DOI: 10.1111/cgf.15007

EUROGRAPHICS 2024 / A. Bermano and E. Kalogerakis
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 43 (2024), Number 2

Raster-to-Graph: Floorplan Recognition via Autoregressive Graph Prediction with an Attention Transformer

Sizhe Hu Wenming Wu [†] Ruolin Su Wanni Hou Liping Zheng Benzhu Xu

Hefei University of Technology, China

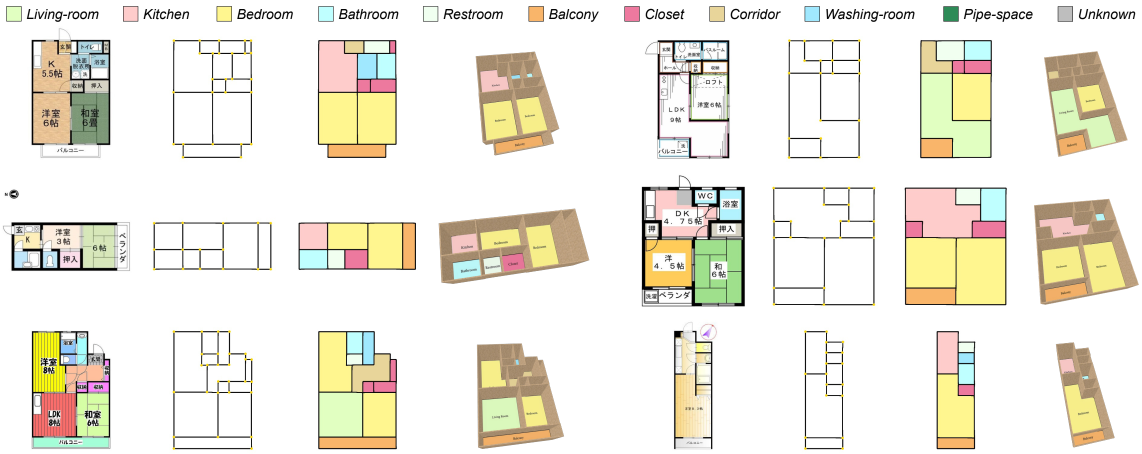


Figure 1: *We propose a novel floorplan vectorization framework for recognizing the semantics and structures of rasterized floorplans. Here shows a gallery of rasterized floorplans and the corresponding vectorized results obtained through our framework. For each group, from left to right: a given rasterized floorplan, a structural graph obtained by our method, a vectorized floorplan with various colors representing different categories of rooms and a popup 3D model of the floorplan.*

Abstract

Recognizing the detailed information embedded in rasterized floorplans is at the research forefront in the community of computer graphics and vision. With the advent of deep neural networks, automatic floorplan recognition has made tremendous breakthroughs. However, co-recognizing both the structures and semantics of floorplans through one neural network remains a significant challenge. In this paper, we introduce a novel framework Raster-to-Graph, which automatically achieves structural and semantic recognition of floorplans. We represent vectorized floorplans as structural graphs embedded with floorplan semantics, thus transforming the floorplan recognition task into a structural graph prediction problem. We design an autoregressive prediction framework using the neural network architecture of the visual attention Transformer, iteratively predicting the wall junctions and wall segments of floorplans in the order of graph traversal. Additionally, we propose a large-scale floorplan dataset containing over 10,000 real-world residential floorplans. Our autoregressive framework can automatically recognize the structures and semantics of floorplans. Extensive experiments demonstrate the effectiveness of our framework, showing significant improvements on all metrics. Qualitative and quantitative evaluations indicate that our framework outperforms existing state-of-the-art methods. Code and dataset for this paper are available at: https://github.com/HSZVIS/Raster-to-Graph.

CCS Concepts

• ***Computing methodologies*** → ***Shape modeling; Computer vision;***

1. Introduction

Residential floorplans are usually rasterized to images for publication while discarding geometry structure and semantics in-

[†] The corresponding author: wwming@hfut.edu.cn (Wenming Wu)

formation, which limits further analysis, synthesis, or modification [LWKF17]. Floorplan recognition aims to recognize semantics and reconstruct structures from rasterized floorplans, which is of great interest to the community of computer graphics and vision. Converting rasterized floorplans into vector form would bring significant advancements in broader domains such as architectural design, scene modeling, virtual reality, and robot navigation.

Recently, automatic floorplan recognition has made tremendous breakthroughs attributed to the emergence of deep neural networks. However, co-recognizing both the structures and semantics of floorplans through one neural network remains a significant challenge. Previous works have either focused solely on floorplan semantic recognition, neglecting structural reconstruction [ZLYF19, LWG*21], or leaved structural reconstruction as a secondary stage [LWKF17, LZYZ21], relying heavily on heuristic post-processing or additional optimization. In addition, the floorplan dataset also remains one of the main challenges, with existing publicly available datasets having either insufficient floorplan samples [LSK*15, LLC*20, LWKF17] or poor data quality [KYH*19], limiting the use and performance of deep neural networks. [LZYZ21] introduces a dataset comprising 7,000 annotated residential floorplans. However, this dataset has not been made publicly available.

In this paper, we propose Raster-to-Graph, or R2G for short, a novel automatic framework achieving floorplan recognition on both structures and semantics. Our goal is straightforward, to recognize floorplans through one neural network. Existing deep models for structural reconstruction of floorplan are non-autoregressive [ZNF20, CQF22]. They suffer from a lack of context awareness as they do not consider the sequential dependencies between elements, resulting in lower prediction accuracy and redundant predictions. To this end, we turn to autoregressive floorplan recognition. The main idea is inspired by the deep generative model for floorplan synthesis [SWL*22], which iteratively generates the structure of a floorplan through an autoregressive model.

Concretely, we represent a vector floorplan as a structural graph, whose nodes are wall junctions and edges are wall segments. To recognize floorplan semantics, we consider the room categories as one of the node attributes. In this way, the task of floorplan recognition can be viewed as structural graph prediction. Given a rasterized floorplan, we train an autoregressive model to iteratively predict wall junctions and wall segments in the order of graph traversal. The autoregressive model is the use of a visual attention Transformer [ZSL*20] with careful design for the recognition targets. Furthermore, we propose a large-scale floorplan dataset containing more than 10,000 annotated floorplans of realistic residential buildings. Each floorplan is represented as vector graphics composed of labeled junctions, walls, and rooms.

Our autoregressive model can automatically recognize the structures and semantics of floorplans. Extensive experiments demonstrate the validity of our automatic recognition framework, making significant improvements on all the metrics. Qualitative and quantitative evaluations show that the proposed method outperforms the existing state-of-the-art. This paper seeks to push the frontier of neural network architecture for floorplan recognition. We contribute the following:

- An automatic recognition framework to obtain high-quality vectorized floorplans from rasterized images through one neural network.
- A novel autoregressive model that iteratively predicts structures and semantics of floorplans in the order of graph traversal.
- A large-scale floorplan dataset containing more than 10,000 realistic residential floorplans with dense annotations both on structures and semantics. To the best of our knowledge, this is currently the largest dataset available for floorplan recognition. The dataset has much potential to inspire more research.

2. Related work

Previous research has often considered floorplan recognition as either semantic recognition, or structural recognition. In the work where both semantic and structural are considered, semantic recognition still forms the system core, while structural reconstruction is typically addressed as a secondary stage.

2.1. Semantic recognition of floorplan

Semantic recognition is of great crucial for floorplan recognition, involving the prediction of various element categories in the floorplan. Many of these methods [DXS17, ZLYF19, ZHZD20, KKY21, XYA*21, LWG*21] adopt semantic segmentation techniques based on the convolutional neural network (CNN), providing pixel-level recognition results. [DXS17] proposes a method that combines wall segmentation, object detection, and optical character recognition to segment walls and vectorize elements in the floorplan. [ZLYF19] treates floorplan recognition as pixel-level semantic segmentation. They model a hierarchy of floorplan elements by analysing spatial relationships, and introduces a spatial context module with a room-boundary-guided attention mechanism to guide the room segmentation. [XYA*21] focuses on detecting elements like walls, doors, and rooms through semantic segmentation and introduces a room boundary attention aggregation mechanism for segmenting floorplans. Different from [ZLYF19], it enables mutual guidance between the boundary and rooms. To parse floorplans, [LWG*21] presents a framework that combines semantic neural networks with a postprocessed room segmentation. These semantic segmentation-based methods often overlook structural elements such as walls in floorplans. They are unable to provide users with usable vectorized recognition results. [YKSE23] formulate the floorplan reconstruction problem as polygon estimation and used a single-stage Transformer to predict room types and sequences of corners. However, due to their polygon representation, they do not obtain aligned wall structures.

2.2. Structural reconstruction of floorplan

Structural reconstruction [ZNF20, NF20, SY21, FZL*21, CQF22] is also one of the goals of floorplan recognition, which involves vectorizing structural elements in the floorplan. [ZNF20] proposes a message-passing neural architecture Conv-MPN to reconstruct outdoor buildings as floorplans from a single RGB image. Similarly, [NF20] proposes a method that first employs a CNN to detect geometric primitives and infer their relationships, then utilizes integer programming to obtain a floorplan. Recently, HEAT [CQF22] introduces a two-stage network structure for 2D planar structure extraction. HEAT first uses a corner detection network and non-maximum suppression to acquire corners and then pair them up to form edges, followed by edge classification, which achieves state-of-the-art structural reconstruction for architectural floorplans. [FLPH21] uses the traditional shape detection method to detect primitives for partitioning the space into polygons, and obtains walls through optimization. These structural reconstruction methods only reconstruct floorplan structures, overlooking the semantic information. In contrast, our method can simultaneously capture the semantics and the structure of floorplans.

