the floor plan that they think is more plausible. To be fair, we randomly choose examples used for user studies from our generated results. For each pair, the order of floor plans is randomized. We provide a questionnaire used for comparison to human-created floor plans in the supplementary material.

We use the same user study design for four competitors. Each user study includes 30 forced-choice comparison tasks. In one out of each of the 15 tasks, we perform a "vigilance test", in which an obviously wrong answer (specially, one floor plan with a randomized, jumbled arrangement of random rooms) is displayed.

For each user study, the number of participants enrolled is 86, 81, 85 and 99, respectively. The participants are classified into general users and designers who practice interior design as a profession. If one participant does not achieve 100% accuracy on the vigilance tests, we discard this response. The statistics of the participants in each comparison study are shown in Table 1. For each participant, we record the number (denoted as N) of floor plans that are generated by our method and preferred by that participant.

Results. The average and standard deviation for all the general users or designers in one user study are denoted as N_{avg} and N_{std}, respectively. The histograms in Fig. 10 show the distributions of N. Note that in Table 1 and Fig. 10, we only record the data from participants who pass the vigilance tests in each user study.

A score of around 14 (28 non-vigilance choices in total) indicates that the two methods are comparable. From the distributions in Fig. 10, our method is comparable to the competitor Human and outperforms the other three competitors.

Some participants failed the vigilance tests. We found about 7% of participants spent less than 1 minute and 21% of participants

Table 1. Statistics for user studies. We report the number of participants ("#part") who pass the vigilance tests, their age range, the average age a_{avg}, the standard deviation of their ages a_{dev}, the number of males and females, and the number of participants who practice interior design as a profession ("#prof"). For these professional designers, the time (in years) spent in this profession is recorded, and we report the average (y_{avg}) and the standard deviation (y_{dev}). We also record the time (in minutes) spent by participants in completing the study, and we report the average (t_{avg}) and the standard deviation (t_{dev}).

Competitor	#part	Age Range	$a_{\text{avg}}/a_{\text{dev}}$	Male, Female	#prof	$y_{\text{avg}}/y_{\text{dev}}$	$t_{\text{avg}}/t_{\text{dev}}$
ISSNet+MIQP	71	[18, 50]	25.71/5.37	48, 23	24	3.33/2.58	3.30/1.58
Stage1+MIQP	60	[18, 49]	24.90/4.50	41, 19	26	2.04/1.34	2.88/1.78
ISSNet+Stage2	58	[18, 50]	25.34/5.94	37, 21	29	2.41/2.08	3.23/1.58
Human	71	[20, 50]	26.62/5.60	45, 26	33	3.38/2.60	4.97/2.56

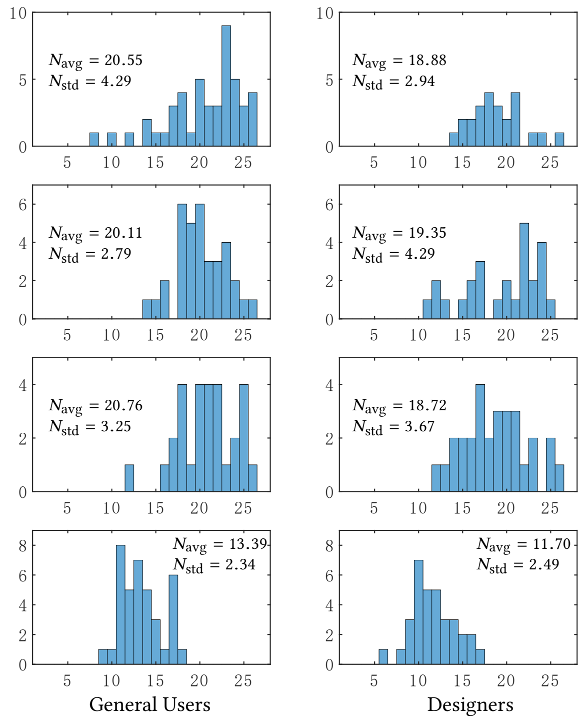


Fig. 10. Distributions of N. From the top, each successive row represents the results of comparisons of ISSNet+MIQP, Stage1+MIQP, ISSNet+Stage2, and Human.

spent less than 2 minutes finishing comparison tasks. So quite a few participants are perfunctory and dropped by the vigilance test.

We also observe that the average score of general users is less than designers'. For example, in the results of comparisons of Human, N_{avg} is 13.39 for general users and 11.70 for designers. Designers usually focus on the details (e.g., orientations and relative sizes between rooms) of the floor plans, while the general users often judge the plausibility with personal preference. Our method may not learn those details very well. So designers favored the human-designed layouts more than general users.

