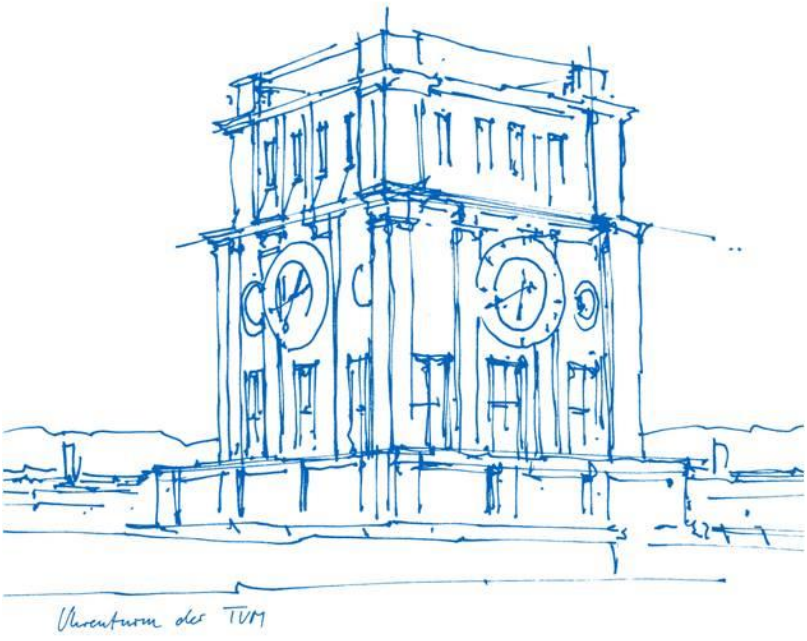
Technical University of Munich
TUM School of Engineering and Design

A Novel Method of Graph-Based Representation Learning for Floorplan CAD Drawings

Zhaohua Zheng



Technical University of Munich
TUM School of Engineering and Design

A Novel Method of Graph-Based Representation Learning for Floorplan CAD Drawings

Zhaohua Zheng

Complete reprint of the dissertation approved by the TUM School of Engineering and Design of the Technical University of Munich for the award of the

Doktor der Ingenieurwissenschaften (Dr.-Ing).

Chair: Prof. Dr. Pierluigi D'Acunto

Examiners:

1. Prof. Dr.-Ing Frank Petzold
2. Assoc. Prof. Dr. Chengyu Sun

The dissertation was submitted to the Technical University of Munich on 18 September 2023 and accepted by the TUM School of Engineering and Design on 13 December 2023.

A Novel Method of Graph-Based Representation Learning for Floorplan CAD Drawings

Abstract

The integration of information technology into the architecture, engineering, and construction (AEC) industries has revolutionized architectural processes with automation techniques. Computer-aided design (CAD) tools have brought geometric precision and efficiency, transforming architectural production. Tasks related to CAD drawings, such as conversion, reconstruction, detection, recognition, generation, and synthesis, pose challenges for researchers. These tasks often require significant labor and time resources, making manual solutions impractical, particularly for large-scale projects.

Data-driven methods, particularly deep learning, offer promise due to their robustness and adaptability. However, current methods lack a proper CAD drawing representation method and neural network architecture tailored to AEC tasks, which could fully embed the features in CAD drawing data. To tackling this problem, an approach is proposed based on line graphs and planar graphs with graph convolutional modules and is validated its effectiveness for floorplan detection and generation, which demonstrates superior performance compared to current methods.

This research contributes to advancing automation in architectural applications, providing a representation learning method as well as a general backbone structure for AI tasks in floorplan CAD drawings. This research centers on the advancement of floorplan CAD drawing representation through the application of neural networks, particularly in the domains of floorplan detection and generation. In this pioneering work, a novel graph-based representation methodology is introduced, along with a sophisticated backbone network structure for convolutions. These innovations are designed to significantly enhance the integration of essential geometric and topological features derived from real-world data.

The study's rigorous methodology includes a comprehensive series of experiments, meticulously designed to validate both the efficiency and precision of the proposed approach. The results underscore the remarkable performance and potential of the method, showcasing its robustness in various ablation scenarios. Furthermore, these findings highlight the far-reaching implications of this research, demonstrating its applicability to a broad spectrum of AEC applications beyond floorplan CAD.

Content

1 Introduction

1.1 Background

Ever since the integration of information technology in the architecture, engineering, and construction (AEC) industries, automation techniques have become a cornerstone of architectural applications for visualization, designing, simulation, and management. The implementation of computer-aided design (CAD) and computer-aided manufacture (CAM) have brought about unparalleled levels of geometric precision and speed, which have transformed the process of architectural production and construction. The use of CAD and CAM software has boosted efficiency and performance to new heights. In the realm of management, professional software has made planning and scheduling far more accurate, flexible, and transparent, leading to significant savings in both time and funds.

In the initial phase, digital models were primarily used for depicting geometry in architecture. However, with the advancement of CAD, these models became more versatile and were used for visualizing, calculating, and simulating. CAD drawings no longer just contain geometric information; they also store semantic data for various practical applications. For instance, in a floorplan CAD drawing, the architectural elements' geometrical entities are saved in layers with a hierarchical structure. Additionally, the floorplan CAD drawing also contains other crucial details such as annotations, hatching, text, axis, and construction details.

As automation technology becomes increasingly integrated into architecture design and modeling, researchers are exploring algorithms to accomplish basic tasks in the production process. The tasks related to CAD drawings can be roughly classified as conversion and reconstruction (Filipski and Flandrena 1992, Noack 2001, Intwala 2020), detection and recognition (Truong *et al.* 2012, Ernst and Roddis 1994, Vosniakos 1992, Koutamanis and Mitossi 1992), generation (Yin *et al.* 2008, Barki *et al.* 2015, Yang *et al.* 2020), and synthesis (Dori and Tombre 1995, Li *et al.* 2022). In recent decades, developing reliable algorithms for improving production in AEC applications has always been a topic with a high level of interest in Computer graphics (CG) and AEC communities.

These basic tasks share some similarities. They do not involve advanced creativity or communication ability. In fact, a senior architectural student may complete some of the tasks, such as modeling with 3D software or drawing simple floorplans. For example, a worker with short-term training has the competent to the work of annotating semantic labels for CAD drawings with proper tools (Fan *et al.* 2021). However, a large amount of demand makes manual solutions almost impossible. The single-building-level projects may be handled with manual labor. But on city-scale hiring experts to manually process many buildings is an enormous project. The great number of labor requirements also reflected on the time. Some of the tasks are iterative processes (Nauata *et al.* 2021). A design project usually needs to be modified several turns between architects and customers and be developed with the coordination among many professions in different stages. Thus, these features promote automatic algorithms for these basic tasks, so that the relevant human labor

and time can be saved.

Meanwhile, some tasks are not so easy even for human designers, which are still non-trivial problems to develop a fully automatic method to solve the problems (Yin *et al.* 2008). Another feature of these basic tasks is that the specific conditions of practical problems are usually very complicated (Yin *et al.* 2008). There are two types of this complexity. On the one hand, for some tasks, although each specific case can be defined as a well-studied problem with clear conditions, the large number of cases makes the case-by-case solution a super complicated system (Lim *et al.* 201). On the other hand, some tasks seem easy but are hard to find reliable solutions (Dominguez *et al.* 2012 and Dosch *et al.* 2000). The main challenges to the promoting automatic method are the robustness and constraints in different applications.

Compared with the traditional rule-based algorithms, data-driven methods naturally have advantages in robustness. The basic idea of data-driven methods is to figure out the interpolation in the sample space. With proper base functions, the closer the sample space is to the data in real practice, the better the performance of the approximation. Compared to interpolation with traditional base functions, deep learning methods shows overall better performance on almost all tasks. Theoretically, with sufficient samples, the target function can be well approximated with higher robustness using proper deep learning methods.

As deep learning technology flourishes in computer vision (CV), many researchers try to introduce classic neural networks to solve the problems in AEC industries (Liu *et al.* 2017, Wu *et al.* 201). Although these attempts are validated to have better performance than traditional methods on some certain tasks. However, some of the research just convert AEC problems into CV problems and apply mature networks on customized datasets (Xiao *et al.* 2020, Huang and Zheng 2018). Usually, part of the information would be missed during this simple data conversion. Also, the universal data representation does not always fit the requirements of the AEC practice.

In this research, the topic focuses on how to define a proper CAD drawings representation method for deep learning method as well as the responding backbone neural network structure, which can be applied to other floorplan CAD drawing based problems. With analysis of the features of the CAD drawing data as well as the requirements in practical applications, the author proposes an approach to represent floorplan CAD drawings with line graphs and planar graphs and conduct corresponding graph convolutional modules. To validate the effectiveness of the proposed method, the representation is applied with two popular tasks, i.e., floorplan detection and floorplan generation. With abundant experiments and ablations, the results show that the proposed method solves these tasks with better performance and is closer to the practice requirements compared with the current deep learning methods.

1.2 CAD drawing representation

Computer-aided design (CAD) is already one of the basic technologies in designing and manufacturing, in which computers are used to generate digital prototypes of a product and play an

important role in the creation, modification, analysis, and optimization processes. Although the products in the real world are 3D geometries, 2D CAD drawing is the more common representation of research documents in practice (Kanungo *et al.* 1995). For the buildings built in the past 20 years, research plans dominated the architectural workflow (Yin *et al.* 2008). 2D CAD drawings usually convey accurate geometry, rich semantics, and domain-specific knowledge of a product design, with basic geometric primitives, such as segments, arcs, circles, and ellipses.

In architectural projects, a set of technical documentation of one building usually contains floorplans, facades, sections, and other detail diagrams. The 3D geometric information of the building is projected into orthogonal planes on different positions. The depth in the 2D scene is represented by different line types. All elements in the CAD drawing are drawn with vector primitives with geometric constraints in order to keep the accuracy in manufacturing. The primitives describe the shape, orientation, location, and size of the architectural elements and the geometric constraints depict the topological relationship among the primitives. Although this implicit geometric representation may not be direct for visualization, simulation, and rendering, it has the obvious advantage of keeping higher precision.

Besides the geometry information, various types of annotations are set around primitives to enrich the semantics in the 2D CAD drawing files. Similar to geometric information, semantic information has different detail levels in different design stages. The basic semantic information is usually the labels of each geometric primitive indicating the element type. With the project development, more semantic information about elements is enriched in the CAD files, including sizes, materials, text, and constructive details. Usually, this category information of the primitives is stored in the hierarchical structure layers and the descriptive information is stored as text attached in CAD drawing files.

Compared with the facades and sections, floorplans usually contain more architectural information, since the floorplan illustrated the locations, relationships, and usages of all rooms in one layer. Meanwhile, the structure for bearing loads in one building, i.e., columns, beams, and walls, usually corresponded in different layers, due to the vertical mechanical properties. A building with a regular shape could be roughly reconstructed with floorplans of one or several layers. A fine architectural model could be constructed by combining other information retrieved in sections and facades, e.g., heights and openings.

There are many tasks based on floorplan CAD drawings, e.g., architectural elements detection on the floorplan and floorplan generation. Although these tasks have different requirements, the deep learning methods tackling them have similar architecture, with a generic backbone and special heads and objective functions (Zhu *et al.* 2020, Khan *et al.* 2021). The priority is the way to represent CAD drawings and the corresponding neural network backbone. By observation of floorplan CAD files in public datasets and documents in real projects, the author inducts the features of floorplan CAD drawings and propose proper graph-based CAD drawings representation methods facing deep learning methods. Although the method is only tested on several tasks, it may be extended to other problems based on CAD drawings in AEC industries, such as neighborhood design in urban planning and the detection and generation of facades.

Besides applications in AEC industries, this deep learning method contrary to CAD drawings has wide usage in other fields. In the field of computer graphics (CG) and computer vision (CV), there are some similar detection and generation tasks. For example, high level-of-details city information model may require detecting elements on facades. City scene generation and synthesizing in the game industry may also achieve high performance with the proposed method in this dissertation.

1.3 Related applications

As mentioned in the background, there are lots of applications based on floorplan CAD drawings. Among them, are the two most popular tasks, which are studied from rule-based algorithms to deep learning methods, since they have big commercial usage in practices (Huang and Zheng 2018). In this research, these two tasks are taken as examples for applying the proposed CAD drawings representation method and related graph convolution structures.