2.3. Structure-semantic recognition of floorplan

Some methods [LWKF17, CLWF19, JYY20, KPKY21, WSC*21, SRFL21, LZYZ21, DWLZ21, YWY21, PK21] attempt to obtain vectorized representations of rasterized floorplans after performing semantic segmentation. These methods heavily rely on additional post-processing or algorithm optimizations to reconstruct the structure and enhance recognition accuracy. [LWKF17] obtains semantic segmentations of floorplan rooms and wall junctions using CNN and then solves the integer programming problem to obtain vectorized floorplans. In [JYY20], neural networks are employed in the segmentation of floorplan walls and doors, and then traditional image processing techniques such as Harris corner detection are used to convert wall and door segmentations into vector form. [LZYZ21] proposes a floorplan vectorization method consisting of recognition and reconstruction. After object detection, pixel-level structural elements are identified through semantic segmentation, and reconstruction is achieved through heuristic optimization and post-processing. [DWLZ21] treats the semantic segmentation of floorplans as an image-to-image translation problem and use Generative Adversarial Networks (GANs) to obtain segmentation of primitives. They vectorize floorplan elements using handcrafted filters and perform room classification using GNNs. In [YWY21], a rule-based graph transformation is applied on top of the semantic segmentation of floorplans to obtain bubble diagrams. This bubble diagram representation is a vectorized result that is more conducive to subsequent utilization compared to segmentation results. The use of heuristic post-processing or additional optimization reduces the scalability of methods, making them overly complex and less suitable for generalization. In contrast, our method directly utilizes neural networks to predict the structure and semantic information, naturally converting it into vectorized results.

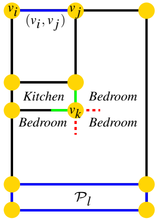
2.4. Transformer-based object detection

The neural network architecture of the visual attention Transformer is used in our framework. Transformer [VSP*17] is initially introduced in the field of natural language processing. At the core of the Transformer is the multi-head attention mechanism, which learns long-range dependencies between elements. DETR [CMS*20] brings the Transformer into the task of object detection which uses ResNet to extract feature maps and employs the Transformer to capture long-range dependencies between pixel-level image features. Deformable DETR [ZSL*20] shares a similar architecture with DETR but introduces adaptive sampling attention via linear projection layers, known as the deformable attention mechanism. The deformable attention mechanism in Deformable DETR significantly reduces computational complexity, allowing for the use of lower-level and larger-scale feature maps. In our task, on one hand, detecting wall junctions in floorplans can be viewed as small object detection. On the other hand, the features of floorplan images exhibit complex long-range dependencies. Additionally, the attention between floorplan elements should be sparse, making Deformable DETR an ideal choice for floorplan recognition. Furthermore, Deformable DETR simplifies our framework. To this end, we design an autoregressive prediction framework using the neural network architecture of Deformable DETR.

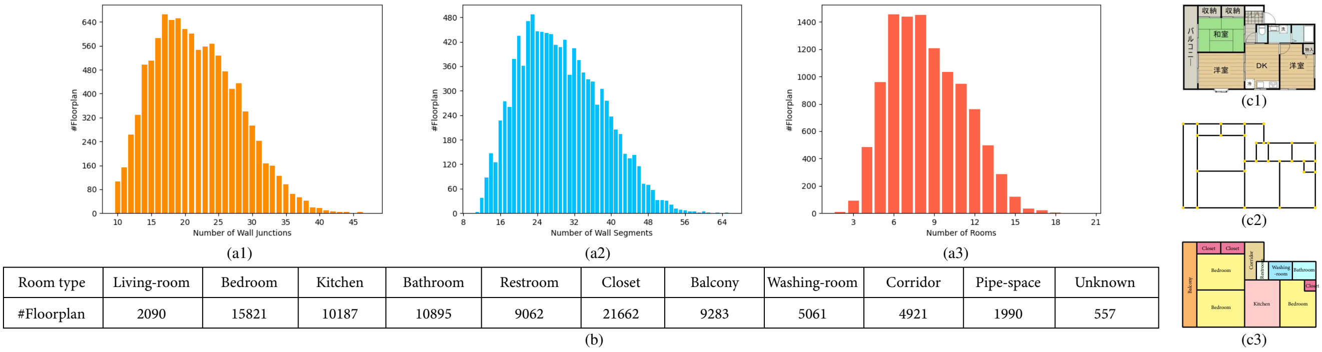
At last, it is essential to emphasize that Raster-to-Graph is inspired by WallPlan [SWL*22] but differs significantly. In terms of the goal, WallPlan is used to generate floorplans with given boundary constraints, while Raster-to-Graph is designed for floorplan recognition. In terms of methodology, WallPlan strictly adheres to the breadth-first traversal to generate wall graphs, whereas Raster-to-Graph employs a non-fixed order, a more general graph traversal. This flexibility and robustness in graph traversal make our iterative recognition process more adaptable. For the task of floorplan recognition, ensuring that each step in the recognition process accurately corresponds to every step in a strict graph traversal is highly challenging. This can potentially lead to unexpected interruptions in the inference process. Concerning the model structure, WallPlan employs a semantic segmentation network to predict pixel-level wall junctions, wall segments, and rooms, necessitating post-processing for extracting vectorized results. In contrast, Raster-to-Graph utilizes a fully connected network to predict wall junctions and wall segments, obtaining vectorized results.

3. Overview

The framework of Raster-to-Graph for floorplan recognition is illustrated in Figure 3. In the following, we discuss the floorplan representation and recognition challenges of floorplan recognition, and then present our framework for floorplan recognition.



Representation. We have borrowed the floorplan representation from [SWL*22], where floorplans can be represented using structural graphs with semantics, as shown in the right inset. Specifically, the structural graphs consist of nodes and edges: the wall junctions are abstractly represented as nodes of the graph, while the wall segments connecting wall junctions are represented as edges. The nodes and edges partition the indoor space into several non-overlapping polygonal regions, forming rooms of the floorplan. As a result, the structural graph with semantics is represented as a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_0, v_1, ..., v_n\}$ is the set of nodes, and $\mathcal{E} = \{(v_i, v_j) \mid v_i, v_j \in \mathcal{V}\}$ is the set of edges. Each node is defined by multiple attributes, denoted as $v_i = (\mathcal{C}_i, t_i, \mathcal{R}_i)$. $\mathcal{C}_i = (x_i, y_i)$ represents the positional coordinate of v_i. The node type is denoted as t_i, which falls into one of the 13 types proposed in [LWKF17], where node types associated with wall junctions are categorized as *I-shaped*, *L-shaped*, *T-shaped*, and *X-shaped*, and further subdivided into 13 types based on different orientations. Additionally, $\mathcal{R}_i = (r_i^{tl}, r_i^{tr}, r_i^{bl}, r_i^{br})$



(a1) (a2) (a3)

Room type	Living-room	Bedroom	Kitchen	Bathroom	Restroom	Closet	Balcony	Washing-room	Corridor	Pipe-space	Unknown
#Floorplan	2090	15821	10187	10895	9062	21662	9283	5061	4921	1990	557

(b)

(c1) (c2) (c3)

Figure 2: *Overview of Dataset. (a1)-(a3): Statistics on the occurrence of wall junctions (a1), wall segments (a2), and rooms (a3). (b): Statistics on the occurrence of each room category. (c1)-(c3): A typical example in our dataset: rasterized floorplan image (c1), the corresponding structural graph (c2) and semantic floorplan (c3).*

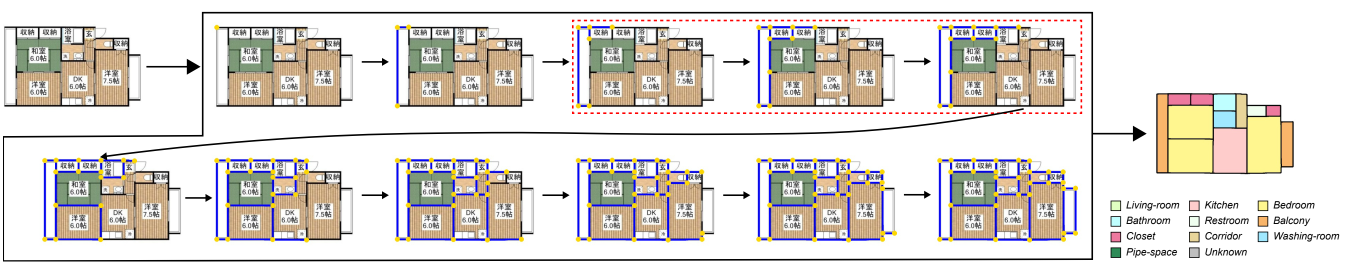


Figure 3: *Overview of Raster-to-Graph. We represent the vector floorplan as a structural graph with semantics, and then the task of floorplan recognition can be viewed as structural graph prediction. Our method takes the rasterized floorplan as input, iteratively predicts nodes (shown in yellow) and edges (shown in blue) of the structural graph in the order of graph traversal, and finally obtains a complete structural graph. A vectorized floorplan can be easily extracted using the structural graph. It's worth noting that during the iterative process, our framework does not strictly adhere to the fixed order of graph traversal, as indicated in red. This is due to our training strategy, which allows a non-fixed order, a more general graph traversal, enhancing the flexibility of our prediction process.*

records the room categories in the four directions: *top-left*, *top-right*, *bottom-left*, and *bottom-right* of v_i. Taking the node v_k in the right inset as an example, the node type of v_k is a variant of the *L-shaped* shown in green, and the room categories in the four directions of v_k are *Kitchen*, *Bedroom*, *Bedroom* and *Bedroom*, in clockwise order. Let $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_N\}$ denote the set of polygonal loops for each room. Each loop $\mathcal{P}_i$ is defined by a sequence of nodes: $\mathcal{P}_i = \{v_{i,1}, v_{i,2}, ..., v_{i,N_i} \mid v_{i,j} \in \mathcal{V}\}$. The room category of $\mathcal{P}_i$ is the room category common to all nodes constructing this room.