6.3 Comparisons

Comparison to ISSNet+MIQP. ISSNet computes category probabilities on a single pixel, which operates like an image-to-pixel function, and has problems with global consistency and noise suppression

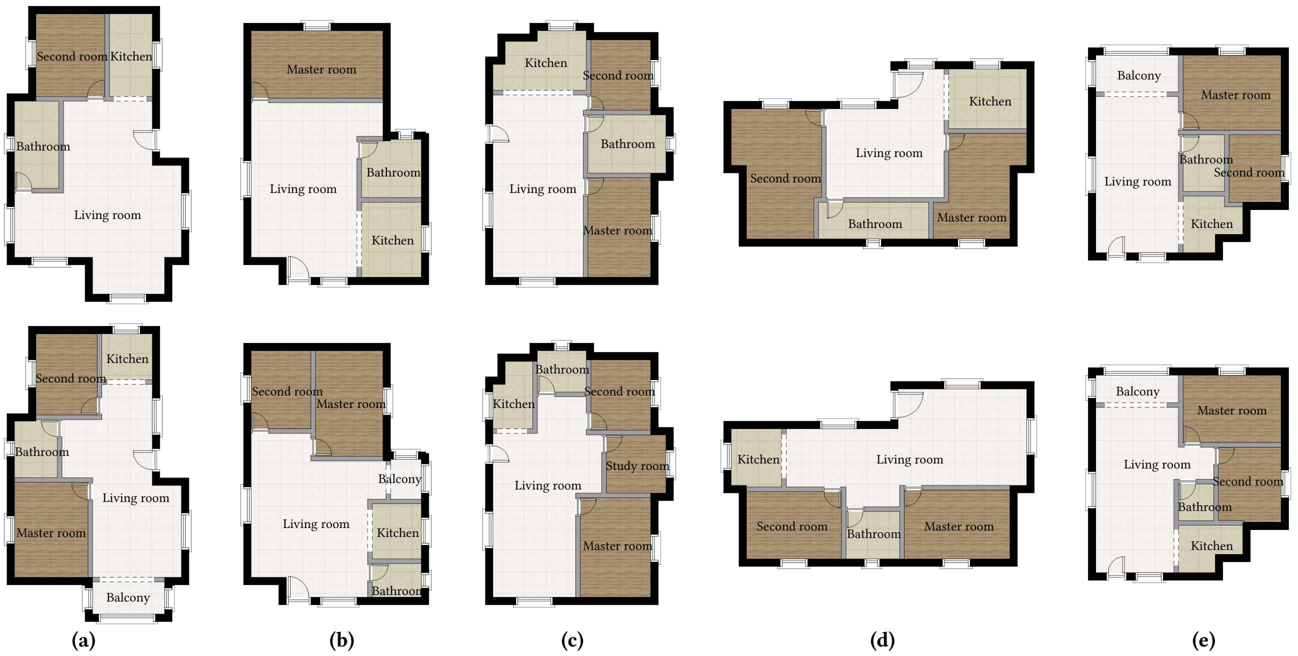


Fig. 11. Comparison to ISSNet+MIQP. Top row: results of ISSNet+MIQP. Bottom row: our results. In column (a), ISSNet+MIQP omits a necessary master room, which our method preserves. Column (b) shows that ISSNet+MIQP generates a sparse floor plan with only four rooms while the result of our method looks more realistic. ISSNet+MIQP synthesizes some rooms with unreasonable sizes in columns (c) and (d), such as the kitchen in column (c) and the bathroom in column (d). In column (e), ISSNet+MIQP generates a floor plan where the second room is blocked by the bathroom.

when sampling. MIQP is a hierarchical optimization with geometric constraints and topology constraints. To this end, room locations predicted by ISSNet are formulated into position constraints for MIQP. We do not use any connection constrains between two rooms since the connections are unclear. Moreover, too many constraints for MIQP may result in contradictions and lead to no solutions.

In the user study, average 20.55 floor plans of our method are preferred by general users and average 18.88 by designers, which indicates that participants preferred floor plans generated by our method to those generated by ISSNet+MIQP. Fig. 11 shows some representative results generated by ISSNet+MIQP and our method. ISSNet+MIQP generates floor plans with some necessary rooms missing which leads to sparse space allocation (columns (a) and (b)), and some rooms with unreasonable geometric sizes and shapes (columns (c) and (d)). In contrast, our method performs better in terms of global consistency and achieves better plausibility.

Comparison to Stage1+MIQP. Stage1+MIQP uses our first stage approach to locate rooms and then uses MIQP to generate walls. Similar to ISSNet+MIQP, the room locations predicted by our network serve as the location constraints for MIQP. We also do not add any connection constraints between two rooms to avoid the issue of constraint contradiction.

In the user study, average 20.11 floor plans of our method are preferred by general users and average 19.35 by designers, which indicates that participants preferred results generated by our method over those generated by Stage1+MIQP. The room types and locations predicted by our network become the initializations for MIQP. Although our network performs well at predicting room types and locations, MIQP fails in many examples with a few issues. The first issue is accessibility. In Fig. 12, columns (a) and (b) show that the master rooms generated by Stage1+MIQP are blocked by other rooms and cannot be entered. In column (c), the entrance is incorrectly connected to the bathroom. Geometric dimensions are another problem. In columns (d) and (e), MIQP generates some rooms with abnormal sizes, which are not suitable for residential buildings.