1.3.1 Floorplan detection

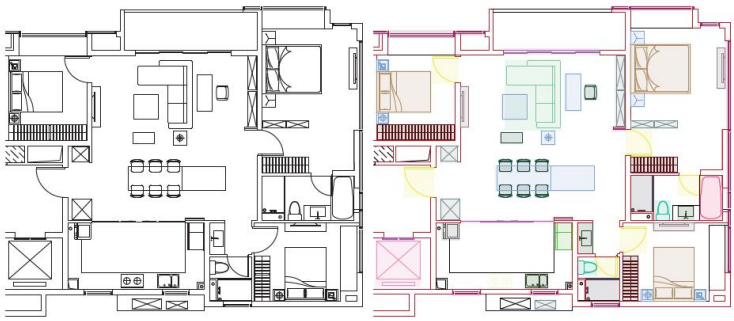
In the field of AEC fields, floorplan detection problems are studied during building modeling. As the rising demands on modeling of existing buildings, approaches are proposed with different types of sources data, including images, pointcloud, and CAD documents. Compared with other data obtained with optical devices, geometric data contained in CAD files is more accurate. However, unlike the geometric information, semantic errors are not so obvious and may be reserved in the camera-ready version data. This phenomenon mentioned in PanoCAD (Fan *et al.* 2021) may be even worse if a project is subcontracted several times. Traditionally, the data-cleaning process of alignment of correct semantic tags to corresponding geometric primitives still requires human labor, which restricts larger-scale applications with a high level of detail.

Floorplan detection is an interesting task for many decades in communities of many subjects. In the field of CV, the floorplan detection problem is studied as an optical character recognition problem, in which CAD drawings are treated as raster images. Spotting and recognizing symbols from CAD drawings is the first step towards understanding their content, which is crucial to many real-world industrial applications, see **Figure** 1. For example, 3D digital building modeling with semantic has a growing demand in various architecture engineering areas such as pipe arrangement, construction inspection, and equipment maintenance. A floorplan usually contains complete details of a floor in an orthogonal top-down view. By developing efficient perception algorithms on CAD drawings, a huge amount of labor and time could be saved for creating digital models.

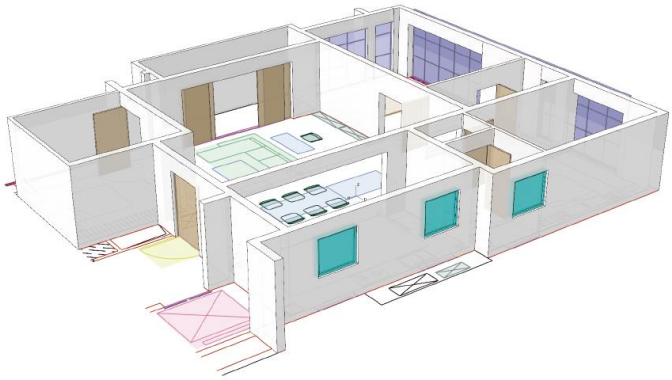
In document analysis and recognition, symbol spotting refers to locating and recognizing graphic symbols embedded in the document, e.g., circuit diagram and floor plan. Previous methods usually format the process into two stages: spotting the target symbol first, then classifying it. Typical non-data-driven approaches are conducted in a query-by-example fashion. However, the query template is not always available and symbols representing the same object may vary dramatically, limiting their application in production. More recent data-driven methods leverage the power of deep learning and show better generalization ability.

Traditional symbol spotting usually deals with instance symbols representing countable things, like tables, sofas, and beds. Following the idea extended the definition by recognizing the semantics of uncountable stuff and named it panoptic symbol spotting. Therefore, all components in a CAD drawing are covered in one task altogether. For example, the wall represented by a group of parallel lines was properly handled, which however was treated as background.

CAD drawings are vector graphics consisting of segments, arcs, circles, and ellipses, posing unique challenges to the symbol spotting problem. CAD drawings are vector graphics consisting of segments, arcs, circles, and ellipses, posing unique challenges to the symbol spotting problem. However, applying existing CNN-based algorithms on rendered CAD images will lose the topology and accuracy required in following applications like 3D model reconstruction.



(a) CAD drawing (b) Panoptic symbol spotting



(c) Reconstructed 3D model of wall, window and door elements

Figure 1 3D model of a residual building from 2D floorplan CAD drawings. This research focuses on the first step, which is detecting all architectural elements in the CAD drawing of floorplans.

Facing the above challenges, a solution is provided in this research with deep learning methods combined with architectural knowledge. The author proposes a network namely CAD-GATnet (Zheng *et al.* 2022) on the representation ability of feature space with additional regularity terms and gets better results on parts of classes. Although there is research using the GNN network to detect the lines of walls in CAD drawings, walls cannot be reconstructed only using labeled segments. Vector graphs are converted to triangulars and then combined CNN-based wall prediction to a generated clean polygon of walls, which could be used for other applications.

1.3.2 Floorplan generation

Floorplan design is a vital process in building design projects, in which the functions and relations of architectural space are approximately established. In most cases, floorplan design is a process that costs highly both on time and funds. On one hand, a proper floor plan requires massive iterative modification and development under the communication between architects and customers. On another hand, concreting the idea from customers into the floorplan drawings asked architects to have rich experience as well as professional knowledge.

As mentioned in HouseGAN (Nauata *et al.* 2021), in the North American residual building design market, only a small part of users could afford a high-quality personalized house design service. This potential market promotes automatic floorplan generation algorithms developing for decades, which should quickly generate floorplan drawings with given input proposals from users as well as have flexibility on demands expansibility.

In traditional architectural design methods, the floorplan design task is usually a hierarchical design process, which could be divided into several sub-tasks, including room layout design, entrance design, window design, and door design. According to the logic of architectural design, entrances, windows, and other architectural elements in floor plan drawings are sub-attributions of walls. Thus, room layout design is usually the upstream task for other tasks during floorplan design.

Figure 2 Hierarchical floorplan design process, in which the room layout design is the up-stream task. (a) Bubble diagram. (b) Room layout design. (c) Entrance design. (d) Window design. (e) Door design. (f) Fine drawn floorplan.

In this dissertation, the author presents a novel neural-guided method to automatically generate orthogonal room layouts with plausible shapes and the same topology compared with the given bubble diagrams. In former research, a type of graph data, namely bubble diagram, is frequently used to describe the user preference for architecture design, in which a vertex with position and type

attribution represents a certain room in floorplans and an edge connects two vertices if two rooms are adjacent. With such input data, the generated room layout results should have reasonable shapes and satisfy all requirements in the given bubble diagrams.

The metrics for requirements in the given bubble diagrams are well-defined, but the reasonable shape is a rough notion, that refers to the room sizes and shapes that conform to common sense in architectural design. For architects, the room layout design with given bubble diagrams for residual buildings may be a basic task even senior architecture undergraduates could accomplish it. However, formulating this task and conducting an algorithm to generate a proper room layout is a non-trivial problem.

Several works made attempts to solve this problem from using traditional optimization methods 30 years ago to deep learning techniques recently. Graph-based optimization method could figure out topological consistent rectangle room layouts. However, these methods can only generate the topological equivalence class of the given bubble diagram, and the realistic shape of rooms is not guaranteed. By programming shapes grammar into an iterative optimization, Wang can generate some complex cases with better shapes but still fail on keeping topological consistency. The development of deep learning techniques uses image generation networks to obtain plausible room layouts with higher robustness on the given input constraints. However, the network architectures of these methods restrict deep learning methods from generating a completely correct topology. Overall, none of the existing algorithms meets the requirement, especially on topological consistency.

To tackle the above difficulty, the author proposes a novel neural-guided planar graph drawing method for room layout generation by combining the advantages both of deep learning and optimization methods, so that the generated room layouts hold similarity on both shape and topology levels. In this method, a dual-graph-based neural network is used to generate realistic shapes by predicting the topology and geometry of targeting room layouts. Instead of designing a generative network to satisfy the planar and orthogonal constraints, an additional post-process optimization stage is added to obtain the results. In this stage, the orthogonal planar graph drawing problem is formulated as a minimum-cost flow problem and solved it in pseudo-polynomial time.

1.4 Overview

1.4.1 Research questions

The overall research topic is developing graph-based representation learning methods for CAD drawings and conducting related neural network structures. Two applications in this dissertation were derived from a real problem in the industry during the author’s research internship at Alibaba Inc. The floorplan detection problem is a demand to develop a high-level automatic approach to reconstruct 3D digital models of several blocks in the Xiong'an New Area. Until now, part of the work is applied in production. And floorplan generation is an open problem in CV and AEC communities, derived from the actual needs in practical applications. With the consideration that the

problem is based on real projects, the following research questions arise:

- How to represent a CAD drawing?
- How to conduct a suitable backbone network for CAD-drawing-based tasks?
- How to detect element instances in a graph during floorplan detection?
- How to keep topology consistent during floorplan generation?

The thesis focuses on developing a methodology for using deep learning methods on CAD-drawing-based tasks. Although mature deep learning methods built for classical problems in CV are borrowed to solve these problems by converting CAD drawings as raster images, the performance is not so satisfying. In this research, the author meticulously analyzes the unique characteristics of the CAD drawing data and propose two graph-based data structures to describe the floorplan drawings.

Meanwhile, floorplans are artificial patterns, which have large differences from natural images. Usually, there are strong regularities in floorplans, such as parallelism, orthogonality, and duality. The author also inducts these properties and improve or propose novel network structures which could remarkably improve the performance of these tasks.

The rest part of the research could be separated into the detection part and the generation part. In the floorplan detection part, the symbol spotting problem integrated with symbol recognition and instance detection is solved with an end-to-end network. In the floorplan generation task, the floorplan generation with frequently used constraints is solved with a hybrid method, in which the topology and the geometry are predicted with a dual graph-based neural network, and an optimization method is introduced to process the predicted results under the given constraints.

1.4.2 Main Challenges

The main challenges are partly reflected literally in the research questions. The main challenges are three-fold:

- The data processing and representation.
- How to define convolution structure to the proposed representation?
- How to deploy the representation method to detection and generation tasks?

The data challenge would be faced in almost all data-driven problems. In the detection problem, the input data used in the project is raw CAD drawings provided by the architecture owners, which are mainly as-built CAD drawings. Since the scale of the building ranges from houses to large public architectures, it is difficult to feed the neural networks with whole floorplans. Meanwhile, although all geometric primitives in floorplan dataset are well annotated by human expert, it is necessary to

conduct rules to convert raw files to vectors graphics for deep learning methods. For the dataset is established by extracting floorplans on brochures, the raster images still need to be processed into vector data, then to related graph-based representations. Techniques about computer graphics and image processing are involved in this stage.

With the proper graph representations, the corresponding convolution structures are also required to be designed. Usually, the feature is defined on vertex space in common graph data. In the proposed graph representations, features are also defined in edge and face space. The challenge is about how to design a graph-based neural network structure, which could make fully use of the input information.

The detection and generation problem are well studied in raster images and solved by treating CAD drawings as raster images and direct borrowing the mature detection and generation neural networks designed for natural images. In this research, line graphs are used to describe the CAD drawings, since they are native vector graphics. The reason and the analysis are detailed in the methodology section. However, there are still problems when deploy the representation to the specific applications. In floorplan detection, although there are node and edge classification tasks on classic graphic data, there are few tasks aiming to detect the instances. To figure out each architectural element in floorplan CAD drawings, the instance detection problem needs to be further defined. In floorplan generation, classic graph generation only cares about the topology generation rather than the embedded shape of the graph. Thus, there is also a challenge on how to redefine the proposed floorplan graph generation task to meet the requirement in practical applications.

1.4.3 Objectives of the thesis

To tackle the proposed research questions, the author takes a hypothesis that using a deep learning method with a delicately designed representation method and neural network structure according to the features of floorplan CAD drawings data could the improve the performance compared with other current deep learning methods. The objectives of the thesis are listed as follows:

- Induct the feature of floorplan CAD drawing data.
- Define related problems on graph data.
- Propose feasible network structures for each task.
- Perform experiments to validate the proposed networks.
- Analyze and evaluate the results.

Although one of the important usages of CAD drawings is visualization during each stage of architectural design n, a raster image is not the best way for representation. The performance of image-based solutions is restricted more or less by this pixel representation in the practical applications in AEC industries. Thus, the author analyzes CAD drawing files from the semantic and

geometric perspectives and induct the features which could be used to improve the performance. These induced features are described in the methodology section 0 and lead to a special graph-based representation for deep learning methods.

Unlike the classic graph data, e.g., social networks or citation networks in the field of computer science, the graph representation of CAD drawings also has some special properties. Most buildings are man-made structures with strong regularities. This property reflected in floorplan CAD drawings are situations, including parallel and orthogonal walls, periodically occurred structural elements and furniture, and axis for alignment. A novel network backbone structure is also designed in this research, particularly toward these regularity features. This backbone is not just a case-by-case study result, but a structure that can be popularized for other tasks based on floorplan CAD drawing data. This motivation and insight into the designed network backbone structure are illustrated in the methodology section 0.