Dataset. The task of floorplan recognition is constrained by the scale of the dataset and the quality of data annotations. To address the challenges posed by training data, we have leveraged an available, massively scaled realistic floorplan dataset, LIFULL HOME'S Dataset [LC16], to construct an extensive floorplan recognition dataset. Our dataset comprises 10,804 high-quality floorplans, each with sufficient structural and semantic annotations. Creating such a large-scale dataset from scratch naturally requires a significant amount of manpower. To simplify the process, we first extract an amount of 60,000 floorplans from LIFULL HOME'S Dataset and apply Raster-to-Vector [LWKF17] to generate initial annotation information. To standardize the representation, we preprocess the original floorplans into a resolution of 512 × 512 and centered, resulting in three-channel images. Finally, we extract the graph structures from initial annotations, eliminating: (i) samples with disconnected graphs; (ii) samples that are either too simple or too complex (with the number of graph nodes limited to between 10 and 50); (iii) samples containing isolated graph nodes. After this, we can obtain approximately 30,000 annotated data. Following this, we engage human annotators to perform manual checks and coordinate corrections on these data. Through this meticulous process, we finally can obtain over 10,000 refined annotations. An overview of our floorplan dataset is shown in Figure 2. Rooms were categorized into 11 different categories: *Living-room*, *Kitchen*, *Bedroom*, *Bathroom*, *Restroom*, *Balcony*, *Closet*, *Corridor*, *Washing-room*, *Pipe-space*, and *Outside*. For regions with unidentified categories, we uniformly label them as *Unknown*.

Problem & challenges. Given a rasterized floorplan $\mathcal{I} \in [0,255]^{3\times H\times W}$ (with 3 color channels, H, W representing the resolution of this floorplan image, here we set $H = W = 512$), our goal is to provide the vectorized representation of the floorplan: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{G}$ represents a structured graph with semantics, $\mathcal{V}$ denotes the set of nodes (wall junctions), and $\mathcal{E}$ is the set of edges (wall segments). This problem poses two significant challenges. First, creating vectorized floorplans that meet the requirements of practical applications necessitates incorporating vector information such as wall junctions, wall segments, and room polygons. Given the complexity of vector data structures, designing a highly individualized deep learning network to achieve vectorization of floorplans becomes a crucial challenge. Second, the interdependencies among wall junctions, wall segments, and room polygons involve geometric and topological relationships. Independently recognizing the structure and semantics of floorplans can not effectively utilize these relationships. Therefore, an integrated approach that leverages these relationships is superior to designing separate extraction modules. However, devising an integrated model that exploits the correlations between the structures and semantics of floorplans poses another challenge.

Methodology. Inspired by [SWL*22], we represent vectorized floorplans as structural graphs with semantics. This reformulates the traditional floorplan recognition problem into seeking a mapping from a rasterized floorplan to a structural graph, denoted as $\mathcal{M} : \mathcal{I} \longmapsto \mathcal{G}$. Obtaining such a global mapping assumes that floorplan elements are independent of each other, without considering any sequential relationships among them. Unlike existing methods that directly acquire a global mapping from rasterized floorplans to vectorized representations, we decompose the global mapping into a series of local mappings. In this way, we model the problem of recognizing $\mathcal{G}$ as an autoregressive subgraph generation process. Given an input floorplan $\mathcal{I}$, our proposed autoregressive model Raster-to-Graph sequentially traverses $\mathcal{I}$, recognizing new wall junctions and wall segments, eventually forming a complete, closed, and connected structural graph $\mathcal{G}$. The process of recognizing the structural graph $\mathcal{G}$ from the floorplan $\mathcal{I}$ is an iterative process, with each iteration generating a new subgraph based on the previously generated subgraph. Let $\mathcal{G}_{t-1}$ represent the subgraph recognized in the $(t-1)$-th iteration, with its corresponding floorplan mask on $\mathcal{I}$ for this subgraph be denoted as $\mathcal{I}_{t-1}$. Raster-to-Graph can obtain $\mathcal{G}_t$ in the t-th iteration by determining the new set of nodes and edges that are newly added to $\mathcal{G}_{t-1}$.

In summary, the mapping from the floorplan $\mathcal{I}$ to the structural graph $\mathcal{G}$ is achieved through an autogressive process:

$$\mathcal{M} : \mathcal{I} \longmapsto \mathcal{G} \Longleftrightarrow \mathcal{M} : (\mathcal{I}, \mathcal{G}_{t-1}) \longmapsto \mathcal{G}_t \tag{1}$$

where for each step $t \in 1, 2, ..., T$,

$$\mathcal{G}_t = \mathcal{M}(\mathcal{I}, \mathcal{G}_{t-1}) = \text{R2G}(\mathcal{I}, \mathcal{I}_{t-1}) \tag{2}$$

Here $\mathcal{I}_{t-1} = \mathcal{I}(\mathcal{G}_{t-1})$ represents the floorplan mask corresponding to the subgraph $\mathcal{G}_{t-1}$ on $\mathcal{I}$. Specifically, each iteration involves node prediction and graph construction. The floorplan $\mathcal{I}$, along with the corresponding floorplan mask $\mathcal{I}_{t-1}$, serve as inputs to node prediction, which outputs a set of new nodes, denoted as $\Delta\mathcal{V}_t$. Then, the newly generated $\Delta\mathcal{V}_t$ is added to the old subgraph $\mathcal{G}_{t-1}$ to construct a new subgraph $\mathcal{G}_t$. This process is repeated until the complete structural graph $\mathcal{G}_T$ is obtained. We further extract all rooms to obtain the final vectorized result.

4. Method

We aim to transform rasterized floorplans into structural graphs with semantics, thereby achieving the vectorization of floorplans. To this end, we propose a novel floorplan vectorization framework Raster-to-Graph for recognizing the semantics and structures of rasterized floorplans. As shown in Figure 3, Raster-to-Graph takes as input a rasterized floorplan $\mathcal{I}$ and outputs a vectorized result $\mathcal{G}$ as output. The whole recognition process functions as an iteration, with each iteration consisting of two key steps: (i) Node prediction, which predicts nodes with multiple attributes, including node coordinates, node types, and room categories, and (ii) Graph construction, which automatically constructs structural graphs by leveraging the rich attributes associated with predicted nodes.

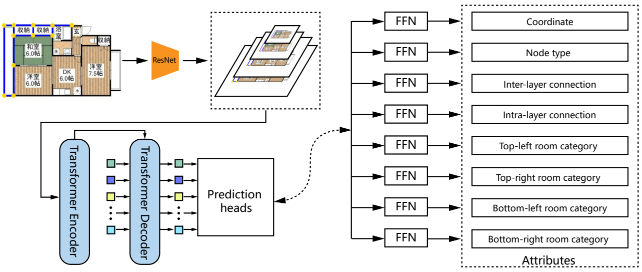


Figure 4: *The network architecture of our node prediction model consists of two main components: CNN backbone and Transformer. ResNet as the CNN backbone is employed to extract multi-scale features from the subgraph mask. The Transformer receives image features and a fixed number of learnable node queries as inputs and outputs the same number of embeddings, which aggregates information from both the node queries and the image features to capture long-range dependencies among nodes. The output embeddings can be used for node attribute prediction by passing to the shared FFNs. There are 8 shared FFNs. Each branched FFN is responsible for predicting various attributes of nodes (coordinates, node types, and room types) or the topological relationships between nodes (inter-layer connections, and intra-layer connections).*

4.1. Node predication

We approach the problem of predicting nodes in structural graphs (i.e., wall junctions in floorplans) as a keypoint detection task and design our node prediction model with reference to the Deformable DETR [ZSL*20]. As previously mentioned, we do not perform an all-at-once detection of all nodes on the floorplan. Instead, we predict them iteratively, step by step. In each iteration, the input to node prediction model consists of the original floorplan image $\mathcal{I}$ and the subgraph $\mathcal{G}_{t-1}$ obtained from the previous iteration. For the convenience of encoding representation, we further merge $\mathcal{I}$ and $\mathcal{G}_{t-1}$, constructing a subgraph mask $\mathcal{I}_{t-1}$ as the actual input to the node prediction model. Specifically, we visualize $\mathcal{G}_{t-1}$ on $\mathcal{I}$. We draw a 9×9 yellow square centered on the wall junction for each node in $\mathcal{G}_{t-1}$. For edges in $\mathcal{G}_{t-1}$, we draw blue line segments with a pixel width of 2, as shown in Figure 4.

Given a subgraph mask, we first employ a convolutional neural network as the backbone to extract image features. These features are then fed into the Transformer encoder, which utilizes multiple layers of self-attention mechanisms and feedforward neural networks to capture contextual information within the input features, enabling a deep understanding of the input. The Transformer decoder then takes as input a fixed number of learnable positional embeddings, which we call node queries, and the output feature embedding passed by the Transformer encoder. It uses self-attention mechanisms to focus on the already generated subgraph, thereby generating node embeddings for the next iteration. This process is iterative, allowing the model to progressively predict nodes to adapt to the complexity of the task. This autoregressive prediction is highly valuable in handling graphical data and enables the model to capture the sequential relationships within the input data while efficiently addressing long-range dependencies.

CNN backbone. Given a rasterized floorplan image $\mathcal{I}$, we utilize a pre-trained ResNet-50 [HZRS16] as the backbone to extract image features. Given that wall junctions are extremely small-scale objects, and predicting their coordinates demands better spatial generalization capabilities, we need to leverage low-level image features. On the other hand, we also require high-level image features for the semantic prediction of graph nodes. To the end, we employ multi-scale feature maps from the CNN backbone to construct a four-level image feature pyramid and apply a 1×1 convolution to reduce the channel dimension of the high-level feature maps to C: $f^0_{pyramid} \in \mathbb{R}^{C\times\frac{H}{4}\times\frac{W}{4}}$, $f^1_{pyramid} \in \mathbb{R}^{C\times\frac{H}{8}\times\frac{W}{8}}$, $f^2_{pyramid} \in \mathbb{R}^{C\times\frac{H}{16}\times\frac{W}{16}}$, $f^3_{pyramid} \in \mathbb{R}^{C\times\frac{H}{32}\times\frac{W}{32}}$. The typical value of C we use is $C = 256$. This approach aims to consider information at different scales when dealing with floorplans, thereby better catering to both the precise coordinate prediction and semantic understanding of the nodes within the floorplan.