Comparison to ISSNet+Stage2. While ISSNet serves as an image-to-pixel function, our room locating network can be treated as an image-to-image function, which predicts the possible locations of all types of rooms in an input image. Our network has greater integrity and consistency in global layouts compared to ISSNet. We then compare our method to ISSNet+Stage2 using ISSNet for locating rooms and our wall locating network to generate walls.

In the user study, average 20.76 floor plans of our method are preferred by general users and average 18.72 by designers, which indicates that participants preferred results generated by our method. ISSNet has the global consistency problem and may introduce noise due to its image-to-pixel learning process, which causes that necessary rooms are omitted in many cases, as shown in Fig. 13 (a) and (b). Column (c) shows ISSNet+Stage2 generates a floor plan with only four rooms while our method generates a six-room floor plan. Although a four-room floor plan is acceptable, participants preferred the more intensively populated floor plan generated by our method. In column (e), ISSNet+Stage2 synthesizes an abnormal floor plan where the balcony is connected to the bathroom, which violates the privacy of the bathroom. Benefiting from the image-to-image learning process, our method performs better in terms

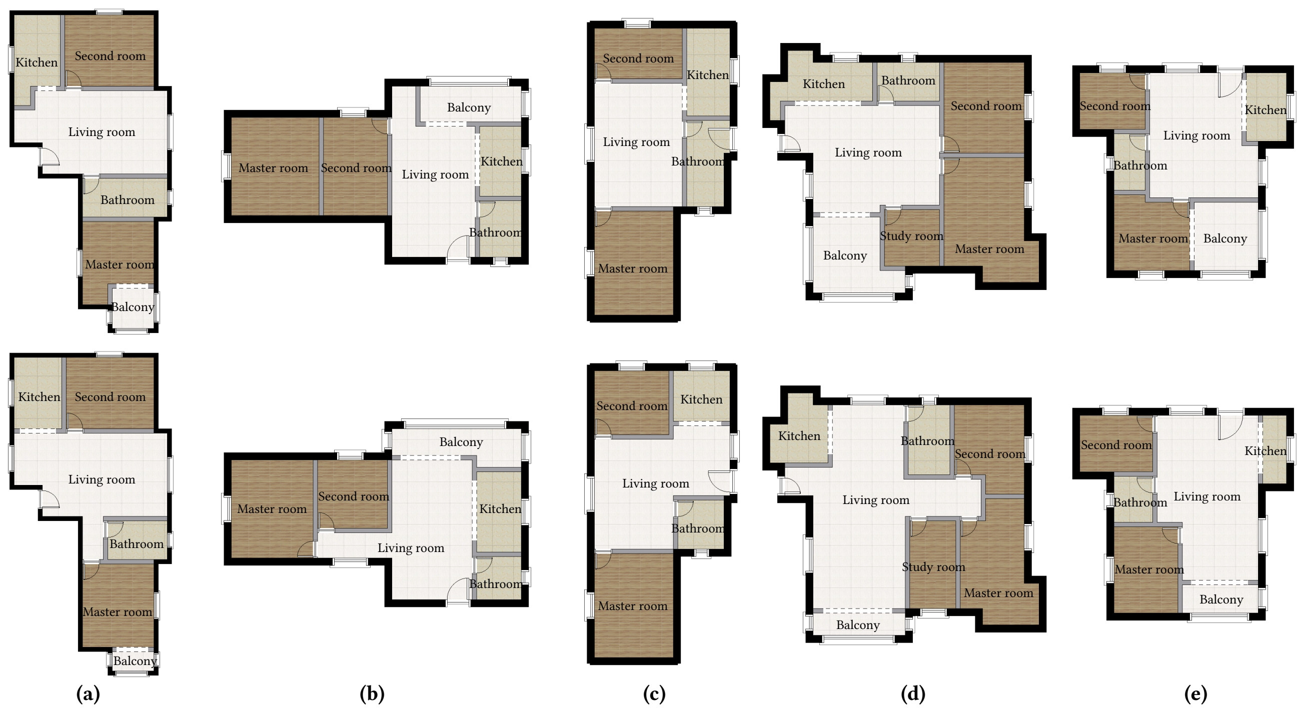


Fig. 12. Comparison to Stage1+MIQP. Top row: results of Stage1+MIQP. Bottom row: our results. Columns (a), (b) and (c) show the connection problem of Stage1+MIQP. Master rooms generated by Stage1+MIQP cannot be entered in columns (a) and (b), and the entrance, which should be connected to the living room, is incorrectly connected to the bathroom in column (c). In columns (d) and (e), the examples show that Stage1+MIQP synthesizes floor plans with unreasonable geometric dimensions. The balconies in column (d) and column (e) generated by Stage1+MIQP are much larger than normal.