0Only the backbone is not enough to solve the specific problems. Since the floorplans in diverse usages still have a different level of information, the author further analyzes the floorplans in the situations of detection and generation. The completed networks with the proposed backbone structure are introduced for the floorplan detection task and floorplan generation task respectively. These two networks are illustrated in section 3.3 and section 0 of the methodology.

With the proposed networks, the author designs and performs adequate experiments to illustrate the performance of each task. Besides the proposed method, parallel experiments are also performed with state-of-the-art methods using similar and generic settings for a fair comparison. Qualitative and quantitative results and details in the experiments are illustrated in experiment section 0.

More ablation studies are conducted for efficiency and the validation of the proposed method and network structures. Besides, the author also analyzes the results in the experiment section and point out the failed cases and limitations of the proposed method.

1.4.4 Main contributions

To tackle the above challenges, a novel method of graph-based representation learning for floorplan CAD drawings is proposed in this research. In general, the primary contributions of this study can be divided into two aspects in such interdisciplinary field of artificial intelligence in construction.

In theoretical aspects:

- Common geometry and topology features in floorplans CAD drawings are decoupled for representations in deep learning.
- Line graph and planar graphs are used to depict floorplans CAD drawings features.
- Special-designed graph convolution are defined as common backbone to better embed features in line and planar graph.

In application aspects:

- Two specific network structures are deployed based on proposed backbone for applications in AEC industries.

- The customized algorithmic workflow produces results with topology consistency, which is closer to the practical demands of the construction industry.

- The proposed representation and structures have better performance in the detection and generation tasks compared with current methods.

From the perspective of architecture, this study analyzes data features and demand characteristics, combining deep learning to propose efficient detection and generation algorithms. From an artificial intelligence standpoint, this study introduces an innovative and effective network architecture for defining graph convolutions. Further details regarding theoretical and technical innovations will be presented in the conclusions of the relevant chapters.

1.4.5 Structure of the thesis

This thesis consists of several parts. The main structure is shown as figure 3.

The introduction part (Chapter 1) is a general introduction, in which some concepts are introduced, and the objective is clearly defined. The upper-mentioned challenges would be illustrated elaborately as well as the corresponding methods proposed in this thesis.

In the state-of-the-art section (Chapter 2), related works in both AEC and CV would be classified and enumerated. The relationship between related works and methods in this thesis would also be explained in this section. After that, content based on peer-reviewed publications is presented to respond to the questions put forward ahead.

Chapter 3 is the methodology response to the CAD representation and corresponding network structures. The features of the floorplan CAD drawings are analyzed and inducted in 0. After that, the author proposes a generic backbone structure with such features in 0. Details and theories for floorplan detection and floorplan generation are illustrated in 3.3 and 0 respectively.

Chapter 4 is the experiments for the proposed method. The experiments are conducted in two parts responding to the detection and generation tasks separately. The comparison experiments of floorplan detection are illustrated in 4.1 and the related experiments of floorplan generation are shown in 4.2.

Chapter 5 discusses the results in the upper section. More ablation studies are conducted to analyze the insight of the proposed method and to validate the structures of the proposed networks. Similar to the upper chapter, detection, and generation are discussed separately in 5.1 and 5.2.

The thesis ends with Chapter 6, which summarizes the methodology as well as the experiments and makes an outlook on the related tasks and future work.

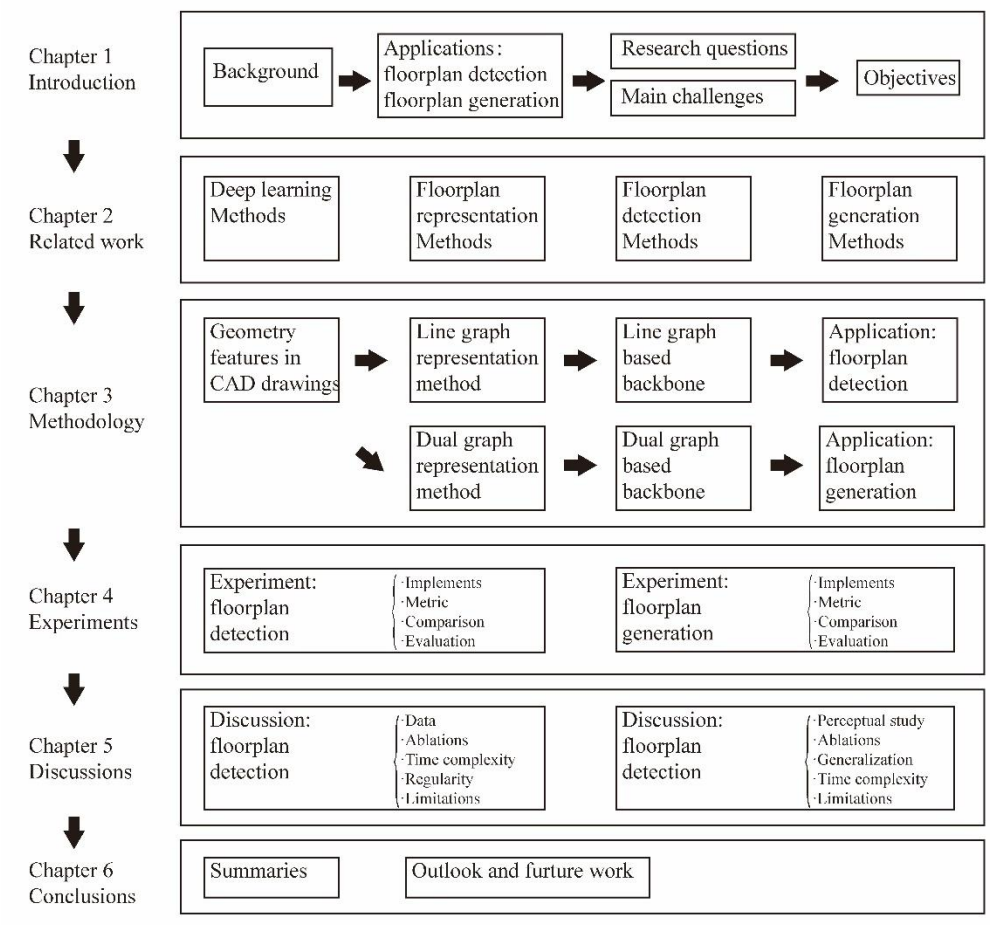


Figure 3 The diagram of the dissertation structure.

2 State of the art

This section briefly introduces the basic structure of the neural network, the classic deep learning tasks, and classic neural networks. After that, floorplan detection and generation are summarized from traditional solutions decades ago to deep learning methods in recent years.

2.1 Deep learning methods

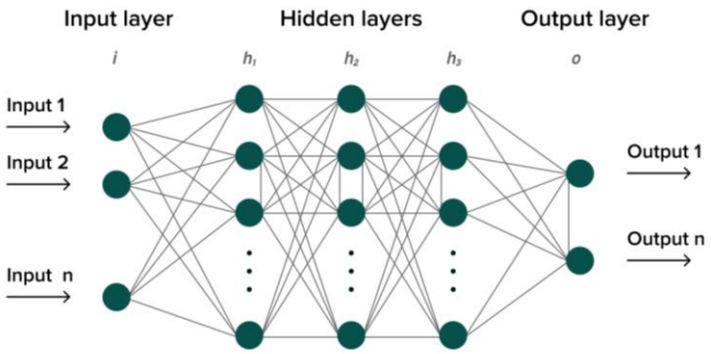


Figure 4 A brief architecture illustration of a naïve neuaral network. With a given vector as input, a target vector is predicted. Each neural unite in the network is a nonlinear base function, which is trained with the backpropagation algorithm.

Deep learning is a machine learning technique for representation learning, based on neural networks (Deng *et al.* 2014, Bengio *et al.* 2009, 2013, Lecun *et al.* 2015). As is shown in Figure 4, the structure of a neural network is composed of three types of layers, i.e., the input layer, hidden layer, and output layer. Usually, neural networks with more than two hidden layers are called deep neural networks.

Each node in the network imitates the neurons in the biological nervous system (Fukushima 1980), which is a composition of a linear function with learnable weights and bias, and a nonlinear function called the activation function. Forward propagation refers to the vector space of the input feature passing each layer and calculating the loss with a defined loss function. The update process of the parameter in neural networks uses a backpropagation algorithm (Werbos *et al.* 1974), which is a gradient decent algorithm to optimize the parameters in every node based on the chain rule. Thus, the activation function and the loss function should be differentiable. Although there are many optimizers for training networks, the most important hyperparameter is called the learning rate. This parameter is also called step length in the context of numerical optimization.

With the growth of computing power, deep learning technique improves dramatically in nearly twenty years. Variants and complicated structures of neural networks are proposed and show better performance than traditional methods. Some of them even surpass human experts in certain tasks.

2.1.1 Convolution neural networks

Convolutional neural network (CNN) is a class of artificial neural networks most applied to analyze visual imagery. CNNs use a mathematical operation called convolution in place of general matrix multiplication in at least one of their layers. They are specifically designed to process pixel data and are used in image recognition and processing. They have applications in image and video recognition, recommender systems, image classification, image segmentation, and natural language processing. A standard CNN is usually composed of several convolutional layers, pooling layers, and fully connected layers. The convolutional kernels in the convolutional layers usually have limited size which filters local features, while the global features are extracted with convolution layers going deeper and pooling layer halfling the size of the images.

Although CNN was proposed in the early time (Fukushima 1980), it can only handle some toy tasks until the AlexNet (Krizhevky *et al.* 201). The following models (Zeiler *et al.* 2014, Szegedy *et al.* 2015) improve the structure of CNN and have better performance. VGGNet (Simonyan *et al.* 2014) is another milestone eliminating the initial fully connecting layer and only takes 3×3 convolutional kernel and 2×2 pooling size, which decrease the number of parameters and improve the efficiency of parallel computing. ResNet (He *et al.* 2016) introduce a shortcut in the different convolutional layers, which solves the problem of network degradation as the number of layers deepens.

2.1.2 Generative network

The generative network includes variational autoencoder networks (VAE) and generative adversarial networks (GAN). VAE is related to the autoencoder model because the two have a certain affinity in structure, but there is a big difference in goals and mathematical expressions. VAE belongs to the probability generation model, and the neural network is only one of its components, which can be divided into encoders and decoders according to different usages. An encoder maps input variables to a latent space corresponding to the parameters of a variational distribution so that multiple different samples from the same distribution can be produced. The decoder does the opposite, mapping from the latent space back to the input space to generate data points. Such models were originally designed for unsupervised learning but have also shown excellent effectiveness in semi-supervised (Dilokthanakul *et al.* 2016, Hsu *et al.* 2017) and supervised learning (Ehsan *et al.* 2017, Xu *et al.* 2017).

GAN learns through two neural networks playing games against each other. This method was proposed by Goodfellow *et al.* (2014). Generative adversarial networks consist of a generative network and a discriminative network. The generator network randomly samples from the latent space as input, and its output needs to approximate the distribution in the training set as much as possible. The input of the discriminative network is the real sample or the output of the generation network, and its purpose is to distinguish the output of the generation network, while the generative network should deceive the discriminative network as much as possible. The two networks fight against each other and constantly adjust the parameters. The goal is to make the discriminant network unable to judge whether the output of the generating network is true or not. Many popular

GANs (Goodfellow *et al.* 2020, Karras *et al.* 2017,2019), can already generate realistic results in generative tasks.

2.1.3 Graph neural network

There has been increasing interest in the extension of deep learning methods to graphs. Although traditional deep learning methods have been applied to extract features of Euclidean space data with great success, the data in many practical application scenarios are generated from non-Euclidean spaces, and traditional deep learning methods are not effective in processing non-Euclidean space data. Researchers drew on the ideas of convolutional networks, recurrent networks, and deep autoencoders to define and design a neural network structure for processing graph data. Graph neural networks have been widely used in the fields of computer vision, recommender systems, traffic, chemistry, and others.

Graph neural networks can be roughly divided into three categories, graph convolution networks (GCN), graph attention networks (GAT), and graph autoencoders (GAE). GCNs fall into two categories, spectral and spatial-based methods. Spectral-based methods define graph convolutions by introducing filters from the perspective of graph signal processing, where graph convolution operations are interpreted as removing noise from graph signals in the research of Kipf *et al.* (2016). Spatial-based methods characterize graph convolutions as aggregating feature information from nearby neighbors (Gilmer *et al.* 2017). GAT proposed by Velickovic *et al.* (2017) replace the graph convolution with the learnable attention mechanism among the vertices in the graph and shows better performance compared with the vanilla GCN. GAEs are autoencoder networks based on graph data proposed by Kipf *et al.* (2016) and are usually used for the graph generation.