Transformer encoder. In the image feature pyramid, the feature position directly influences the prediction of node coordinates. To represent the feature position, we establish spatial position encoding $f^i_{position} = [\gamma(x), \gamma(y)] \in \mathbb{R}^C, i = 0,1,2,3$ for the image feature $f^i_{pyramid}$, where x and y denote the coordinates of the features, and $\gamma(t)$ is defined as $\gamma(t) = [sin(w_0 t), cos(w_0 t), \ldots, sin(w_{63} t), cos(w_{63} t)]$, with $w_i = (\frac{1}{10000})^{\frac{2i}{128}}$ for $i = 0,1,\ldots,63$ as described in [VSP*17]. Furthermore, to represent the feature scale, we establish feature scale encoding $f^i_{scale} \in \mathbb{R}^C, i = 0,1,2,3$ for the image feature $f^i_{pyramid}$. f^i_{scale} is a randomly initialized and learnable vector. Then, for each feature scale, we combine the image feature with their corresponding spatial position encoding and feature scale encoding to obtain a fused feature $f^i_{fuse} = f^i_{pyramid} + f^i_{position} + f^i_{scale}, i = 0,1,2,3$. At last, we fuse the above features of all four scales of the image feature pyramid to obtain the final fuse feature $f_{fuse} = \sum_{i=0}^{3} f^i_{fuse}$. The spatial dimensions of the fused image feature pyramid f_{fuse} is collapsed into one dimension, resulting in a $C \times (\frac{H}{4} \times \frac{W}{4} + \frac{H}{8} \times \frac{W}{8} + \frac{H}{16} \times \frac{W}{16} + \frac{H}{32} \times \frac{W}{32})$ feature map, serving as the input to the Transformer encoder. It aims to capture both position and scale information of image features, enhancing the prediction of node coordinates. The encoder outputs N embeddings of size C, denoted as $O_{encoder}$, which contain node coordinates, associated wall segments, and semantic information. They will be further processed by the Transformer decoder.

Transformer decoder. The input set of the Transformer decoder is $O_{encoder} \cup \{q_j\}_{j=1}^N$, where $O_{encoder}$ is the output embedding of the Transformer encoder and $q_j \in \mathbb{R}^C$ is a learnable query vector. $\{q_j\}_{j=1}^N$ participate in training and learn the priors of nodes. In the inference, $\{q_j\}_{j=1}^N$ is used as the decoder's input and provides priors for prediction. The output embeddings of the Transformer decoder can be used for node attribute prediction by passing to the prediction heads consisting of shared feed forward networks (FFNs). Each branched FFN is responsible for predicting various attributes of graph nodes. The output of the Transformer decoder is a set of new nodes $\Delta\mathcal{V}_t$ consisting of multiple attributes as below:

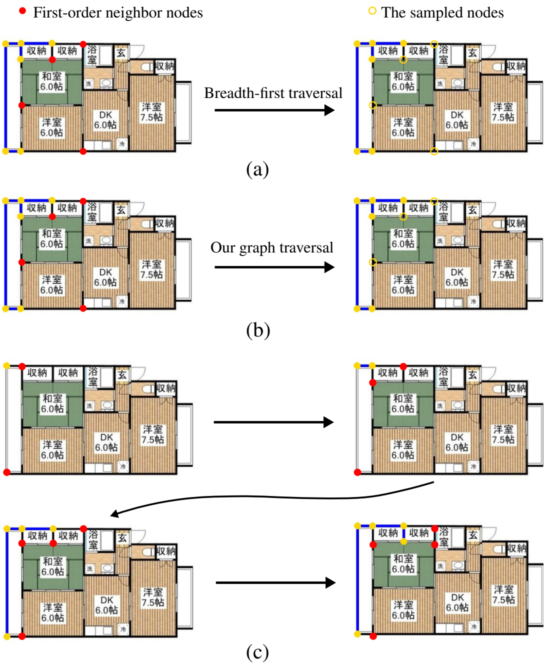
- *Positional coordinate*: $\mathcal{C}_i = (x_i, y_i)$ where $x_i \in (0,1)$ and $y_i \in (0,1)$. We use an FFN to predict the normalized central coordinates of the node, and this is a regression task.
- *Node type*: t_i. Since we predict a fixed-size set of N nodes, where N is usually larger than the actual number of nodes, an additional special class label $\varnothing$ is used to represent that no node is detected. In addition, we have added an "End" type as a termination condition for the prediction. Therefore, for node type, there are in total 15 (13+1+1) different types. We use an FFN for the node type classification.
- *Room category*: $\mathcal{R}_i = (r^{tl}_i, r^{tr}_i, r^{bl}_i, r^{br}_i)$. For each of the four directions, we have in total 11 different categories. Similar to node type classification, we use an FFN for room category classification in each of the four directions.

Only relying on the above node attributes is insufficient for achieving the iterative prediction of structural graphs. This is because the topological relationships between nodes cannot be determined. Specifically, the relationships between the subgraph from the previous level and the nodes predicted at the current level cannot be established. These topological relationships include (i) inter-level connections: connections between the nodes predicted at the current level and the subgraph from the previous level, and (ii) intra-level connections: connections between the nodes predicted at the current level. Therefore, we separately predict the inter-level and intra-level connections, As shown in Figure 6:

- *Inter-level connection*: $Inter_i \in \{0,1\}^4$ respectively records the presence of connections in the four directions of v_i: *up*, *down*, *left*, and *right*, where 1 indicates the presence of a connection, and 0 indicates the absence. For each node, there are possible connections in four directions, resulting in a total of 16 inter-level connection types. We employ an FFN for inter-level connection classification.
- *Intra-level connection*: $Intra_i \in \{0,1\}^4$, defined in the same way as inter-level connections. Similar to inter-level connections, there are 16 connection types. We use another FFN for intra-level connection classification.

The output of the node prediction is a new node set $\Delta\mathcal{V}_t$ with predicted attributes. This approach allows us to iteratively predict nodes in a way that takes into account both the floorplan image and previously predicted subgraph, enhancing the prediction accuracy.

Training Strategy. Our goal is to enable the neural network to predict first-order neighbor nodes $\Delta\mathcal{V}_t$ of the subgraph $\mathcal{G}_{t-1}$ given the floorplan image $\mathcal{I}$ and the subgraph $\mathcal{G}_{t-1}$. To obtain $\mathcal{G}_{t-1}$ and $\Delta\mathcal{V}_t$, one direct approach is to use the sampling method in WallPlan [SWL*22] where each step of the structural graph traversal is used as training samples. However, this prediction process requires precisely matching every step of the graph traversal, otherwise, the iteration process terminates. Due to the uncertainty of



(a)

(b)

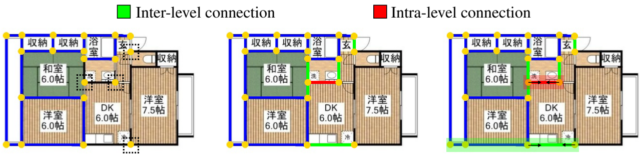
(c)

Figure 5: *An illustration of different training strategies of structural graph traversal. (a) Breadth-first traversal where all first-order neighbor nodes serve as the sampled nodes. (b) Our training strategy of more general graph traversal where random samples of first-order neighbor serve as the sampled nodes. (c) A constructing process of training sample.*

neural networks, ensuring that each recognition step accurately corresponds to every step in the graph traversal is highly challenging. This can potentially lead to unexpected interruptions during the inference process, such as the disconnection of the subgraph from the newly predicted nodes.

To avoid this problem and enhance the flexibility of our model, we propose to use a more general sampling method to obtain training data, randomly visiting nodes in the first-order neighbor with a certain probability during graph traversal, i.e., probabilistic sampling. The idea is to allow a certain degree of error during the iterations and not to strictly adhere to the fixed order of graph traversal (either depth-first or breadth-first), instead of a non-fixed order, a more general graph traversal. The training data is constructed as follows: Let the number of graph nodes be n, and the node number of $\mathcal{G}_{t-1}$ be n_{t-1}. We uniformly sample n_{t-1} from $[0,n]$:

- *If* $n_{t-1} = 0$: The constructed subgraph is an empty graph.
- *If* $0 < n_{t-1} < n$: Without loss of generality, we first choose the top-left node as the starting node of graph traversal. Then, we randomly sample $n_{t-1} - 1$ nodes in the order of breadth-first traversal with a sampling probability of $p = 0.5$. To obtain n_{t-1} nodes, starting from the top-left node, we randomly visit nodes in the first-order neighbor of sampled nodes with $p = 0.5$ and mark the visited nodes as sampled. The process can sample all n_{t-1} nodes in a finite iterative time. Figure 5(c) shows a training sample constructing process. We set the sampling probability $p = 0.5$, and compare different p in Table 1.



(a)

(b)

(c)

Figure 6: *An illustration of our graph construction. (a) The subgraph and the predicted nodes (in the dotted box) for graph construction. (b) Constructed graph. The inter-level connections between the subgraph and the predicted nodes are shown in green, and the intra-level connections between the predicted nodes are shown in red. (c) Line search for different kinds of connections. The bandwidth indicates the threshold we set in line search.*

- *If* $n_{t-1} = n$: The constructed subgraph is the final graph $\mathcal{G}$.

The new node set $\Delta\mathcal{V}_t$ to be predicted for the next level are all the first-order neighbor nodes of the subgraph $\mathcal{G}_{t-1}$. If $\mathcal{G}_{t-1} = \varnothing$, $\Delta\mathcal{V}_t$ only contains the starting node. If $\mathcal{G}_{t-1} = \mathcal{G}$, $\Delta\mathcal{V}_t$ only contains an "End" symbol as a termination of prediction.