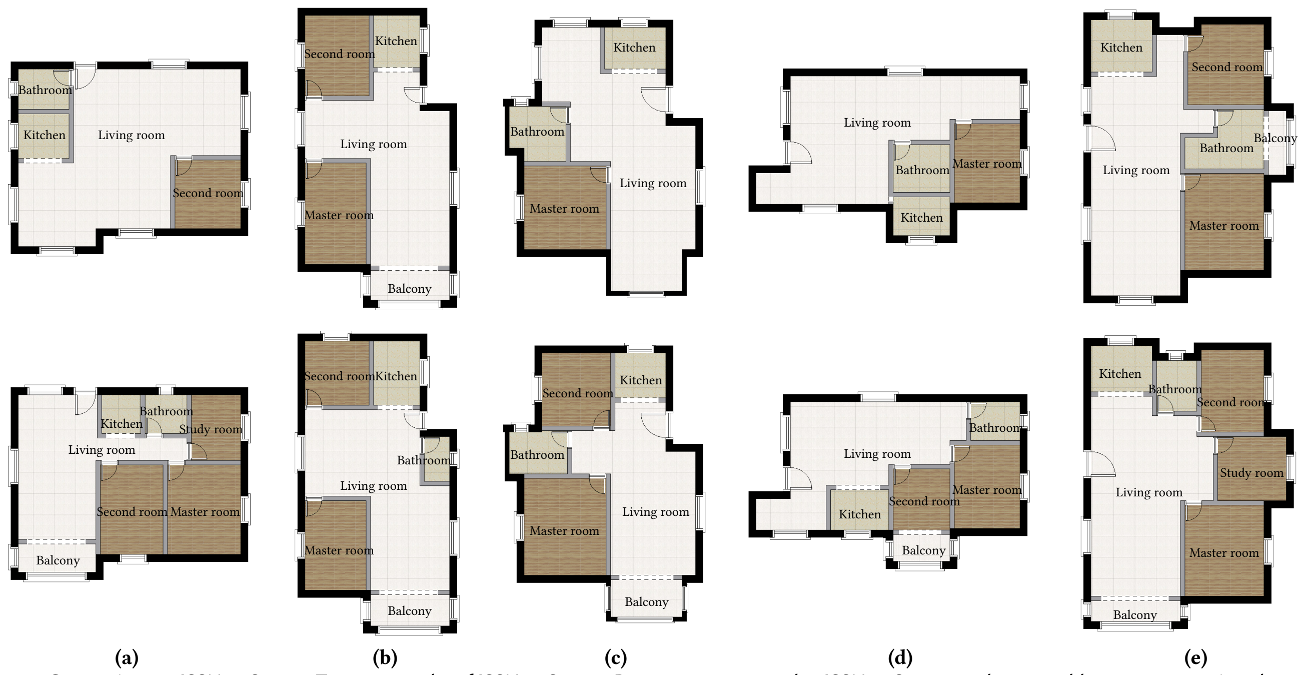


Fig. 13. Comparison to ISSNet+Stage2. Top row: results of ISSNet+Stage2. Bottom row: our results. ISSNet+Stage2 neglects to add a master room in column (a), and omits a bathroom in column (b). In column (c), ISSNet+Stage2 generates a sparse floor plan with only four necessary rooms while our method generates six. In column (d), the floor plan generated by ISSNet+Stage2 does not have enough space for users to enter the kitchen. The example of ISSNet+Stage2 in column (e) shows an unusual layout where the bathroom is equipped with a balcony.