2.1.4 Transformers and multi-modal

Transformer, also known as self-attention, is a module of the network, firstly proposed in the field of natural language process (Vaswani *et al.* 2017), then shows great potential in the field of computer vision (Dosovitskiy *et al.* 2020, Liu *et al.* 2021). The insight of the transformer is inspired by the attention mechanism, using a learnable attention score matrix to represent the relationship intensity between every two tokens. Once the transformer showed great success on the task of English German translation task, many more upgrading networks occurred. Dai *et al.* (2019) used relative positional encoding rather than absolute positional encoding in long sequences. Zaheer *et al.* (2020) found that the attention matrix in the long sequence is usually sparse and local-focused. Guo *et al.* (2021) proposed a star-satellite structure network to decrease the complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$. The query-key-value layers in transformer, naturally provided a good solution for the alignment of multi-modal. Carion *et al*. (2020) took embedded image patches as tokens and formulated the detection problem as a translation problem. As the permutation invariant of the transformer, for symmetric elements data (Maron *et al.* 2020), e.g., graph and point cloud, the transformer also has a good structure for aggregation and pooling (Ying *et al.* 2018). Velickovic *et al.* (2017) added a transformer structure on a graph neural network to aggregate neighbors to the center node. As the geometric primitives are usually different, compared with simple graphs, a heterogeneous graph could more precisely depict a CAD drawing. Hong *et al.* (2020) and Wang *et al.* (2019) added a

transformer structure on the graph constructed with a meta-path. Hu *et al.* (2020) projected different relationships into the same feature space, which replaces the meta-path with a transformer.

2.2 CAD drawings representation

Floorplan representation has been a fundamental issue in designing a floorplan (Yao *et al.* 2003). During production, the CAD drawings exported by professional software are usually saved in various generic data formats, such as DXF, ACIS, DWG, etc. However, the design-oriented files usually contain redundant information and require specific parsers to extract target content. Two frequently used representations of CAD drawing in recognition and generation tasks are introduced in this subsection.

Image representation. Since the most common user interface of floorplan CAD drawings are 2D visualized canvas, *i.e.*, monitors and research printings, the intuitive representation is raster images. The demand for using raster images to represent CAD drawings is twofold. Both of them originate from the practice scenes. Although research drawings are already replaced with digital drawings in many scenarios, architectures-built decades ago only have printed CAD drawings. With digitizing processing of the documentation, many researchers try to conduct algorithms to vectorize the raster drawing images scanned from research documentation. Doshi *et al.* (2000) split the CAD images into simple patterns and merged the detection results. Or *et al.* (2005) propose a pipeline for converting CAD images to digital files and detecting walls in the drawings. Dominguez *et al.* (2012) and Zhu *et al.* (2014) use parallel pairs (PP) algorithms to detect wall segments. Combining image-based neural networks, Huang and Zheng (2018) use a mature segmentation network to predict a label mask for the given room layouts. Li *et al.* (2017) and Zeng *et al.* (2019) propose originally designed neural network structures to predict the labels of each pixel in the raster CAD drawing images. Another application is generating room layouts for illustration in real estate. One approach applies conditional generative adversarial networks (cGAN) to generate floorplan as raster images (Nauata *et al.* 2020,2021, Wu *et al.* 2019, Huang and Zheng 2018[57], Rahbar *et al.* 2019,2022).

The shortages of image representation are obvious. Firstly, as mentioned in the above works, the geometry primitives are explicitly implied in the raster images. Secondly, many parameters are required during rendering. Some of the rule-based algorithms are very sensitive to rendering parameters, such as line weight and line type. Thirdly, the geometric constraints are not described in the raster images.

Graph representation. Another commonly used representation for CAD drawings is the graph (Trudeau, 1993). In discrete mathematics, a graph is a structure composed of a set and the relationship among the elements in this set (West 2001, Tutte 2001[212]). In the past decades, graph theory has demonstrated that it has an effective tool for manipulating space diagram problems and floorplan design in both AEC and very large-scale integration (VLSI) (Zhi *et al.* 2003). Huang *et al.* (2008) use the naturally defined graph to describe the geometry primitives in the floorplan CAD drawings for recognition, in which geometric vertices and lines are set as vertices and edges in the graph. Further, the binary tree was also widely used in mosaic or slicing floorplans, especially for the circuit

design, which has many variations with different combination numbers (Otten 1982, Knuth 1972, Guo *et al.* 1999, Hong *et al.* 2000). Other approaches generate floorplans with space diagrams describing the rooms and their inter-relationship, such as X-Y separated constrained graph (Schwarz *et al.*1994a,1994b), topology-geometry separated constraint graph (Levin 1964, Garson 1970, Roth *et al.* 1982). In deep learning approaches, the graph representation just inherits the naturally defined graph and applies to the vanilla graph neural networks (Hu *et al.* 2020, Sun *et al.* 2022).

limitations in these graph-based representation methods are twofold. On one hand, graphs with complex structures are usually used in VLSI designs and require strict conditions. Due to the different practice scenarios, the conditions in VLSI are usually not held in AEC. On the other hand, the naturally defined graph on floorplan CAD drawings cannot fully describe the geometry information. Unlike other graph-based data, *e.g.*, citation networks or social networks, the feature space is usually defined on the edge spaces instead of the vertex spaces.

2.3 Related work in floorplan detection

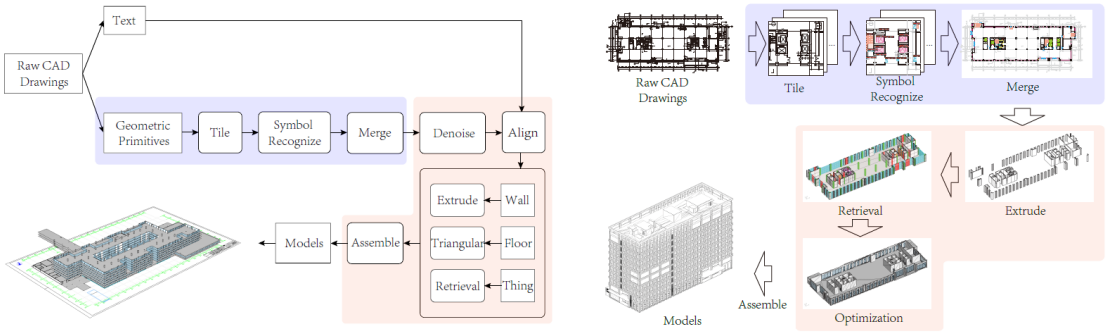


Figure 5 In many frameworks of 3D model reconstruction, the architectual elements on as-built floorplan CAD drawings are first dectected and than converted into 3D models.

2.3.1 Tradition methods

Before the appearance of deep learning methods, there are research using 2D architectural plans to generate massing shapes of 3D building models. Most methods at that time focused on model generation and annotated primitives with a high level of manual labor. So *et al.* (1998) proposed a pipeline with fully manual symbol recognition and automatic model generation. In the same year, Lewis and Sequin (1998) proposed a prototype system namely Building Model Generator, by automatically grouping geometries into specific layers and then manually mapping semantic names with corresponding layers. Lim *et al.* (2018) provided a method with a simplification process after manual annotation, improving the robustness of data. Although manual annotations show the best results, for most researchers, a process that could automatically detect all symbols is more attractive.

Before the wide usage of deep learning in the field of computer graphics and computer-aided design, most researchers developed expert systems based on inducted empirical rules. Dosch *et al.* (2000),

firstly split raster images of architectural plans into overlapping tiles, then used a Hough transform-based approach for segmentation and finally merged the tiles by Hausdorff distance defined on the pixel level. There are elements with explicit geometric features, which could be detected with handcraft symbol descriptors to describe the shape (Nguyen *et al.* 2008, 2009, Rusinol *et al.* 2010). In this research, query symbols and instances in documents are matched with window shifting. By considering some elements, *e.g.*, wall and beam, are depicted as two parallel pairs (PPs) in 2D architecture plans, Lu *et al.* (2005), proposed an algorithm that detected structural entities. Lu *et al.* (2007), proposed a hierarchical SINEHIR model, which could recognize more complicated symbols, e.g., doors, windows, stairs, etc. Or *et al.* (2005), provided an algorithm namely span list and take quarter circle representing doors as a priori, after detecting lines on raster images.

Beside detection on raster images, Medjdoub *et al.* (2000) used graphs to depict the relationship between two primitives. Zhi *et al.* (2003) used planar graphs to describe the connection and location of walls and doors in plans, to figure out corridors by detecting loops in graphs. Instead of matching queries and keys on the pixel level, Dutta *et al.* (2013a, 2013b) proposed a series of algorithms that utilize graph matching for query target symbols in CAD drawings. Dominguez *et al.* (2012) detected walls with wall-adjacency graphs (WAGS) and gained better performance than detecting PPs.

Almost all rule-based algorithms work well on simple and isolated symbols. When cases become complicated, such as symbols embedded in the background, these algorithms usually would fail. The reason is obvious, all rules are designed against some typical cases, making algorithms sensitive to noise and topology errors.

2.3.2 Deep learning methods

With the development of GPU, deep learning technique shows superior performance on computer vision tasks than traditional methods based on hand-crafted descriptors. Naturally, researchers attempted varied networks of symbol recognition. Due to the formation of embedded feature space of CAD data, networks could be roughly separated into two types, CNN-based networks, and GNN-based networks.

CNN is short for convolutional neural network, which is a typical network showing great performance on the tasks for raster images, including classification, detection, and segmentation (He *et al.* 2016, Long *et al.* 2016, He *et al.* 2017, Ren *et al.* 2015, Redmon *et al.* 2016). CNN could also be used for symbol recognition if rendering CAD drawings as raster images. With proper annotations, symbol recognition on vector graphs could be transferred into detection or instance segmentation on images. Besides, additional post-process is needed to map pixels back to the original primitives.

In the field of computer vision, due to the lack of high-quality CAD drawings datasets, research on symbol recognition is usually based on floor plans and human sketches, which share similar properties. Sun *et al.* (2012) and Li *et al.* (2018) used CNN to align proper semantic labels for strokes in sketches. Huang *et al.* (2014) predicted labels for each pixel and figure out labels of stroke by voting. Compared with sketches, floorplans are more relevant to 2D CAD drawings. Liu *et al.*

(2017) used masked CNN to detect the room and junctions of the wall, the 3D indoor scenes are reconstructed after a post-processing step. Zeng *et al.* (2019) used VGG as the backbone, with attention between the boundary detection heads and room-type prediction heads, solved cases that Liu cannot handle.

GNN is short for graph neural networks. Compared with raster images, graphs are usually simple complex without Euclidean data for not having a certain structure among neighbors (Bornstein *et al.* 2017, Zhou *et al.* 2020). As graphs are a common data type that depicts the node and relationship among them, several outstanding networks work well on tasks for graphic data, including graph classification, node classification, link prediction, and graph generation (Defferrard *et al.* 2016, Kipf *et al.* 2016a, Atwood *et al.* 2016, Wang *et al.* 2019, Kipf *et al.* 2016b). Many researchers used graphs with primitives as nodes and relationships or adjacency as edges to optimize recognition results (Horna *et al.* 2009). Li *et al.* (2018) added a graph cut after image segmentation. Yang *et al.* (2020) proposed a mix pooing module which aggregates features of node level to stroke level and improved the performance a lot. Lambourne *et al.* (2021) took B-rep surfaces rather than vertices as nodes and proposed a network for 3D CAD model classification. Fan *et al.* (2021) combined CNN and GCN and proposed a network using GCN smoothing the features from CNN and improving the results. Also, they made great a contribution to proposing the FloorplanCAD dataset.

Methods based on CNN share two vital shortages. One is different to determine super parameters when rendering CAD drawings to raster images. The primitives are varied in scale and not distributed in uniform densities. Another is that the overlapping primitives would hold the same features after rendering into images. In this research, CAD drawings are represented as heterogeneous graphs with regularity terms based on architectural design knowledge.

2.4 Related work in floorplan generation

Figure 6 In traditional building design processes (Lu *et al.* 2017, Merrall *et al.* 2010)), especially in residual building design, the room functions and relative positions are determined with bubble diagram. All following steps are designed with the requirements in the bubble diagram.

2.4.1 Traditional methods

Traditional methods conduct rule-based algorithms or formulate optimization for structured data generation (Arvin *et al.* 2002, Medjdoub *et al.* 2000, Michalek *et al.* 2002, Liu *et al. 2*013). In early efforts, shape grammars are introduced to develop iterative algorithms for room layout generation. Further, Wang and Zhang (2020) use shape grammar to generate room boxes layout based on the room adjacency relationship. Rectangular drawing algorithms for planar graphs are also used to generate layouts, meanwhile, house owners' preferences are considered as constraints (Nishizeki and Rahman, 2004, Yeap and Sarrafzadeh, 1993). With real-world data, Merrall *et al.* (2010) train a Bayesian network to generate room adjacency and obtain floorplan by stochastic optimization. Wu *et al*. (2018) propose a hierarchical framework to solve this task with mixed integer programming optimization.