To improve the accuracy and robustness of coordinate prediction, we apply a perturbation to the input subgraph $\mathcal{G}_{t-1}$ during the training process, by adding Gaussian noise $\mathcal{N}(\mu,\sigma)$ to the node coordinates. By adding noise to input data, the model becomes more adept at handling imperfect or noisy data. This enhances the stability of the model when confronted with noise or variations, reducing the risk of overfitting. On the other hand, noise can simulate data uncertainty, introduce data diversity, and aid the model in generalizing better to new data with a certain degree of noise.

4.2. Graph construction

Node prediction can only predict a set of new nodes, we have to connect the newly predicted nodes to the previous subgraph, thus constructing the new subgraph. The input for graph construction consists of the subgraph $\mathcal{G}_{t-1}$ of the previous level and the newly predicted nodes $\Delta\mathcal{V}_t$ of the current level. Graph construction utilizes the attributes of the newly predicted nodes $\Delta\mathcal{V}_t$ to establish connections between $\Delta\mathcal{V}_t$ and $\mathcal{G}_{t-1}$, thereby generating the new subgraph $\mathcal{G}_t$, as shown in Figure 6.

For each $v_m \in \Delta\mathcal{V}_t$, we first acquire the inter-level connections of v_m. We sequentially traverse in the four directions: *up*, *down*, *left*, and *right*. If there exists a connection in a certain direction, we perform a line search to find the node $v_n \in \mathcal{G}_t$ that is closest to v_m and set (v_m, v_n) to be an inter-level connection for v_m. For intra-level connections of v_m, we perform a bi-directional line search on both two predicted nodes. If the connection of two new nodes matches, then we set the intra-level connection, as shown in Figure 6(c). For example, v_m has a right intra-level connection and v_l has a left intra-level connection), then we select (v_m, v_l) to be an intra-level connection for v_i. In our implementation, due to the existence of errors in node coordinate prediction, a strict line search may fail to find connecting nodes. Therefore, we allow for a certain threshold range (we set the threshold to be 5 pixels) in specific directions for searching.

Given the rasterized floorplan image $\mathcal{I}$, Raster-to-Graph predicts new subgraphs in an autoregressive manner. Specifically, in each iteration, Raster-to-Graph performs node prediction and graph construction. It begins by predicting the nodes of the current level based on the subgraph of the previous level and then adds the newly predicted nodes to the subgraph of the previous level through graph construction. The autoregressive process stops when there is only one newly predicted node with a node type of "End", resulting in the complete structural graph $\mathcal{G}$.

Room extraction. With the structural graph $\mathcal{G}$, we can easily extract all rooms, including the room polygons and room semantics. By extracting all the shortest cycles in the structural graph $\mathcal{G}$, we obtain all room polygons. To determine room semantics, we extract the attribute of the room category of all nodes that construct the room polygons. We choose the most frequently occurring room category among all nodes sharing room categories in the direction of the room polygon as the final room semantics, considering that the room category prediction may have errors, and not all nodes will share the same room category.

5. Experiment and evaluation

5.1. Implementation

The design of our model follows the architecture of Deformable DETR [ZSL*20], tailored for our object detection task. We have made the following adjustments to the hyperparameters. We have increased the number of key sampling points of deformable attention from 4 to 20. The deformable encoder has 1 layer, while the decoder has 6 layers. We set the number of node queries to $N = 500$ while keeping other settings unchanged.

For FFNs in prediction heads, the coordinate regression adopts a 4-layer FFN, the intra-level classification uses a 2-layer FFN, and other tasks have 1 layer. We have trained the model for 300 epochs on an NVIDIA GeForce RTX 4090, with a batch size of 16. We used the Adam optimizer [KB14] with an initial learning rate of 2e-4, except for the CNN backbone and linear projection layers, which have a learning rate of 2e-5. Weight decay is set to 1e-5, and the learning rate is decayed by a factor of 0.1 at the 80-th epoch. Raster-to-Graph is trained on our proposed dataset, with 500 floorplans for validation, 500 floorplans for testing and the rest for training.

The loss settings are the same as Deformable DETR. Coordinate regression uses L1 loss, and other tasks use Focal loss with $\alpha = 0.25, \gamma = 2$. We train these branches jointly, and the final loss is a weighted summation of branch losses. For the weights of branch losses, we set 20 for coordinate regression, 2 for node type classification, inter-level classification, and intra-level classification, and 0.5 for all four room category losses.

5.2. Evaluation metric

To evaluate the performance of our floorplan recognition method, we have established multiple evaluation metrics, primarily focusing on the recognition of wall junctions, wall segments, regions (room polygons), and rooms (regions with room categories).

- **Wall Junction**: We calculate the Chebyshev distance [Cog16] between the predicted wall junction and the ground truth wall junction. If the distance is less than a given threshold, the wall junction prediction is considered true. The given threshold we set in experiments is 5 pixels.
- **Wall Segment**: We calculate the Chebyshev distances between the predicted wall junctions and the wall junctions of ground truth on both sides of the predicted wall segment. The larger distance is taken as the distance for the predicted wall segment. If the distance is less than a given threshold, the wall segment prediction is considered true. The given threshold we set is 5 pixels.
- **Region**: We use the Intersection over Union (IoU) to measure the error between the predicted regions and the ground truth regions. If the IoU is larger than a given threshold, the region prediction is considered true. The given threshold we set is 0.7.
- **Room**: A room prediction is considered true only when the corresponding region and room category predictions are both considered true.
- **Structure**: When all wall junctions and wall segments of a floorplan are considered true, which means the floorplan structure is perfectly recognized, then the structure prediction is considered true. We calculate the percentage of samples whose structure is true in the testing dataset.
- **Overall**: The overall prediction is considered true when the structure and semantic predictions are both considered true, which measures the overall floorplan recognition. We calculate the percentage of samples whose overall is true in the testing dataset.

5.3. Ablation study

We propose an autoregressive approach for floorplan recognition. To demonstrate the effectiveness of the manner of autoregressive, we have conducted an ablation experiment with the non-autoregression method. We also have explored the impact of different training strategies.

Autoregression/non-autoregression. To obtain the non-autoregression version of our method, we predict all wall junctions and wall segments in one step. We have removed two branches: inter-level connections and intra-level connections, from the node attribute prediction. We denote the non-autoregression method as R2G-NA. Figure 7(b) and (d) show the comparison between our autoregressive and the non-autoregressive method. From (b), it is evident that recognition results in Rows 2, 3, and 5 have lost some details of floorplans. Moreover, the non-autoregressive model also struggles to handle complicated floorplans, such as the lower right in Row 1 and the upper left in Row 4. The non-autoregressive method also makes some errors in the main parts of floorplans, as shown in Row 1 where the largest room is predicted as *Living-room* instead of the correct room category *Bedroom*. In contrast, our autoregressive method performs exceptionally well on these complex floorplans. It is worth noting that our method still predicts the correct room category for the pipe space in Row 4, while the corresponding "ground truth" in the dataset gives the wrong label. The non-autoregressive model tends to generate more incomplete structures, showing a noticeably weaker recognition effect, primarily characterized by structural errors like missing or fabricated wall segments and incorrect semantic recognition.