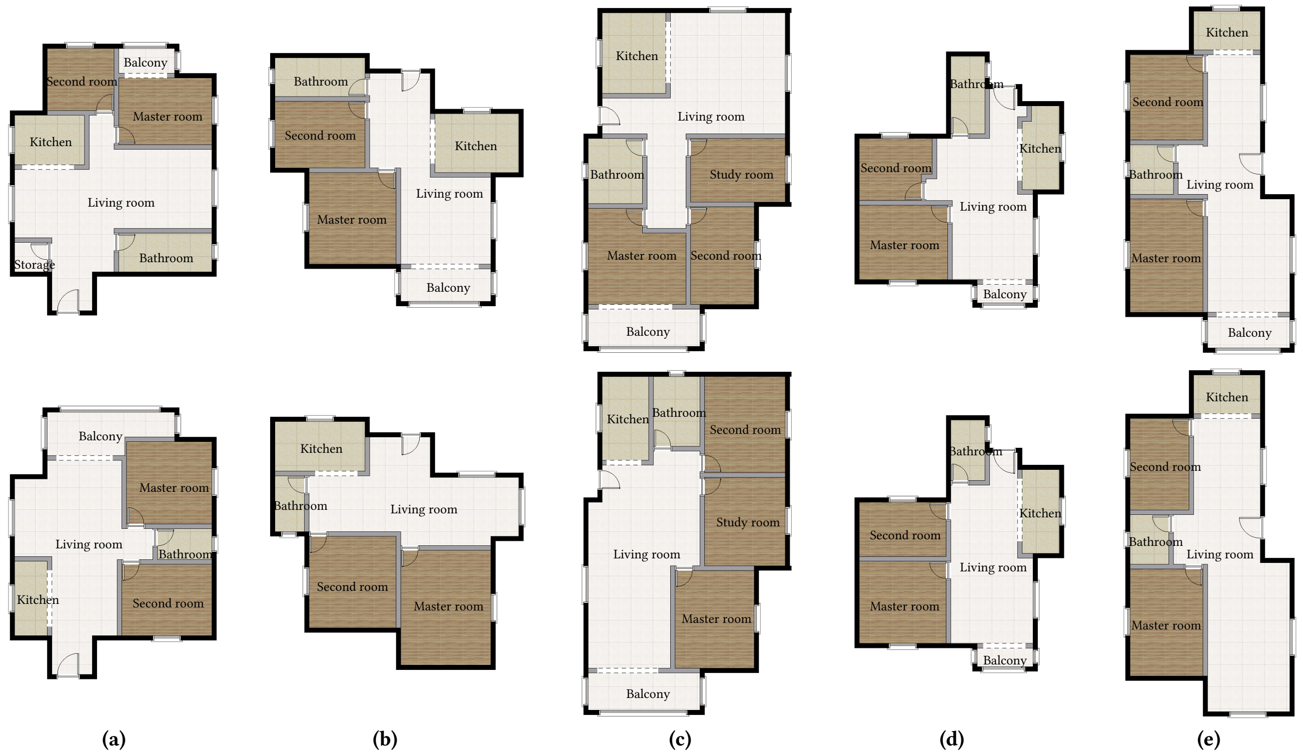


Fig. 14. Comparison to Human. Top row: results of Human. Bottom row: our results. Columns (a), (b), and (c) show that our method generates reasonable floor plans that differ slightly from the original human-designed results. Columns (d) and (e) show that our method generates floor plans similar to original human-designed results with only few differences.

of room distribution and thus achieves better global integrity and consistency compared to ISSNet+Stage2.

Comparison to Human. We finally compare the floor plans generated by our method with original floor plans from our dataset. These floor plans are designed manually using a combination of intuition, prior experience, and professional knowledge.

In the user study, average 13.39 floor plans of our method are preferred by general users and average 11.70 by designers. Participants show a slight preference for the human-designed floor plans to those generated by our method. In Fig. 14, columns (d) and (e) illustrate that our method generates floor plans similar to those created by humans with only few differences, which proves the validity of our method. For columns (a), (b), and (c), our method provides different design options compared to the original human-created floor plans. Note that the human-created results of Fig. 11, Fig. 12, and Fig. 13 are shown in the supplementary material.

6.4 Evaluation and discussion

Room constraints. Our method can support the location constraint that specifies the locations of some rooms. According to personal preferences and requirements, users first choose the locations for several specific rooms inside the given boundary. Then our method generates other rooms and respects user's design intent. Fig. 15 shows several examples generated by our method, where one or more room locations are specified by users.

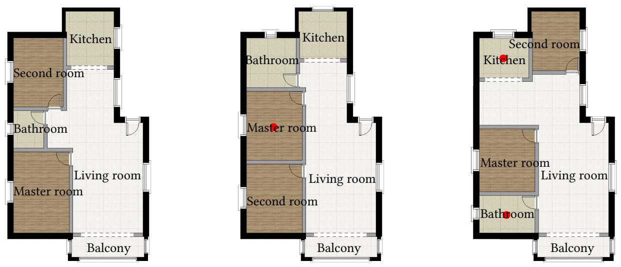


Fig. 15. Examples of floor plans synthesized by our method, given one or two room locations specified by users (red dots). From left to right: no specified room, specified master room, and specified kitchen and bathroom.

Nearest neighbors. We examine the ability of our method to generate floor plans beyond its training dataset, which does not just memorize the training dataset. Given a boundary, we generate our result and use the boundary to search the nearest neighbor in the training dataset. All of our results are different from the nearest neighbors. Fig. 16 shows three pairs of them. In addition, given the same boundary as input, our method can also generate different results from the human-created results, as shown in the user study. It indicates the result variety and generalization of our method.

Non axis-aligned input. Although RPLAN only contains floor plans with axis-aligned walls, our method still works well when the input boundary is non axis-aligned. We show six examples in Fig. 17

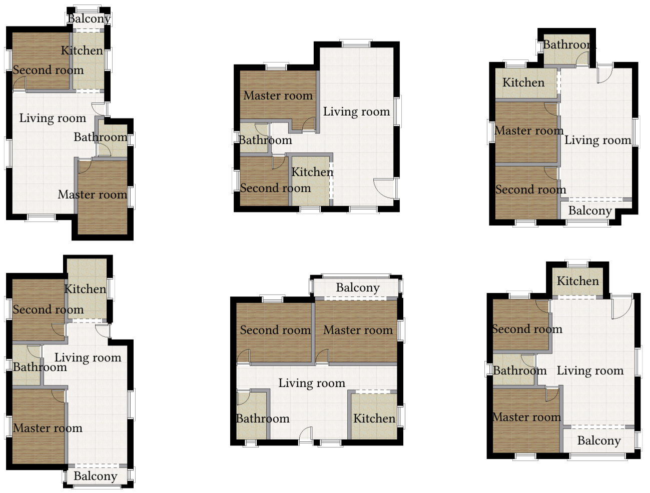


Fig. 16. Comparison to the nearest neighbors in the training dataset. Top row: the nearest neighbors. Bottom row: our synthesized results.