Although these methods may be applied in computer games, they still cannot meet the requirements of architectural design. These methods either need elaborately designed constraints or can only generate topological equivalent results, whose shape similarities are not guaranteed.

2.4.2 Deep learning methods

Since floorplans are commonly represented as images, it is intuitive to classify floorplan generation into image synthesis and generation problems. The most popular neural approaches are Generative Adversarial Networks (GANs) (Goodfellow *et al.* 2014, Karres *et al.* 2017, Zhang *et al.* 2019, Brock *et al.* 2018, Karres *et al.* 2019). Naturally, GANs are used on room layout generation tasks (Li *et al.* 2020). Wu *et al.* (2019) predict walls by supervised learning component masks with a Convolutional Neural Network with given house boundaries. Their following by Sun *et al.* (2022) using GraphNet progressively draws walls with a parallel LabelNet, which requires no post-processing stage. Another influential work is the HouseGAN series by Nauata *et al.* (2020,2021), using conditional GAN to generate rooms one by one under bubble diagram constraints.

Compared with traditional methods, these pixel-level methods show great superiority in realistic results as well as robustness to constraints. Although the constraints are considered when designing network architecture, the deconvolution structure in these networks has fewer advantages on topology prediction.

As bubble diagrams are usually described as graphs, variants of Graph Neural Networks (GNNs) (Kipf *et al.* 2016a, Zhang *et al.* 2018, Ying *et al.* 2018, Velickovic *et al.* 2017, Kipf *et al.* 2016b, Scarselli *et al.*2008) are used for network design and feature encoding. With the assumption that all rooms are rectangular, Graph2Plan by Hu *et al.* (2020) takes given bubble diagram constraints as input and predicts the locations and sizes of the room boxes. Chen *et al.* (2020) generates room boxes from language descriptions with a graph convolutional network. Two upper methods require a post-processing stage to obtain desired floorplans. Para *et al.* (2021), propose a two-stage method, using GNN to predict room vertices and adjacency and optimize results with linear programming.

These graph-based methods are usually less computationally intensive since the graph is a more

elegant and efficient representation of floorplan drawings. However, these methods cannot guarantee the generated floorplans conform to the given constraints. Furthermore, taking rectangles as the basic room shapes limits their ability to generate diverse room layouts.

2.5 Summary

2.5.1 Issues in floorplan detection

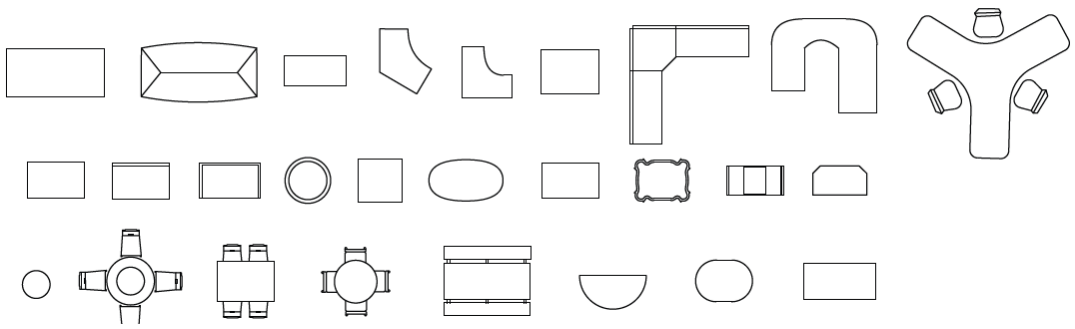


Figure 7 One challenge is the difference in the same architectural elements type. For example, tables may have various drawing ways with simple or complicate geometry in different projects.

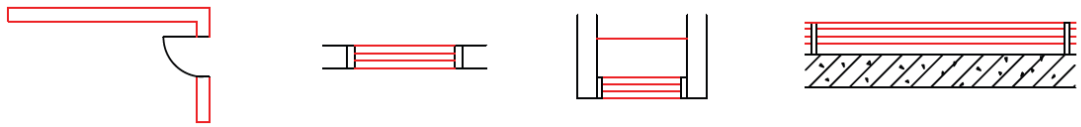


Figure 8 Walls, windows, bay windows, and isolations has similar drawings.

Another challenge is the similarities between different type architectural elements. This challege is threefold:

- Lacking primitive-grained instance segmentation methods
- Representation of the as-built floorplan CAD drawings
- To distinguish the difference between inner-class primitives and the similarity of inter-class primitives.

2.5.2 Issues in floorplan generation

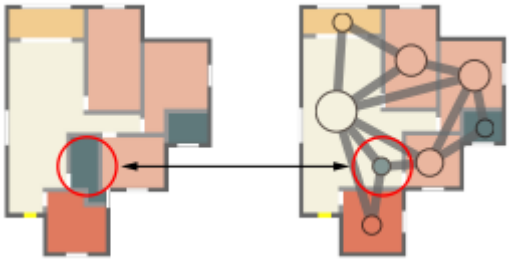


Figure 9 Generated floorplan (right) with wrong room topology (missing one room) .

The main challenge in floorplan generation is the topological consistent. Current methods are either make errors in topology or generater unreal floorplans. Thus, a good method should satisfy the following properties:

- A method can generate vector graphics of room layout diagrams.
- Generating room layouts to satisfy all strict constraints in the input conditions.

3 Methodology

Tasks like floorplan detection and floorplan generation are typical tasks, which require certain professional knowledge and high cost on time and human labor. An automatic method may complete these tasks with the help of computers. Since it is hard to establish a model-driven method with believable quantification of the parameters of architectural design, data-driven methods, especially deep learning methods are more suitable for these tasks. Reliable automatic methods with deep learning techniques may greatly improve efficiency and productivity.

Existing deep learning methods in other related fields could complete these tasks to some extent by converting floorplan CAD drawings to other data formats. Compared with raster image data and graph data, CAD drawings data have special features, which are worth researching. Current Methods based on such converted data usually have some shortages in these tasks. An originally designed method developed with consideration of floorplan CAD drawings features may have better performance in floorplan detection and generation.

The methodology section contains four parts. The theoretical analysis of basic features and backbone network structures are introduced in the first two sections. The following two sections separately illustrated the methods for floorplan detection and floorplan generation.

In section 0, the floorplan CAD drawings in different architectural design stages are introduced. and the special properties of the CAD drawings are inducted from the perspective of geometry. Parameterization, regularity, planarity, and duality features of floorplan CAD drawings are introduced in detail. The approaches to constructing graphs for detection and generation tasks are illustrated in the last subsection.

In section 0, two universal neural network structures are conducted with the graph representations of two tasks respectively. Two network structures are not specially designed for certain datasets but can be embedded as backbones in related research. The insight of the network structure designing is the notion of discrete differential geometry. Related mathematical concepts are also presented while introducing the neural network structures.

In section 3.3, the complete method of floorplan detection based on the proposed network backbone is illustrated in detail. A clear definition of the detection task is provided as well as the initial features of the graph representation of the floorplan CAD drawings. After that, a novel network, namely CAD-GATnet, based on the author's publication is introduced, including the network architecture as well as the objective functions of this task.

In section 3.4. the complete method of floorplan generation with constraints, namely Dualgraph2Plan is illustrated. In the problem definition part, the formulation of the problem and the constraints are interpreted in detail. The overall architecture of the Dualgraph2Plan, as well as its three modules, are introduced in detail with the following four subsections. Besides, the additional operations for other optional constraints in floorplan generation are illustrated in the last subsection.

Although this research is based on floorplan CAD drawings, the proposed method can be applied to other data with similar geometric features.

3.1 Features and graph-based representations

Floorplan CAD drawings are usually saved as vector graphics. Although the floorplan CAD drawings are converted into raster images and graph data during visualization and production, there are special features in floorplan CAD drawings compared with raster images and graph data. The author analyzes the geometric features of floorplan CAD drawings in this section. The difference between raster image representation and vector graphics representation are compared in section 3.1.1. Compared with generic graphs, there are additional geometric features in floorplan CAD drawings to describe the regularity patterns, which is introduced in section 3.1.2. Besides, geometric features, floorplan CAD drawings also hold some special topology features. The topology features including planarity and duality are illustrated in section 0. Based on such features, two ways to construct graphs for floorplans in the detection and generation task are illustrated in section 0.

Figure 10 Floorplans contain different levels of informantion in different architectual design stages. (a) Bubble diagram. (b) Sketches. (c) Room layout diagram. (d) Fine drawing floorplan. (e) As-built floorplan.

As mentioned in the introduction section, floorplans usually contained different levels of both geometric and semantic information. As is shown in Figure 10, a floorplan usually origins from a bubble diagram (a) during the visual thinking process (Arnheim 1997), in which types and relations are illustrated. More specific floorplans based on the bubble diagrams are concreted by architects with hand-drawing sketches (b). Once the room layout is determined, a simple room layout diagram (c) is drawn for exhibition and sale. Based on the room layout diagram, a fine drawing floorplan (d) is developed with further details, such as furniture and structural elements. Before the building construction, an as-built floorplan (e) is proposed with complete information on other professions besides architecture as well as rich semantic annotations.

Since the target floorplans are as-built CAD drawings (e) and room layout diagram (c) in floorplan detection and generation tasks respectively, the author focuses on these two floorplans during feature analysis. Although both two floorplans hold parameterization and regularity, there are still differences in planarity and duality features.

3.1.1 Parameterization of geometric primitives

The geometric primitives refer to the basic geometric patterns composing a complicated floorplan CAD drawing. Usually, there are three frequently used geometric primitives in floorplan drawings, i.e., segments, arcs, and ellipses, see Figure 11. A Circle can be considered as a special case of ellipses. These basic primitives make up architectural elements first. Then the architectural elements compose a complete floorplan.

These geometric primitives are usually represented as vector graphics during storage, in which each geometric primitive is a vertex, and the edges represent the relationships among the geometric primitives. Unlike other classic graph data, the attribution of elements in vector graphics usually contains positional information, which is used for illustrating the locations on the canvas. This representation is the so-called parameterization of geometric primitives.

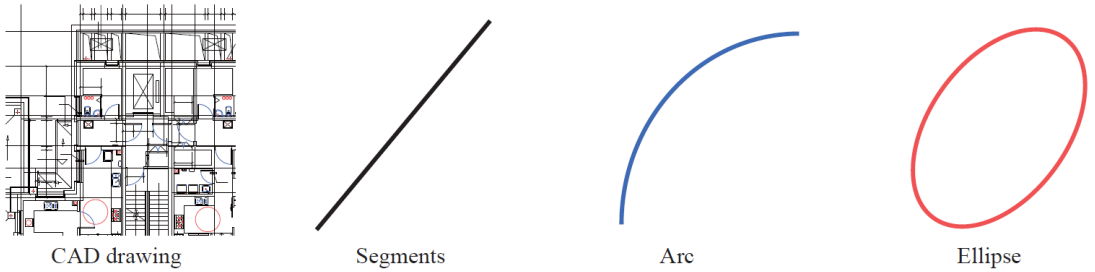


Figure 11 In one floorplan CAD drawing, architectual elements are usuallly composed of basic geometric primitives, such as segments, arcs and ellipses.

Compared to raster image representation, parameterization is a more suitable way for manufacturing. As is shown in Figure 12, a raster image sampling on the canvas with fixed pixel size gets the value with an indicator function. This representation of floorplan CAD drawings is discrete and explicit. Unlike the continuous texture in natural images, the common geometric primitives in CAD drawings shown in Figure 11, are usually 1D curves. In practice, the indicator function is replaced with an approximation with hat function bases, to have more continuous visualized results.

The advantages and disadvantages of the raster image representation are both obvious. Firstly, pixel-based image is naturally suitable for visualization since it requires no sampling stage and can be zoomed in and out with mature algorithms in image processing. Secondly, pixel images are Euclidean structure data, which holds good properties in deep learning, such as translation invariance, regular grids, and similarity on different scales. The 2D convolutional kernel function can be easily defined on raster images, from the perspective of discrete convolution or Gaussian integral. The translation invariance ensures the features filtered unchanged with convolutional kernel consistent in different positions. The similarity on different scales fits the pyramid structure of the images in different resolutions, which could enlarge the receptive field of the kernel during convolution and pooling layers in a convolutional neural network.