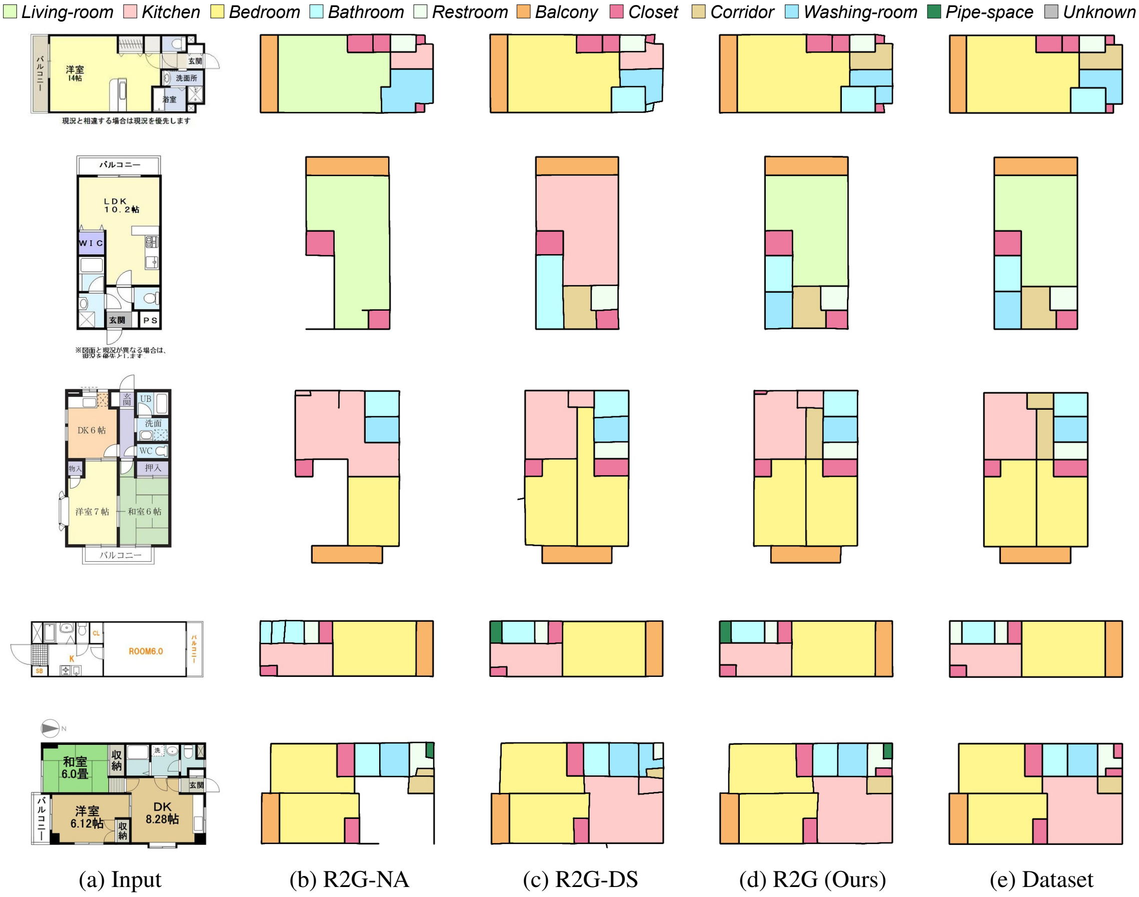


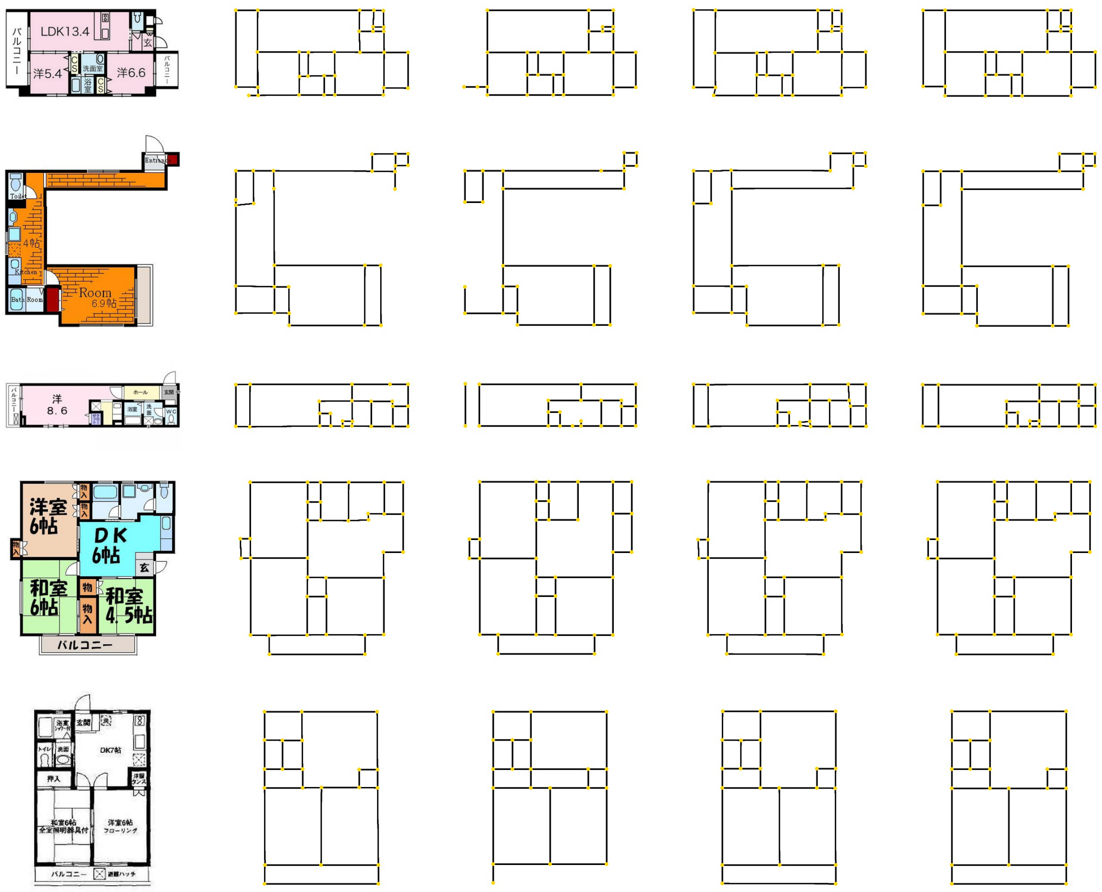
Figure 7: *Ablation study. From (b)-(d), only closed polygons are colored as rooms. In (e), we present the corresponding "ground truth" in our dataset. Note there might be a minor level of annotation errors in the semantic labeling of our dataset. For example, in Row 4, The Pipe-space has been incorrectly labeled as the Restroom.*

Table 1: *Ablation study. t(s): average inference time. R2G-Non-Autoregression: the ablation experiment of our non-autoregression model. R2G-Deterministic-Sampling: the ablation experiment of our deterministic-sampling. R2G (Ours): our probabilistic-sampling with probability p=0.5. The* ***bolded*** *is the best result. We have also reported the p-values of R2G-NA for the F-test on F-1 scores of R2G-NA and ours. P-value is a measure of whether the observed difference is statistically significant. The difference is significant if the p-value is less than 0.05. The reported p-values illustrate that our autoregressive method significantly outperforms the non-autoregressive method overall.*

Method	Wall Junction			Wall Segment			Region			Room			Structure	Overall	t(s)
	Prec	Recall	F-1	Prec	Recall	F-1	Prec	Recall	F-1	Prec	Recall	F-1			
R2G-NA	**98.9**	94.8	96.8 (0.082)	96.9	91.0	93.9 (0.002)	96.1	86.1	90.8 (0.000)	**86.0**	77.0	81.2 (0.000)	53.8	25.6	0.13
R2G-DS	98.3	95.9	97.1	96.2	92.8	94.5	96.2	90.3	93.2	84.5	79.3	81.8	60.0	23.6	0.30
R2G (p=0.25)	98.6	**97.5**	**98.0**	96.8	**95.4**	**96.1**	**96.7**	94.3	95.4	85.9	83.7	**84.8**	66.0	28.2	0.30
R2G (p=0.75)	98.7	97.3	**98.0**	**97.1**	95.2	**96.1**	96.6	93.7	95.2	85.6	83.0	84.3	66.4	29.2	0.30
R2G (Ours)	98.7	97.4	**98.0**	96.9	**95.4**	**96.1**	96.6	**94.6**	**95.6**	85.6	**83.8**	84.7	**67.0**	**30.0**	0.30

In contrast, Our autoregressive model achieves better recognition results close to the ground truth, even on complex floorplans with varying styles such as complicated colors, textures, icons, texts, and intricate structures.

We have also computed several metrics for the quantitative comparison between our autoregressive method and the non-autoregressive method, as presented in Table 1. The table clearly demonstrates that our autoregressive method significantly outperforms the non-autoregressive method on almost all metrics. We attribute this to two factors. On the one hand, the non-autoregression method assumes that the probability distribution of each wall junction is independent, while the autoregressive method models the sequential relationships on the floorplan as conditional probability



(a) Input floorplan (b) HEAT (c) R2V (d) R2G (Ours) (e) Dataset

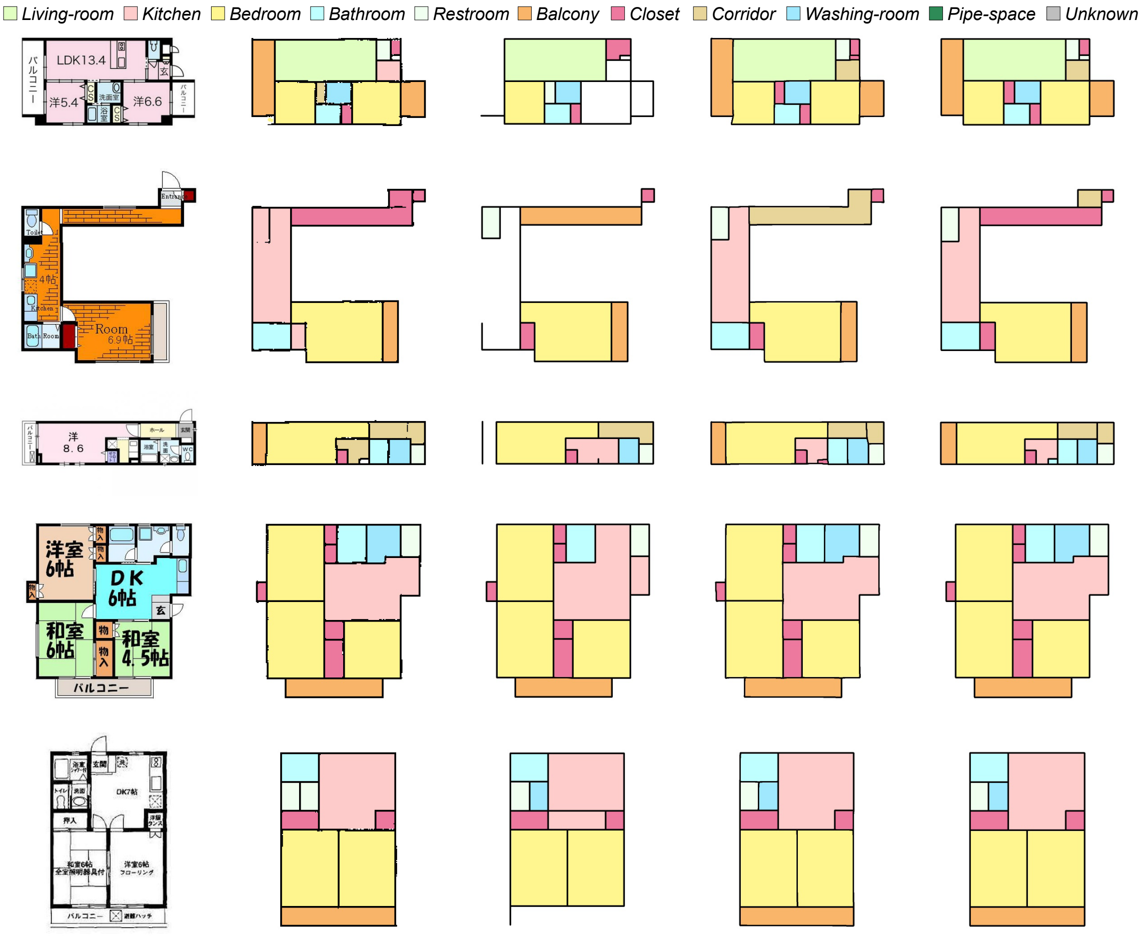
Figure 8: *Qualitative evaluation on structural reconstruction. From (b)-(e), all columns are shown in vectorized floorplans. In (e), we present the corresponding "ground truth" in our dataset.*