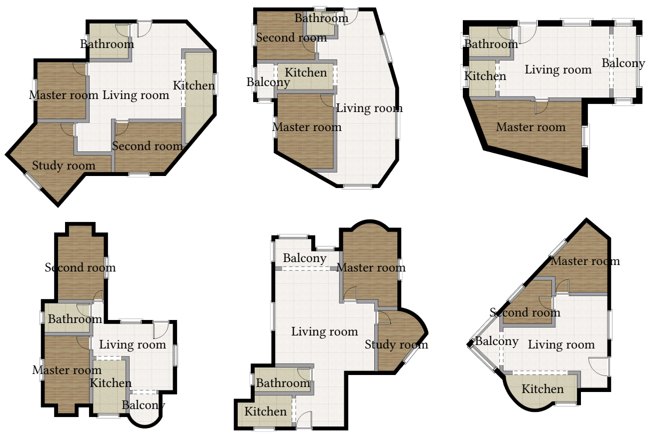


Fig. 17. Floor plan generation for non axis-aligned boundaries. In the bottom row, the examples contain curved walls.

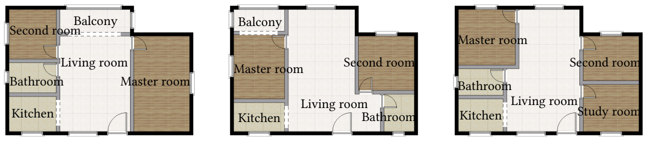


Fig. 18. Synthesizing multiple floor plans given the same boundary as input.

and more examples in the supplementary material. This indicates that our model generates floor plans beyond its training dataset.

Multiple floor plans. Since we sample a new room based on the maximum number of coverage points in the prediction map, only one solution is generated by our sampling method. To generate multiple floor plans given the same boundary as input, we generate a sample based on the probability distribution of predicted rooms. Fig. 18 shows one example, where our method takes the same boundary as input and generates multiple floor plans.

Failure cases. While our method generates plausible floor plans, it does fail in some cases, as shown in Fig. 19. Our method may generate some rooms with inappropriate arrangements. As shown in (a), the door to the master rooms is too small for users to enter. Another failure case is caused by poor wall predictions. In (b), our network predicts walls with a lot of noise, which is difficult to deal with in post-processing, so the vectorization result is problematic. Although our post-processing works well, it cannot handle cases when necessary wall pixels are missing. The broken walls are hard for our post-processing to handle, which leads to incorrect space allocations in (c). We consider floor plans with incorrect walls as the problematic layouts (Fig. 19). We test the frequency of problematic floor plans using 100 generated examples. We generate 94 plausible floor plans and the problematic frequency is low.

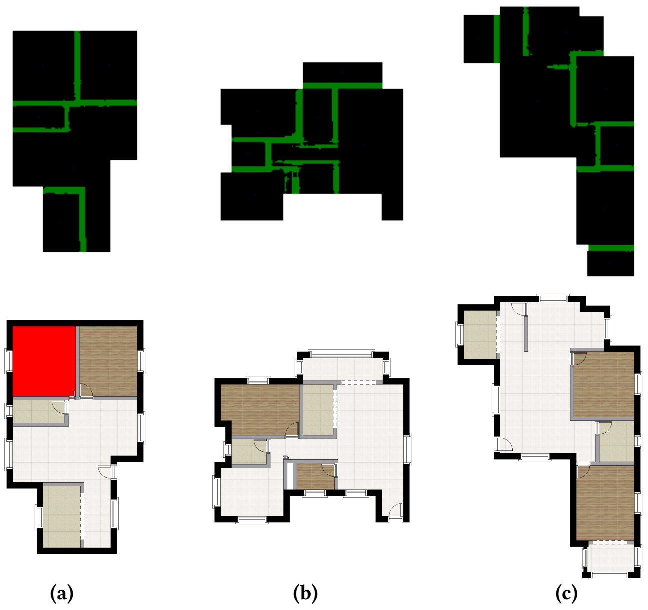


Fig. 19. Typical failed floor plans generated by our method. Top row: walls predicted by our method. Bottom row: vectorization results of our method. (a) shows insufficient space for a door to the master room (the red area). (b) shows too much noise in the original wall map generated by our method, and (c) shows broken walls. Both examples lead to incorrect space allocation in the vectorization.

7 CONCLUSION

Our method provides a novel data-driven technique for automatically and efficiently generating floor plans for residential buildings with fixed boundary. By imitating the human design process, we propose a two-stage approach to generate floor plans. To effectively train our networks, a large-scale dataset containing more than 80K floor plans from real residential buildings is presented. By comparing the plausibility of floor plans through user studies, our method outperforms state-of-the-art methods, and in some cases our floor plans are comparable to human-created ones.