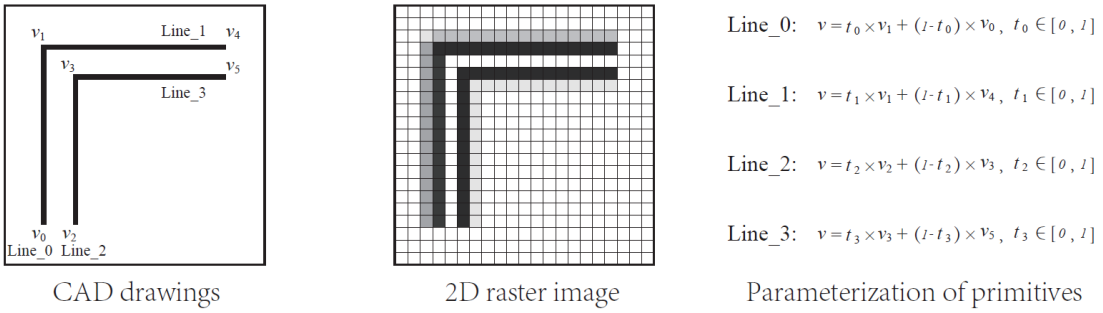


Figure 12 The architectual elements are represented with parameterization of geometry primitives in CAD drawings, while in 2D raster images, ones are represented with pixels.

The main shortage is the limited representation ability of vector graphics. Cases like overlapping primitives and primitives with large differences in scale are inappropriate with raster image representation. Firstly, the value on each pixel usually indicates whether the distance between the center of the pixel and the geometry primitives is lower than some certain threshold. The overlapping primitives have the same image as the single primitives. Representing overlapping primitives in different channels is also impossible. Usually, the number of channels is limited in a raster image, which can only represent limited close primitives. On the other hand, the channel for convolutional kernel definition has an order, which does not hold the permutation invariance for the overlapping primitives. Besides the geometric relationships are not suitable for explicit representation. For example, coincident primitives may have the same image as the close but not connected primitives. Meanwhile, it is hard to determine a threshold fitting primitive with a big difference in scale. Thus, information may also miss during zooming.

On the contrary, the representation of vector graphics is called parameterization. The accurate notion of parameterization can be found in differential geometry (Budroni and Bohm 2009). As is shown in the right column of Figure 12, all points on one 1-d primitive are mapped to $[0,1] \in R^1$. This function is a parameterization of one parameterization. Briefly speaking, all primitives are depicted with start and end points and additional parameters. For linear primitives, i.e., segments, no more additional parameters are needed. For arc and ellipse more parameters are used to depict the geometric information.

The parameterization is an implicit representation. All primitives are described as geometric objects. Primitives with the same type are depicted with the same template and different attributions. The limitations of raster images are improved with the parameterization representation. Since there is no sampling in parameterization, the limitations of resolution and overlapping issues are eliminated. Besides, the geometric constraints, e.g., coincident, tangent, orthogonal, and mirror, can be easily represented with the relationships of two objects.

However, there is no free lunch. The main issues brought by parameterization are also obvious. Compared with the regular grid in the data structure in raster images, the Euclidean structure does not exist in the parameterization of vector graphics. This type of data is called non-Euclidean data (Biljecki *et al.* 2015, Bronstein *et al.* 2017, Grover and Leskovec 2016). The convolution is hard to

define on non-Euclidean data. The expected convolution formulation should not only keep permutation invariance but also be integral to the geometric constraints.

In this research, parameterization of the CAD drawings is used and represented as graphs. Current networks designed for non-Euclidean data take undirected or bidirected graphs to depict CAD drawings, which lack the utilization of unique features in architectural floor plan data and similar datasets. To tackle the upper issues, the author improves the networks for general non-Euclidean data with the geometric features of the floorplans in specific applications. Novel networks with efficient convolution definitions are illustrated in section 3.3 and section 0 respectively. Compared with image-based methods, the proposed methods have obvious advantages, since the proposed method direct completes the tasks on CAD drawings with no data conversion steps. This hypothesis is also validated with abundant experiments in Chapter 0.

3.1.2 Regularity geometric constraints

There is an obvious difference between natural images and CAD drawings. CAD drawings are composed with simple geometric primitives, instead of complicated texture and illumination. In the architectural industry, floorplan CAD drawings also have special features compared with other CAD drawing files. Since most architectures in real projects are regular shapes, the geometry primitives are constraints with regular geometric relations.

Figure 13 A local floorplan CAD drawing without anonations.

As is shown in Figure 13, these regularity geometric constraints are especially abundant in as-built floorplans. For better illustration, a local floorplan of a residual building is clipped and remove all irrelevant elements except geometric primitives. All regularity geometric constraints in this floorplan CAD drawing are colored with different colors, which represents the geometric relations

between two primitives.

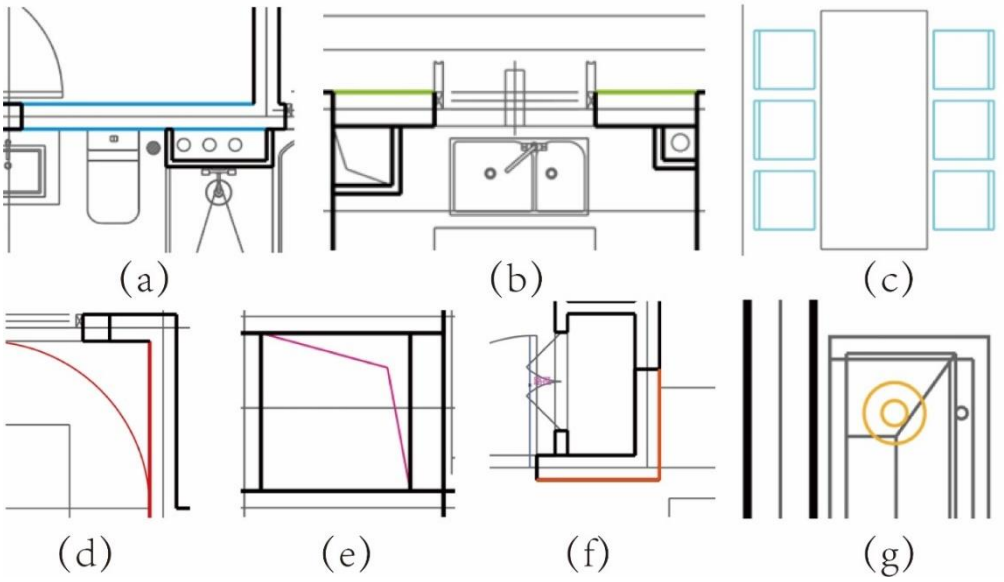


Figure 14 Geometric regularities in the floorplan CAD drawings. (a) Parallelism. (b) Co-linearity. (c) Duplication. (d) Tangentiality. (e) Coincidence. (f) Orthogonality. (g) Concentric.

The common regularity geometric constraints in this floorplan CAD drawings are enumerated in Figure 14, which would be carefully introduced as follows:

(a) Parallelism. Parallelism refers to two non-colinear line segments that share the same slope. This regularity geometric constraint is usually seen in patterns of walls, windows, and some rectangular furniture.

(b) Co-linearity. Co-linearity refers to two parallelism segments on the same line. This regularity geometric constraint is usually seen in the patterns of the opening of walls and array-placed architectural elements.

(c) Duplication. Duplication refers to two primitives with the same intrinsic features and different positions. This regularity geometric constraint is usually seen in the patterns of furniture, which is usually regularly recurring in floorplan CAD drawings.

(d) Tangentiality. Tangentiality refers to the tangent segment and arc or ellipse. This regularity geometric constraint is usually seen in the patterns of staircases, doors, and some furniture.

(e) Coincidence. Coincidence refers to two primitives sharing the same start point or end point. This regularity geometric constraint is usually seen in the patterns of two close primitives belonging to the same architectural element instance.

(f) Orthogonality. Orthogonality refers to two orthogonal line segments. This regularity geometric

constraint is usually seen in the patterns of walls and windows. Furthermore, the two orthogonal line segments are horizontal and vertical in most Floorplan CAD drawings.

(g) Concentric. Concentric refers to two arcs or ellipses sharing the same center and different radii. This regularity geometric constraint is usually seen in the patterns of furniture.

These geometry constraints are necessary for production and can also be seen in most CAD drawing software. Since the floating-point precision of computers has limitations, without restricted geometric constraints, the pattern will lose accurate position and may lead to errors. For example, without coincidence constraints, two lines may seem connected on a small scale but non-intersected when zooming out. Besides, as is described in the upper gist, the occurrence scenes of regularity geometric constraints imply semantic information.

Most regularity geometric constraints of two geometric primitives have transitivity so that the relationship of triple primitives can be also described with upper constraints. Meanwhile, as the content gets richer in the floorplans in different architectural design stages, the geometric constraints get more complicated. Since the geometric constraints depict the relationship of two primitives, the upper bound of the number of geometric constraints of $\boldsymbol{n}$ geometry primitives is $\boldsymbol{n^2}$ in theory. However, in real cases, most geometry primitives are only constrained with their adjacent geometry primitives, and the magnitude of the geometric constraints is much lower than the theoretical upper limit.

In the proposed research, these geometric constraints are integrated into the neural networks. All geometric constraints are embedded as edge features, such that relative position information can be fully used in the relevant tasks to improve performance.

3.1.3 Planarity and duality

Besides geometry features, there are also topology features in the floorplan CAD drawings. General graph data are composed of vertices and edges. Usually, the features are defined on the vertex space, and the edges are described with the adjacent matrix implying the topology of the graph.

In floorplan CAD drawings, there are more properties compared with the general graph data. For the topology features of as-built floorplans in the detection task and the room layout diagram floorplans in the generation task, two graphs are introduced to represent the topology of the floorplans.

3.1.3.1 Line graph

The notion of a line graph comes from research by Harary and Norman (1960). In the graph theory, the line graph of an undirected graph is another graph that represents the adjacencies between edges in the original graph, which is constructed in the following ways:

(a) for each edge in the undirected graph is represented with a vertex in the line graph,

(b) for every two edges have a vertex in common is the edge connected to the corresponding vertices in the line graph.

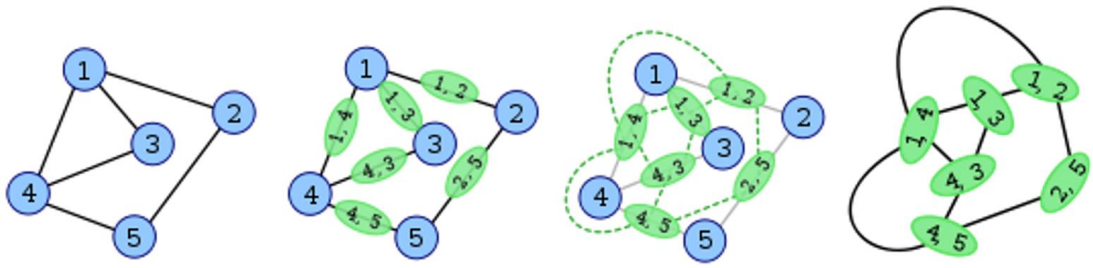


Figure 15 A illustration of the line graph, which the original edges are treated as vertices. Two vertices of line graph are connected if their original edges share a same vertex.

The construction process of the line graph is illustrated in Figure 15. The first graph is the original undirected graph, whose vertices are blue and labeled with 1 to 5. The second graph and the third graph show the vertex and edge construction from the undirected graph to its line graph, which is drawn with green color and dashed lines respectively. By following the construction rule, the edges in the undirected graph are represented as vertices in the line graph and labeled with their endpoint label. And the two vertices in the line graph are connected if their corresponding original edges share the same vertices. The last graph is the constructed line graph. The construction process of the line graph is irreversible, and some graphs are not line graphs. One thing worth mentioning is that the line graph should also be an undirected graph, the labels of the vertices in the line graph are labels set without an order. In practice, the features of the line graph need to carefully design to eliminate the directions.

Since all geometric primitives in floorplan CAD drawings are represented with two endpoints in parameterization representation, it is intuitively to construct a naïve undirected graph for the floorplan CAD drawings, by taking all endpoints as vertices and the primitives as the edges connecting them. Compared with this naïve undirected graph, its line graph is more suitable to represent the floorplan CAD drawings, since the parameters of the geometric primitives are defined on the edge space of the naïve undirected graph, instead of the vertex space, i.e., its endpoints. Thus, the line graph is used to represent the floorplan CAD drawings.

3.1.3.2 Planar graph and dual graph

Although most floorplans can be represented with line graphs, room layout diagrams have some better topology properties, which may improve the performance in generation tasks. This subsection introduces the notion of the planar graph and the dual graph to represent the room layout diagrams and the topology features.