Table 2: *Quantitative evaluation on structural reconstruction. t(s): average inference time. R2G-S: our structure-only model. The color cyan and magenda mark the top-two results. For Room and Overall, we only mark the best result. Void cells: HEAT and R2G-S do not contain semantic recognition.*

Method	Wall Junction			Wall Segment			Region			Room			Structure	Overall	t(s)
	Prec	Recall	F-1	Prec	Recall	F-1	Prec	Recall	F-1	Prec	Recall	F-1			
R2V	94.4	96.7	95.5	95.6	92.6	94.1	86.3	72.6	78.9	76.4	64.2	69.8	14.8	7.0	4.11
HEAT	98.6	98.1	98.4	96.6	95.9	96.2	96.1	94.8	95.4	-	-	-	64.4	-	0.19
R2G-S	98.9	97.5	98.2	97.3	95.5	96.4	97.0	94.3	95.6	-	-	-	66.6	-	0.26
R2G (Ours)	98.7	97.4	98.0	96.9	95.4	96.1	96.6	94.6	95.6	85.6	83.8	84.7	67.0	30.0	0.30

distributions, which are more in line with reality. On the other hand, compared with the non-autoregressive method, our autoregressive method allows diverse data adoption during training, which greatly enriches the training data and improves the prediction accuracy and generalization of the model. Note that the non-autoregressive method has higher precision overall, our explanation is that R2G-NA attempts to capture the whole floorplan information and may prefer more conservative predictions, leading to a higher precision but much lower recall.

Probabilistic/deterministic sampling. To demonstrate the effectiveness of our probabilistic sampling, we have conducted an ablation experiment with deterministic sampling, denoted as R2G-DS. Deterministic sampling requires that all nodes in the first-order neighbor be visited during graph traversal to produce training data



(a) Input floorplan (b) DFPR (c) R2V (d) R2G (Ours) (e) Dataset

Figure 9: *Qualitative evaluation on semantic recognition. From (b)-(e), only closed polygons are colored as rooms. In (e), we present the corresponding "ground truth" in our dataset. Note there might be a minor level of annotation errors in the semantic labeling of our dataset.*

Table 3: *Quantitative evaluation on semantic recognition. Accu: for the wall, it is the ratio of the number of pixels correctly predicted as the wall to the total number of wall pixels. For the room, it is the ratio of the number of pixels predicted as the room to the total number of room pixels. The* ***bolded*** *is the best result.*

Method	Wall	Room	t(s)
	Accu	Accu	
R2V	77.2	73.1	4.11
DFPR	74.4	84.0	1.88
R2G (Ours)	**81.6**	**85.5**	0.30

such as subgraphs. Therefore, the generation process of the deterministic sampling method strictly follows the fixed order of the graph traversal. Figure 7(c) and (d) present the qualitative comparison between two sampling strategies. From (c), it can be observed that the deterministic sampling method still produces errors in the recognition of complex floorplans. In Row 1, the room structure is not correctly predicted. There are also some errors in other results, such as the semantic error in Row 2, as well as additional hanging wall segments outside the floorplan in Rows 3 and 5. The results indicate that our probabilistic sampling method outperforms the deterministic sampling model. The reconstruction quality of the deterministic sampling method is inferior to our probabilistic sampling method, leading to errors such as missing, overlapping, and fabricated wall segments, as well as incorrect semantics. In contrast, the reconstruction results of our probabilistic sampling method show higher quality. This aligns with our expectations. The deterministic sampling method struggles to ensure that every step of the prediction process precisely matches the graph traversal, which can potentially lead to unexpected interruptions in the inference process.

The corresponding quantitative comparison is shown in Table 1. In almost all metrics, our probabilistic sampling method outperforms the deterministic sampling method. The results further prove that our probabilistic sampling method exhibits better performance than the deterministic sampling method. We attribute this to the deterministic sampling not considering potential omissions during the inference process, leading to weaker performance.

In Table 1, we also compare different sampling probabilities. We choose probabilities $p = 0.25$ and 0.75 to compare with $p = 0.5$, and their metrics are similar. Our explanation is that different probabilities could also generate similar training data. We choose $p = 0.5$ as our method because it performs better in metrics of structure and overall.

5.4. Comparison with Baselines

5.4.1. Qualitative evalution

To perform the qualitative evaluation, we have compared our proposed method with the following methods. Figure 8 and Figure 9 illustrate the results of qualitative evaluation on structural reconstruction and semantic recognition, respectively.

R2V [LWKF17]: A hybrid method of semantic recognition and structural reconstruction to vectorize floorplans relies on manually designed optimization and post-processing algorithms to reconstruct the floorplan elements recognized by neural networks. As shown in Figure 8(c) of structural reconstruction, R2V produces numerous structural errors. In Rows 1 and 5, the absence of the balcony leads to hanging wall segments. In Row 2, the overall reconstruction of the room fails. Similarly, Figure 9(c) of semantic recognition also shows some semantic errors of R2V, such as several rooms being mistakenly identified as the outdoor space in Row 1 and the corridor being incorrectly recognized as a balcony in Row 2. R2V displays various errors in both structural and semantic recognition, resulting in imperfect recognition. We attribute this to that the structural reconstruction of R2V relies on optimization with manually designed constraints, making it less robust to input noise compared to neural networks. Additionally, the semantic segmentation of R2V also produces some errors in recognizing wall junctions, leading to poorer wall semantic segmentation. Furthermore, as a non-autoregressive method, R2V assumes independence among wall junctions, making it more challenging to achieve high-quality floorplan recognition.

DFPR [ZLYF19]: A semantic segmentation method for floorplan recognition based on the spatial relationships of floorplan elements, which does not produce vectorized results. DFPR can provide pixel-level recognition results but still introduces some errors in room categories and certain structural elements, such as thicker wall segments or wall segments covered by door icons. As shown in Figure 9(b) of semantic recognition, DFPR introduces some semantic errors. In Row 1, the room labeled as *Closet* is incorrectly predicted as a corridor. Similarly, in Row 3, the central kitchen is recognized as a corridor. The wall segments of the floorplan are diverse and small targets, while some additional icons can interfere with the recognition of the walls, which leads to the confusion of image features after multiple downsampling of the semantic segmentation network of DFPR, resulting in poorer recognition.

HEAT [CQF22]: A structural reconstruction method based on a dual Transformer structure for inferring corners and edges on the rasterized floorplans, which does not provide semantic information for rooms. HEAT excels in recognizing floorplan details and overall geometric structures but still makes errors on some bizarre floorplans. As shown in Figure 8(b) of the structural reconstruction, HEAT demonstrates high-quality reconstruction performance. However, it makes mistakes on some floorplans, such as the pillar in the lower left of Row 1 and the corridor in Row 2.

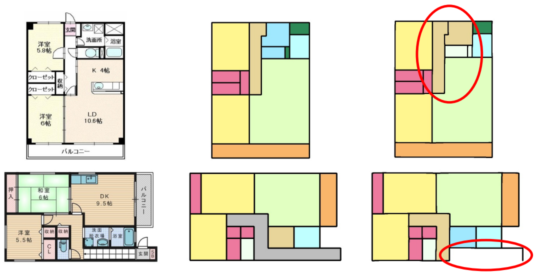
In contrast, the structural reconstruction of our method outperforms DFPR and HEAT. As shown in Figure 8(d), even when faced with complicated colors, textures, icons, and bizarre floorplans, our method consistently produces high-quality recognition results that are close to the ground truth. Our method exhibits exceptional recognition capabilities, maintaining geometric integrity while effectively handling stylistically diverse regions and prominent junctions within floorplans. The semantic recognition of our method is significantly better than DFPR and R2V, as demonstrated in Figure 9(d), our method still exhibits high-quality semantic recognition. It is worth noting that our method predicts the correct room category for the corridor in Row 2, while the corresponding "ground truth" in the dataset gives the wrong label.

5.4.2. Quantitative evalution

We have conducted a quantitative evaluation of R2V and HEAT. Furthermore, to fairly compare our model with HEAT in terms of structural reconstruction, we simplify R2G, reducing it to a pure structure recognition model, denoted as R2G-S, and retraining it in the same manner. In comparison with R2G, the changes in R2G-S are as follows: the number of key sampling points in deformable attention is reduced from 20 to 4, and the four FFNs used for semantic prediction are removed. The rest of the R2G model remains unchanged. Table 2 shows the results of the quantitative evaluation. From the table, it can be seen that our method significantly outperforms R2V. R2V performs significantly poorer than our method on all metrics except for junctions and walls, which can be attributed to that their method generates scattered connection points and walls, making it more challenging to form complete and correct regions. Furthermore, R2V employs manually designed optimizations and post-processing methods, making it much slower compared to our method.

Compared to HEAT, our pure structural model performs similarly on the metrics of wall junction, wall segment, and region, but excels in the metrics of structure and overall, which show better performance in the global structural reconstruction than HEAT. HEAT has similar results to ours in low-level element recognition, but HEAT's recognition error is more diffuse, resulting in poorer high-level structure recognition. Interestingly, the performance of our pure structural model is nearly on par with our complete model. Adding semantic tasks does not burden our model, indicating that even for more complex floorplan recognition tasks, our approach can achieve comprehensive information extraction in floorplans. This demonstrates the scalability of our method. In contrast, HEAT is specifically designed for structural recognition tasks and employs carefully designed edge and corner detection networks as well as edge classification networks. The complexity of the HEAT network makes it challenging to directly extend to floorplan recognition tasks.

To perform quantitative analysis and comparisons of the semantic recognition results, we rasterize the vectorized results of R2V



(a) (b) (c)

Figure 10: *Typical failure cases. (a) Input floorplan. (b) The ground truth. (c) Our prediction. The upper errs in some semantics, which may suffer from slight errors in our semantic annotation. The lower shows a rare structure, which our method performs poorly.*

and our algorithm to obtain semantic floorplans. In our experiments, we set the thickness of the wall to 5 pixels. DFPR can directly produce semantic recognition results. As shown in Table 3, even in pixel-level metrics, our method still outperforms DFPR and R2V in terms of the wall and room.