Constraints in real life. In general, we propose an automatic algorithm for generating floor plans from the given boundaries. However, in actual design work, design with additional constraints (e.g., constrained square footage, support walls, and house orientation) seems to be more meaningful and challenging. One simple and straightforward solution is to introduce more generative models for these additional constraints. Or, we could turn these unfamiliar constraints into room constraints which our method can deal with. We would like to explore more in the future.

More types of buildings. Currently, our method only designs floor plans for one-story residential buildings. For multi-story homes, stairs are necessary to connect two consecutive floors. Our method can be applied by conceptualizing the stairs as a type of room. We generate the first floor plan containing the stairs, and then the second floor plan is generated based on the first. The stairs generated in the first floor should serve as a constraint for the generation of the second floor. However, in the real world, stairs have special shapes and location considerations, and it may be hard for our deep networks to control the generation of stairs. Our two-stage approach, even without the living-room-first-strategy, can also be extended to other types of buildings, such as office buildings, shopping malls, and supermarkets. However, the prerequisite for all these considerations is that we must have relevant data.

ACKNOWLEDGMENTS

We would like to thank Kai Wang for providing their implementation of [Wang et al. 2018], user study participants for evaluating our results, and the anonymous reviewers for their constructive suggestions and comments. This work is supported by the National Natural Science Foundation of China (61802359, 61672482, 11626253) and the Fundamental Research Funds for the Central Universities (WK0010460006, WK0010450004).

REFERENCES

Daniel G. Aliaga, Carlos A. Vanegas, and Bedrich Benes. 2008. Interactive Example-based Urban Layout Synthesis. *ACM Trans. Graph.* 27, 5 (2008), 160:1–160:10.

Scott A Arvin and Donald H House. 2002. Modeling architectural design objectives in physically based space planning. *Automation in Construction* 11, 2 (2002), 213 – 225.

Alper Aydemir, Patric Jensfelt, and John Folkesson. 2012. What can we learn from 38,000 rooms? Reasoning about unexplored space in indoor environments. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 4675–4682.

Arash Bahrehmand, Thomas Batard, Ricardo Marques, Alun Evans, and Josep Blat. 2017. Optimizing layout using spatial quality metrics and user preferences. *Graphical Models* 93 (2017), 25 – 38.

Fan Bao, Dong-Ming Yan, Niloy J. Mitra, and Peter Wonka. 2013. Generating and Exploring Good Building Layouts. *ACM Trans. Graph.* 32, 4 (2013), 122:1–122:10.

Guoning Chen, Gregory Esch, Peter Wonka, Pascal Müller, and Eugene Zhang. 2008. Interactive Procedural Street Modeling. *ACM Trans. Graph.* 27, 3 (2008), 103:1–103:10.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. 2018. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence* 40, 4 (2018), 834–848.

Tian Feng, Lap-Fai Yu, Sai-Kit Yeung, KangKang Yin, and Kun Zhou. 2016. Crowd-driven Mid-scale Layout Design. *ACM Trans. Graph.* 35, 4 (2016), 132:1–132:14.

Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan. 2012. Example-based Synthesis of 3D Object Arrangements. *ACM Trans. Graph.* 31, 6 (2012), 135:1–135:11.

Ross Girshick. 2015. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*. 1440–1448.

Evan Hahn, Prosenjit Bose, and Anthony Whitehead. 2006. Persistent Realtime Building Interior Generation. In *Proceedings of the 2006 ACM SIGGRAPH Symposium on Videogames*. 179–186.

Mikako Harada, Andrew Witkin, and David Baraff. 1995. Interactive Physically-based Manipulation of Discrete/Continuous Models. In *Proc. SIGGRAPH*. 199–208.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

Mark Hendrikx, Sebastiaan Meijer, Joeri Van Der Velden, and Alexandru Iosup. 2013. Procedural Content Generation for Games: A Survey. *ACM Trans. Multimedia Comput. Commun. Appl.* 9, 1 (2013), 1:1–1:22.

Hao Hua. 2016. Irregular architectural layout synthesis with graphical inputs. *Automation in Construction* 72 (2016), 388 – 396.

Ahti Kalervo, Juha Ylioinas, Markus Häikiö, Antti Karhu, and Juho Kannala. 2019. CubiCasa5K: A Dataset and an Improved Multi-Task Model for Floorplan Image Analysis. In *Scandinavian Conference on Image Analysis*. Springer, 28–40.

Jianan Li, Tingfa Xu, Jianming Zhang, Aaron Hertzmann, and Jimei Yang. 2019. Layout-GAN: Generating Graphic Layouts with Wireframe Discriminator. In *International Conference on Learning Representations*.

Robin S Liggett. 2000. Automated facilities layout: past, present and future. *Automation in Construction* 9, 2 (2000), 197 – 215.

Chen Liu, Jiaye Wu, and Yasutaka Furukawa. 2018. FloorNet: A Unified Framework for Floorplan Reconstruction from 3D Scans. In *ECCV 2018*. 203–219.