In graph theory, a planar graph is a graph that can be embedded in a 2D plane. In other words, a planar graph can be drawn on a 2D canvas in such a way that its edges only intersect at their endpoints. Such a drawing is called a planar embedding of the graph. As is shown in Figure 16,

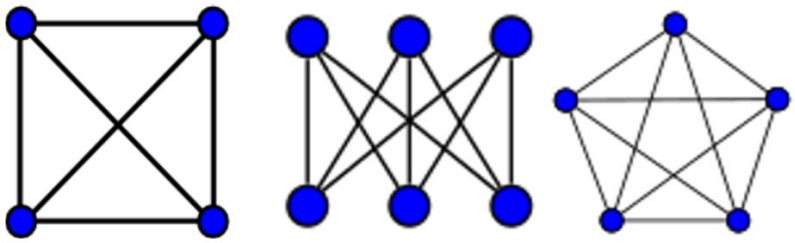


Figure 16 A brief illustration of the planar graph. A planar graph can be embedded on a plane with no additional intersection points on edges. A planar graph must have no K_{3-3}(middle) or K_5(right) as minor graph.

As is shown in Figure 17, a planar graph may have different planar embeddings. This is defined to distinguish the topology of different embeddings. The combinatorial embedding depicts the vertices of each face in the planar embeddings.

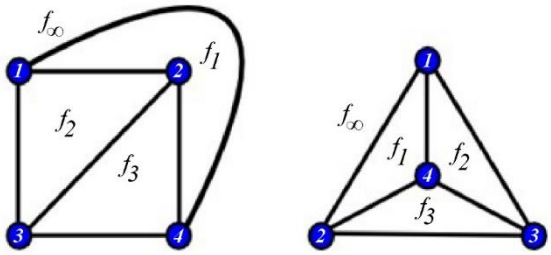


Figure 17 Two different embeddings of a same planar graph.

Take the two different planar embeddings for example, their combinatorial embeddings are shown as follows:

$$left \begin{cases} f_\infty : [1,4,3] \\ f_1 : [1,4,2] \\ f_2 : [1,2,3] \\ f_3 : [2,4,3] \end{cases} \qquad right \begin{cases} f_\infty : [1,3,2] \\ f_1 : [1,4,2] \\ f_2 : [1,3,4] \\ f_3 : [2,4,3] \end{cases}$$

The author first constructs the naïve undirected graph with the upper section. All endpoints of primitives are set as vertices and the line segments are set as edges in this graph. Compared with the as-built floorplan CAD drawings, walls are usually drawn with single lines instead of two parallel lines. Meanwhile, this naïve undirected graph naturally conforms to the definition of the planar graph, with a feasible planar embedding.

One of the topology features of the planar graph is very suitable to the proposed floorplan generation

tasks, namely dual graph. The notion of the dual graph in graph theory is defined as a corresponding graph of a given planar graph. By defining the planar graph as a 2-d oriented manifold, the vertices, edges, and faces are linear spaces with 0,1 and 2 dimensions. The definition of the dual graph is derived from taking the dual space of these linear spaces.

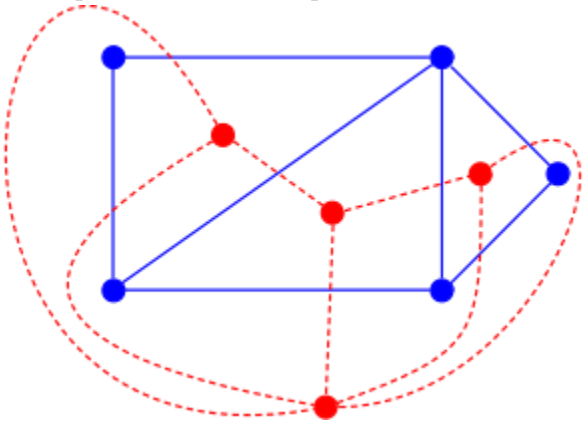


Figure 18 A illustration of a planar graph (blue) and its dual graph (red). Once the embedding of a planar graph is fixed, its dual graph is also fixed, by taking the faces in planar graph as dual vertices and add corresponding dual edges.

As is shown in Figure 18, the planar graph and the dual graph is drawn in blue and red. Unlike the line graph, the construction of a dual graph is bidirectional since the duality is a one-to-one corresponding mapping. The prime graph (blue) and the dual graph (red) are used to represent the planar graph and the dual graph respectively. The faces, edges, and vertices in the prime graph are one-to-one mapped to the vertices, edges, and faces in the dual graph.

As mentioned in the introduction section, the floorplan generation task is usually asked to satisfy the given bubble diagram. Based on the definitions, the prime and dual graph is suitable to describe the bubble diagram and the room layout diagram. With such representations, the topology of the target floorplan is explicitly preserved during generation. Further details are introduced in the following sections.

3.1.4 Graph representation of floorplans

Based on the above geometry and topology features, two graph construction methods are introduced for as-built floorplans and room layout diagrams tackling the issues in the floorplan detection and floorplan generation tasks respectively.

3.1.4.1 Graph construction for as-built floorplans

As mentioned above, each primitive in the as-built floorplan CAD drawings is represented as vertices in the line graph. However, the original definition of the topology of the line graph is not enough in the as-built floorplans. With the consideration of the different regularity geometry constraints in as-built floorplan CAD drawings, a rule to construct the edges in the line graphs in

established.

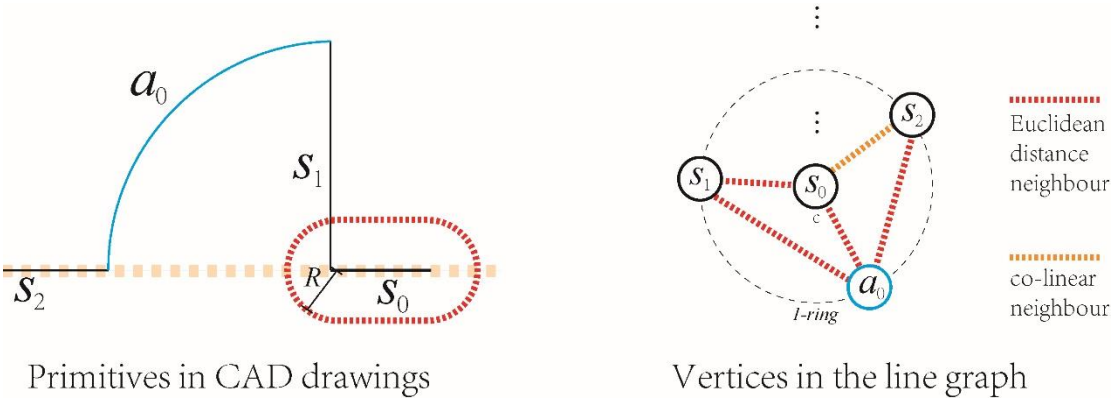


Figure 19 A graph of primtives in as-built CAD drawings is built with the notion of line graph. Since the edges has inconsistent attributions, this graph is a heterogenrous graph.

As is shown in Figure 19, a local part of the floorplan is taken for example to illustrate the edge construction in the line graph. s and a with indexes are used to represent the segments and arcs in the CAD drawings. s_0 is set as the center to show the structure in the line graph. All other primitives are the 1-ring neighbors and 2-ring neighbors of the center vertex.

Although the coincidence conforms to the original definition of the line graph, other regularity geometry constraints should also be considered in building graphs. For different regularity geometry constraints, a threshold R is designed for edge construction. Before edge construction, the author first defines the distance of two primitives in the CAD drawings:

$$d_{ij} = \min_{\boldsymbol{p}\in v_i, \boldsymbol{q}\in v_j} |\boldsymbol{p} - \boldsymbol{q}|.$$

In this research, the distance of two primitives is defined as the minimal Euclidean distance of the points in two primitives respectively. With such a definition, the radius of the receptive field can increase fast with the radius in graph growth.

As the primitives and edges have more than one type, here the author borrows the notion of heterogeneous graph provided by Sun *et al.*: A graph denoted as $\mathcal{G} = (\boldsymbol{\nu}, \boldsymbol{\varepsilon})$, consists of vertex set $\boldsymbol{\nu}$ and an edge set $\boldsymbol{\varepsilon}$. A heterogeneous graph is a graph, whose vertex type mapping function $\boldsymbol{\phi}: \boldsymbol{\nu} \rightarrow \mathcal{A}$ and edge mapping function $\psi: \boldsymbol{\varepsilon} \rightarrow \mathcal{R}$, where $|\mathcal{A}| + |\mathcal{R}| > 2$. This notion origins from the graph data in recommendation algorithms, in which not all vertices or edges are defined in a same feature space. Apparently, the segments, arcs, and ellipses are depicted with different parameters, the proposed line graph representation of the floorplan CAD drawing conforms to the definition of the heterogeneous graph. With such data structure, more related neural networks designed for heterogeneous graph can be taken as reference in floorplan detection and generation tasks.

3.1.4.2 Graph construction for room layout diagram

This section introduces the dual graph construction process tackling the room layout generation tasks with bubble diagrams describing the requirements of customers.

Although some masterpiece architectures are designed with fashion floorplans containing complicated topology, most of the common architectures pursuing the efficient and reasonable use of room space have relatively simple topology. Also, automatic approaches exactly tackle replacing human labor in architectural design with simple topology and regular shapes. Thus, three rational assumptions are taken in this research to limit the range of the generated room layouts.

- The first assumption is the Manhattan assumption (Furukawa *et al.* 2009), which assumes all walls in the floorplan are orthogonal since most buildings are man-made structures with strong regularities, such as adjacency, parallelism, and orthogonality.

- The second one assumes that each room is a simply connected polygon.

- The last one is that each partition wall of two adjacent rooms is a nonself-crossing C^0 continuous polyline.

The first assumption restricts the shape of the generated room layouts. The last two assumptions restrict the topology of targeting room layouts. Based on these three assumptions, the duality between the given bubble diagram and the target room layout diagram is introduced, as well as the dual graph construction process.

The author first gives the definition of a bubble diagram. As is shown in (a) of Figure 10, customers or designers usually use a bubble diagram to depict the expected room layouts with sketches, which illustrated the numbers, types, and relative positions of rooms. This research uses planar graph $\mathcal{G}_{\mathcal{B}} = (\mathcal{V}_{\mathcal{B}}, \mathcal{E}_{\mathcal{B}}, \mathcal{F}_{\mathcal{B}})$ to represent the given bubble diagram, illustrated as (d) in Figure 20, in which $\mathcal{V}_{\mathcal{B}}$ represents the vertex set of all rooms, $\mathcal{E}_{\mathcal{B}}$ represents the edge set of all rooms adjacency, and $\mathcal{F}_{\mathcal{B}}$ are face set of the planar graph without any specific semantic meanings, just for completing the planar graph definition.

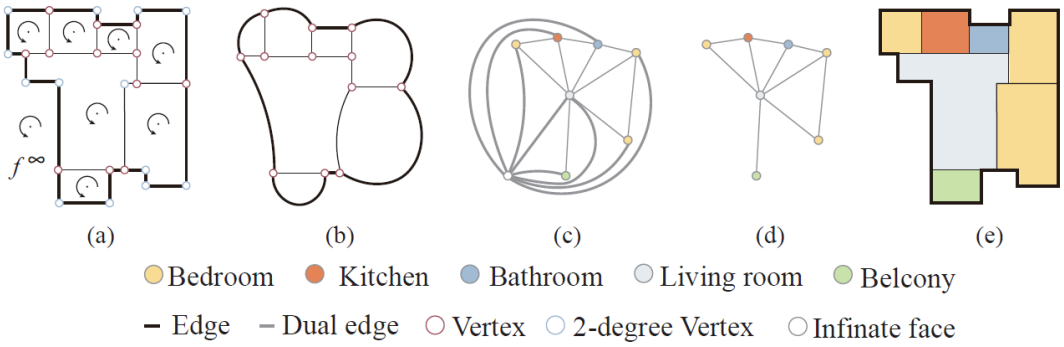


Figure 20 The dual graph construction of the given bubble diagram with the notion of the planar graph and the dual graph. (a) The wall graph of the target floorplan. (b) Edge collapses of the wall graph. (c) Wall graph's dual graph. (d) The given bubble digram. (e) The target room layout diagram.

With the above bubble diagram graph, the author keeps constructing the duality between the bubble diagram and the target room layout diagram. $\mathcal{G}_{\mathcal{L}} = (\mathcal{V}_{\mathcal{L}}, \mathcal{E}_{\mathcal{L}}, \mathcal{F}_{\mathcal{L}})$ is used to represent the target room layout with planar graph definition as previously mentioned, in which $\mathcal{V}_{\mathcal{L}}$ are endpoints of wall segments, $\mathcal{E}_{\mathcal{L}}$ represent walls, and $\mathcal{F}_{\mathcal{L}}$ represent all rooms in the target room layout. To complete the planar graph definition, the infinity face f_∞ is added to $\mathcal{F}_{\mathcal{L}}$. Since the orientation of $\mathcal{G}_{\mathcal{L}}$ has no special semantic meaning, the author just takes counterclockwise as the positive direction of $\mathcal{G}_{\mathcal{L}}$.