Limitations. Our method only recognizes axis-aligned walls. An improvement is to add wall angle prediction, which will be one of our further works. Limited to our current dataset, our method may fail to predict some structures on the rare floorplan images. The accuracy of semantics still needs to be improved, which may be due to slight errors in our semantic annotation. Figure 10 shows some typical failure cases.

6. Conclusion

We propose Raster-to-Graph, a novel automatic framework for floorplan recognition that effectively captures both structures and semantics. By representing vectorized floorplans as structural graphs with floorplan semantics, we have transformed the floorplan recognition task into a structural graph prediction problem. Our autoregressive model, based on the visual attention Transformer, iteratively predicts structures and semantics in the order of graph traversal. Additionally, we have introduced a large-scale floorplan dataset containing over 10,000 annotated real-world residential floorplans. Our extensive experiments and evaluations have demonstrated the effectiveness of the proposed recognition framework, showing significant improvements in the recognition of both structures and semantics compared to existing state-of-the-art methods. Raster-to-Graph not only offers a powerful solution for floorplan recognition, but also provides a valuable dataset and a robust autoregressive model for future research in floorplan analysis, architectural design, and other related applications.

Our work can play a potential role in the content creation of electronic games and virtual reality. We showcase some popup 3D models generated from our floorplan recognition results in Figure 11. Our dataset is not yet perfect, and its scale is not large enough. Some data annotations require further optimization and correction. On the other hand, our dataset only includes horizontal or vertical walls, which limits our method to recognizing only horizontal or vertical wall segments. Expanding to recognize slanted wall segments could be a potential future endeavor. Additionally, extending our approach to open rooms, and disconnected structures, and extracting information such as text and icons will also become interesting potential future work.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive suggestions and comments. This work is supported by the National Natural Science Foundation of China (62102126, 62372153) and the Fundamental Research Funds for the Central Universities of China (JZ2023HGTB0269, JZ2022HGQA0163). In this paper, we used "LIFULL HOME'S Dataset" provided by LIFULL Co., Ltd. via IDR Dataset Service of National Institute of Informatics.

References

[CLWF19] Chen J., Liu C., Wu J., Furukawa Y.: Floor-sp: Inverse cad for floorplans by sequential room-wise shortest path. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 2661–2670. 3

[CMS*20] Carion N., Massa F., Synnaeve G., Usunier N., Kirillov A., Zagoruyko S.: End-to-end object detection with transformers. In *European conference on computer vision* (2020), Springer, pp. 213–229. 3

[Cog16] Coghetto R.: Chebyshev distance. *Formalized Mathematics 24*, 2 (2016), 121–141. 8

[CQF22] Chen J., Qian Y., Furukawa Y.: Heat: Holistic edge attention transformer for structured reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 3866–3875. 2, 3, 12

[DWLZ21] Dong S., Wang W., Li W., Zou K.: Vectorization of floor plans based on edgegan. *Information 12*, 5 (2021), 206. 3

[DXS17] Dodge S., Xu J., Stenger B.: Parsing floor plan images. In *2017 Fifteenth IAPR international conference on machine vision applications (MVA)* (2017), IEEE, pp. 358–361. 2

[FLPH21] Fang H., Lafarge F., Pan C., Huang H.: Floorplan generation from 3d point clouds: A space partitioning approach. *ISPRS Journal of Photogrammetry and Remote Sensing 175* (2021), 44–55. 3

[FZL*21] Fan Z., Zhu L., Li H., Chen X., Zhu S., Tan P.: Floorplancad: A large-scale cad drawing dataset for panoptic symbol spotting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 10128–10137. 2

[HZRS16] He K., Zhang X., Ren S., Sun J.: Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778. 6

[JYY20] Jang H., Yu K., Yang J.: Indoor reconstruction from floorplan images with a deep learning approach. *ISPRS International Journal of Geo-Information 9*, 2 (2020), 65. 3

[KB14] Kingma D. P., Ba J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). 8

[KKY21] Kim H., Kim S., Yu K.: Automatic extraction of indoor spatial information from floor plan image: A patch-based deep learning methodology application on large-scale complex buildings. *ISPRS International Journal of Geo-Information 10*, 12 (2021), 828. 2

[KPKY21] Kim S., Park S., Kim H., Yu K.: Deep floor plan analysis for complicated drawings based on style transfer. *Journal of Computing in Civil Engineering 35*, 2 (2021), 04020066. 3

[KYH*19] Kalervo A., Ylioinas J., Häikiö M., Karhu A., Kannala J.: Cubicasa5k: A dataset and an improved multi-task model for floorplan image analysis. In *Image Analysis: 21st Scandinavian Conference, SCIA 2019, Norrköping, Sweden, June 11–13, 2019, Proceedings 21* (2019), Springer, pp. 28–40. 2

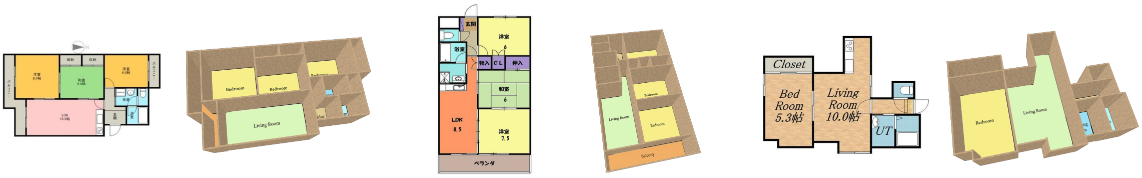


Figure 11: *Some popup 3D models from our floorplan recognition results. The popup 3D model is just generated by extruding the wall primitive to a certain height, similar to [LWKF17].*

[LC16] LIFULL Co. L. .: Lifull home's high resolution floor plan image data. Informatics Research Data Repository, National Institute of Informatics. (dataset). https://doi.org/10.32130/idr.6.2, 2016. 4

[LLC*20] Liu Y., Lai Y., Chen J., Liang L., Deng Q.: Scut-autoalp: A diverse benchmark dataset for automatic architectural layout parsing. *IEICE TRANSACTIONS on Information and Systems 103*, 12 (2020), 2725–2729. 2

[LSK*15] Liu C., Schwing A. G., Kundu K., Urtasun R., Fidler S.: Rent3d: Floor-plan priors for monocular layout estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 3413–3421. 2

[LWG*21] Lu Z., Wang T., Guo J., Meng W., Xiao J., Zhang W., Zhang X.: Data-driven floor plan understanding in rural residential buildings via deep recognition. *Information Sciences 567* (2021), 58–74. 2

[LWKF17] Liu C., Wu J., Kohli P., Furukawa Y.: Raster-to-vector: Revisiting floorplan transformation. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 2195–2203. 2, 3, 4, 12, 14

[LZYZ21] Lv X., Zhao S., Yu X., Zhao B.: Residential floor plan recognition and reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 16717–16726. 2, 3

[NF20] Nauata N., Furukawa Y.: Vectorizing world buildings: Planar graph reconstruction by primitive detection and relationship inference. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16* (2020), Springer, pp. 711–726. 2

[PK21] Park S., Kim H.: 3dplannet: generating 3d models from 2d floor plan images using ensemble methods. *Electronics 10*, 22 (2021), 2729. 3

[SRFL21] Stekovic S., Rad M., Fraundorfer F., Lepetit V.: Montefloor: Extending mcts for reconstructing accurate large-scale floor plans. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 16034–16043. 3

[SWL*22] Sun J., Wu W., Liu L., Min W., Zhang G., Zheng L.: Wallplan: synthesizing floorplans by learning to generate wall graphs. *ACM Transactions on Graphics (TOG) 41*, 4 (2022), 1–14. 2, 3, 5, 6

[SY21] Song J., Yu K.: Framework for indoor elements classification via inductive learning on floor plan graphs. *ISPRS International Journal of Geo-Information 10*, 2 (2021), 97. 2

[VSP*17] Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł., Polosukhin I.: Attention is all you need. *Advances in neural information processing systems 30* (2017). 3, 6

[WSC*21] Wu Y., Shang J., Chen P., Zlatanova S., Hu X., Zhou Z.: Indoor mapping and modeling by parsing floor plan images. *International Journal of Geographical Information Science 35*, 6 (2021), 1205–1231. 3

[XYA*21] Xu Z., Yang C., Alheejawi S., Jha N., Mehadi S., Mandal M.: Floor plan semantic segmentation using deep learning with boundary attention aggregated mechanism. In *Proceedings of the 2021 4th International Conference on Artificial Intelligence and Pattern Recognition* (2021), pp. 346–353. 2

[YKSE23] Yue Y., Kontogianni T., Schindler K., Engelmann F.: Connecting the dots: Floorplan reconstruction using two-level queries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2023), pp. 845–854. 2

[YWY21] Yamada M., Wang X., Yamasaki T.: Graph structure extraction from floor plan images and its application to similar property retrieval. In *2021 IEEE international conference on consumer electronics (ICCE)* (2021), IEEE, pp. 1–5. 3

[ZHZD20] Zhang Y., He Y., Zhu S., Di X.: The direction-aware, learnable, additive kernels and the adversarial network for deep floor plan recognition. *arXiv preprint arXiv:2001.11194* (2020). 2

[ZLYF19] Zeng Z., Li X., Yu Y. K., Fu C.-W.: Deep floor plan recognition using a multi-task network with room-boundary-guided attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 9096–9104. 2, 12

[ZNF20] Zhang F., Nauata N., Furukawa Y.: Conv-mpn: Convolutional message passing neural network for structured outdoor architecture reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 2798–2807. 2

[ZSL*20] Zhu X., Su W., Lu L., Li B., Wang X., Dai J.: Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159* (2020). 2, 3, 5, 8