Chen Liu, Jiajun Wu, Pushmeet Kohli, and Yasutaka Furukawa. 2017. Raster-to-Vector: Revisiting Floorplan Transformation. In *ICCV 2017*. 2214–2222.

Han Liu, Yong-Liang Yang, Sawsan AlHalawani, and Niloy J. Mitra. 2013. Constraint-aware interior layout exploration for pre-cast concrete-based buildings. *The Visual Computer* 29, 6 (2013), 663–673.

Chongyang Ma, Nicholas Vining, Sylvain Lefebvre, and Alla Sheffer. 2014. Game Level Layout from Design Specification. *Comput. Graph. Forum (EG)* 33, 2 (2014), 95–104.

Paul Merrell, Eric Schkufza, and Vladlen Koltun. 2010. Computer-generated Residential Building Layouts. *ACM Trans. Graph.* 29, 6 (2010), 181:1–181:12.

Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. 2011. Interactive Furniture Layout Using Interior Design Guidelines. *ACM Trans. Graph.* 30, 4 (2011), 87:1–87:10.

Jeremy Michalek, Ruchi Choudhary, and Panos Papalambros. 2002. Architectural layout design optimization. *Engineering optimization* 34, 5 (2002), 461–484.

Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. 2006. Procedural Modeling of Buildings. *ACM Trans. Graph.* 25, 3 (2006), 614–623.

Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. 2014. Learning Layouts for Single-Page Graphic Designs. *IEEE. T. Vis. Comput. Gr.* 20, 8 (2014), 1200–1213.

Chi-Han Peng, Yong-Liang Yang, Fan Bao, Daniel Fink, Dong-Ming Yan, Peter Wonka, and Niloy J. Mitra. 2016. Computational Network Design from Functional Specifications. *ACM Trans. Graph.* 35, 4 (2016), 131:1–131:12.

Chi-Han Peng, Yong-Liang Yang, and Peter Wonka. 2014. Computing Layouts with Deformable Templates. *ACM Trans. Graph.* 33, 4 (2014), 99:1–99:11.

Roberto J. Rengel. 2011. *The Interior Plan: Concepts and Exercises.* Fairchild Books.

Daniel Ritchie, Kai Wang, and Yu an Lin. 2019. Fast and Flexible Indoor Scene Synthesis via Deep Convolutional Generative Models. In *CVPR 2019*.

Eugénio Rodrigues, Adélio Rodrigues Gaspar, and Álvaro Gomes. 2013a. An approach to the multi-level space allocation problem in architecture using a hybrid evolutionary technique. *Automation in Construction* 35 (2013), 482–498.

Eugénio Rodrigues, Adélio Rodrigues Gaspar, and Álvaro Gomes. 2013b. An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, Part 1: Methodology. *Computer-Aided Design* 45, 5 (2013), 887–897.

Eugénio Rodrigues, Adélio Rodrigues Gaspar, and Álvaro Gomes. 2013c. An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, Part 2: Validation and performance tests. *Computer-Aided Design* 45, 5 (2013), 898–910.

Julian F. Rosser, Gavin Smith, and Jeremy G. Morley. 2017. Data-driven estimation of building interior plans. *International Journal of Geographical Information Science* 31, 8 (2017), 1652–1674.

Ruben M. Smelik, Tim Tutenel, Rafael Bidarra, and Bedrich Benes. 2014. A Survey on Procedural Modelling for Virtual Worlds. *Comput. Graph. Forum* 33, 6 (2014), 31–50.

Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. 2017. Semantic scene completion from a single depth image. In *CVPR*. 1746–1754.

Kai Wang, Manolis Savva, Angel X. Chang, and Daniel Ritchie. 2018. Deep Convolutional Priors for Indoor Scene Synthesis. *ACM Trans. Graph.* 37, 4 (2018), 70:1–70:14.

Wenming Wu, Lubin Fan, Ligang Liu, and Peter Wonka. 2018. MIQP-based Layout Design for Building Interiors. *Comput. Graph. Forum (EG)* 37, 2 (2018), 511–521.

Shang-Ta Yang, Fu-En Wang, Chi-Han Peng, Peter Wonka, Min Sun, and Hung-Kuo Chu. 2019. DuLa-Net: A Dual-Projection Network for Estimating Room Layouts from a Single RGB Panorama. In *CVPR 2019*.

Yong-Liang Yang, Jun Wang, Etienne Vouga, and Peter Wonka. 2013. Urban Pattern: Layout Design by Hierarchical Domain Splitting. *ACM Trans. Graph.* 32, 6 (2013), 181:1–181:12.

Lap-Fai Yu, Sai-Kit Yeung, Chi-Keung Tang, Demetri Terzopoulos, Tony F. Chan, and Stanley J. Osher. 2011. Make It Home: Automatic Optimization of Furniture Arrangement. *ACM Trans. Graph.* 30, 4 (2011), 86:1–86:12.

Chuhang Zou, Alex Colburn, Qi Shan, and Derek Hoiem. 2018. Layoutnet: Reconstructing the 3d room layout from a single rgb image. In *CVPR 2018*. 2051–2059.