The duality construction process starts from the room layout $\mathcal{G}_{\mathcal{L}}$. The author first collapse edges (Garland and Heckbert 1997) until no edges whose endpoints are 2-degree vertices. $\mathcal{G}_{\mathcal{D}}$ is used to represent the graph after edge collapse, shown as (b) in Figure 20. Then, the dual graph of the $\mathcal{G}_{\mathcal{D}}$ is defined with dual graph definition in section 0 , and $\mathcal{G}_{\mathcal{P}}$ represents the dual graph, shown as (c) in Figure 20. It is easy to construct an injective function $\psi: \mathcal{G}_{\mathcal{B}} \mapsto \mathcal{G}_{\mathcal{P}}$ by mapping the room vertices of bubble diagram to the dual vertices of the faces in the room layout diagram. Since ψ is injective, $\mathcal{G}_{\mathcal{B}}$ and $\psi(\mathcal{G}_{\mathcal{B}})$ are isomorphism. Also, it is obvious that $\psi(\mathcal{G}_{\mathcal{B}})$ equals to $\mathcal{G}_{\mathcal{P}} \setminus \{v_\infty\}$.

Based on the upper duality construction process, the notion is given that generating room layout diagrams with given bubble diagrams, is exactly the inverse direction of the duality construction. Further details are illustrated in the following sections.

3.1.5 Summary

This section primarily outlines the theoretical contributions about data representation of this research. Starting from an analysis of the geometric and topological characteristics of floorplan CAD drawing data, it further employs more specialized graph structures to describe CAD drawings. The specific points of innovation are as follows:

- Geometric and topological features of floorplan CAD drawings are decoupled and inducted separately.

- Line graph and planar graph can more specifically describe floorplan drawings with their

features in detection and generation tasks.

- The geometric features between geometrical primitives are utilized to construct line graphs, thereby adding regularization terms.

- The topology property, i.e., duality relationship between bubble diagrams and floorplans, is utilized to establish the initial topology for generation.

3.2 Corresponding backbone structures

With the consideration of the line graph and the planar graph, two network structures are proposed to embed the features defined on different feature spaces. Both two network structures are the improvement with the features of floorplan CAD drawings. Classic graph neural networks are usually designed the tasks (Wu *et al.* 2020), e.g., vertex prediction or link prediction, which are especially tackling vertex space or edge space only. In the floorplan detection and generation tasks, the expected networks should be able to process the vertex and edge tasks at the same time.

The convolution formulation defined on graph data is introduced in this subsection. Unlike the spatial definition of convolution kernel on raster images, the convolution defined on graphs is a little bit abstracted. Here, the author briefly introduces the graph convolution definition from the perspective of matrix spectral decomposition.

The core idea of the graph convolution definition is to approximate the continuous Laplacian operator on graphs.

The continuously convolution of two function f and g is defined as:

$$(\mathrm{f} \star g)(\mathrm{x}) = \int f(y)g(x-y)dy,$$

Meanwhile, there is also a conclusion in Fourier transformation:

$$(f \star g)(x) = \mathcal{F}^{-1}\left(\mathcal{F}\big(f(x)\big) \circ \mathcal{F}\big(g(x)\big)\right).$$

Therefore, with the Fourier transformation defined on graphs, the convolution can be defined on graphs. Here the definition of continuous Laplacian operator is defined as:

$$\Delta f(x) = \frac{\partial^2 f}{\partial x^2}.$$

Discrete Laplacian is exactly the sum of the difference between the vertex and its 1-ring neighbors. The discrete Laplacian operator L on graph is defined as:

$$L = D - A ,$$

where D is the degree matrix and A is the adjacency matrix of the graph. Usually, to keep magnitude at a consistent level, the symmetric normalized Laplacian operator L_N is used for graph convolution, which is shown as:

$$L_N = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} .$$

Since the bases of Fourier transformation are exactly the eigenvectors of the Laplacian operator:

$$\Delta e^{2\pi i x \cdot v} = \lambda e^{2\pi i x \cdot v} ,$$

the Laplacian operator L_N on graph can has an eigen decomposition like:

$$L_N = U \lambda U^T,$$

where U and λ are eigen vectors and eigen values of the Laplacian matrix respectively. Thus, the Fourier transformation on graphs can be defined as:

$$\mathcal{F}_{\mathcal{G}}(x) = U^T x.$$

The inverse Fourier transformation on graph can be defined in the same way:

$$\mathcal{F}_{\mathcal{G}}^{-1}(x) = U\mathrm{x}.$$

Further the convolution on graph can be written as:

$$f \star g = U(U^T f \circ U^T g) .$$

To eliminate the high computation complexity of eigen decomposition of the Laplacian. The convolution kernel is approximated with Chebyshev polynomial:

$$f_{kernel} \star g \approx \theta \left(\mathrm{I} + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right) \mathrm{g}$$

$$= \theta \left(\widetilde{D}^{-\frac{1}{2}} \tilde{A} \widetilde{D}^{-\frac{1}{2}} \right) \mathrm{g} ,$$

where:

$$\tilde{A} = A + I, \widetilde{D}_{ii} = \sum_j \tilde{A}_{ij}, \theta \text{ are coefficients.}$$

With the nonlinear activation function σ in the neurons of the neural networks, the final

mathematical expression of one graph convolution layer is:

$$H^{l+1} = \sigma\left(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}H^{l}\theta^{l}\right).$$

From the perspective of the interpolation and the spectral decomposition, in the vanilla version of the graph convolution approximate the target function with the Fourier basis of the graph. The weights in each layer of graph convolution could be considered as the coefficients of the basis function, which would be optimized during the training process. Meanwhile, the first-order approximation of the Laplacian decomposition is used in real applications, the structure of the vanilla version of graph convolution could also be regarded as a variation of the message passing neural network (Gilmer *et al.* 2017), from the spatial perspective. More details about the formula derivation of graph convolution can be found in (Kipf *et al.* 2016).

The proposed following improvement on neural network structures for detection and generation tasks attempts other suitable approximations of the decomposition of the Laplacians in the spectral perspective and alternative ways to aggregate the features in the spatial perspective so that the features defined on different spaces could be fused during convolution and solve the prediction tasks on vertices and edges in one network.

3.2.1 Attention structure for line graphs

The main issue of conducting network structures for the line graph of as-built floorplan CAD drawings is to design a proper convolution formulation considering the features of geometric primitives and the geometric constraints.

In general graphs, weights on edges are equivalent, such that the discrete Laplacian operator L can be written as $L = D - A$. However, as is mentioned in section 3.1.4.1, the line graph built from floorplan CAD drawings is a heterogeneous graph, in which weighs on edges have different types. Since different relative position relationships among primitives represented with geometry constraints defined on edges of the line graph imply semantic information of CAD drawings, it is intuitive to align different edges with different weights. In this research, a network is designed in which the discrete Laplacian operator L is approximated with a learnable nonlinear function based on the multi-head attention layer in the vanilla transformer network structure.

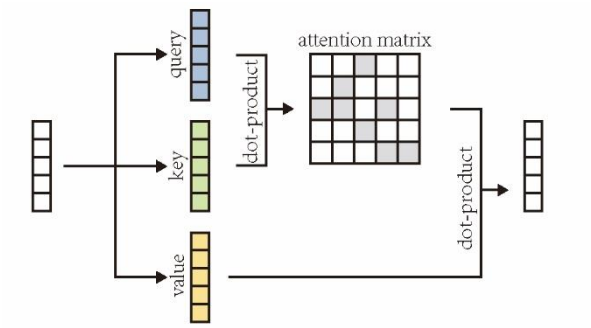


Figure 21 Illustration of the single-head attention layer.

The vanilla transformer is a fashion neural network structure (Vaswani *et al.* 2017). Once it was proposed, the multi-head attention layer in the transformer shows great power in natural language processing and computer vision tasks. As is shown in Figure 21, the structure of the single-head attention layer is illustrated. The vertex features are first mapped to three different feature spaces, namely query, key, and value spaces, with three linear layers respectively. The attention matrix with masks approximates the weight on the edges with the product of the query and key. To keep the output feature in a consistent magnitude, the attention matrix is normalized with the SoftMax function:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}.$$

The mathematical expression of one single-head attention layer is shown as:

$$\mathrm{V}^{\mathrm{l}+1} = \mathrm{softmax}\left(\frac{\mathrm{QK}^{\mathrm{T}}}{\sqrt{\mathrm{d_k}}}\right)\mathrm{V}^{\mathrm{l}}.$$

The parameter $\mathrm{d_k}$ is the dimension of the vertex features. The dimension of each entry in the attention matrix is the number of the head. Therefore, the multi-head attention layer is improved from the single-head attention layer by splitting Q and K and calculating the attention matrix with more than one dimension.

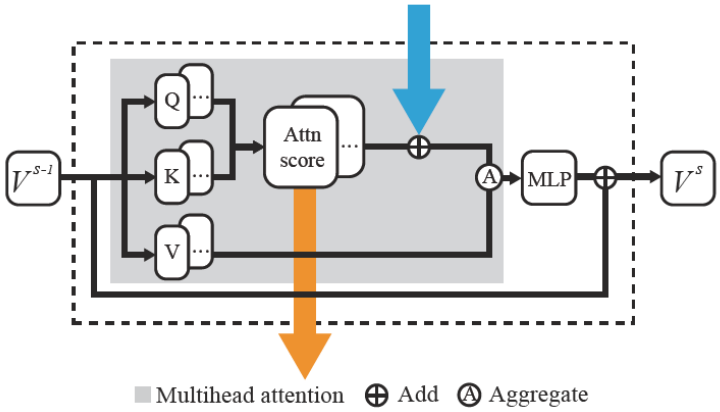


Figure 22 A novel network structure for the line graph of the as-built floorplan CAD drawings in floorplan detection, which is improved from the vanilla transformer structure.

The proposed insight is to integrate the edge features to vertex features based on the structure of the multi-head attention layer to approximate the discrete Laplacian operator on the heterogeneous graph. The main structure of the proposed network is shown in Figure 22. As is shown in the blue arrows in Figure 22, the edge features are mapped to a feature space with the same dimension as the number of the attention heads and then added these edge features with the attention matrix in the multi-head attention layer. The attention matrix is passed as edge features of the next convolution layer, shown as the yellow arrow in Figure 22. The mathematical expression of the proposed improved graph convolution layer based on multi-head attention is shown as:

$$V^{l+1} = \left(\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) + E^l\right) V^l .$$

The theory support of the proposed improved attention-based network structure comes from several aspects.

Firstly, adding edge features with the attention matrix is reasonable. Since the edge feature and attention score are one-to-one corresponding to describe the relationship between two vertices, adding or concatenation is an effective operation to combine these two features.

Secondly, from the perspective of graph convolution, the proposed improvement does not break the mechanism of graph convolution. During calculating the attention matrix, a mask is added such that one vertex only focuses on its 1-ring neighbors on the line graph. In one layer, neighbor features are aggregated to the center vertex. With the number of layers going deeper, the receptive field of the convolution gets larger, so that the context at both local and global levels is embedded with the proposed attention layers.

Thirdly, from the perspective of the transformer network, the mask designed according to the adjacent matrix is very reasonable. Although the vertices in the proposed line graph are not a sequence with a certain order or stable relative positions, the location information is already

embedded in the vertex features. Since the structure of the transformer is permutation invariance, the proposed vertex features can be analogy of the line graph with the long text sequence after adding positional embeddings. According to facts in related research on the transformer structure used for long sequences (Dai *et al.* 2017, Beltagy *et al.* 2020, Zaheer *et al.* 2020), the normalized attention matrix is usually a diagonally dominant matrix. In other words, compared with vertices from a large distance, a vertex would more likely focus on its nearby vertices. Thus, theoretically speaking, masking vertices excepting the 1-ring neighbors does not harm the performance of the multi-head attention structures.

3.2.2 Feature fusion module for planar graphs

Compared with the feature definition in the line graph, the definition of the features in the planar graph is more complicated. Line graph only has feature defined on the edges and vertices, the faces in planar graphs also contain semantic information. In the room layout diagram generation task, the main issue is how to aggregate the features defined as vertex, edge, and face space. To solve this problem, the author designs a novel basic network structure with the notions from the discrete differential geometry.

Before presenting the proposed network, the author first briefly introduces the notions and some operators in discrete differential geometry. In mathematics, a simplicial complex in R^2 called a chain is a set composed of points, line segments and triangles. In an oriented 2d manifold, a k-dimensional simplicial complex is a vector with {0, 1, -1} in the k-dimensional space, namely chain space. The boundary operator is a function mapping a k-dimensional simplicial complex to its boundary, which is a (k-1)-dimensional simplicial complex. Since vertex, edge, and face spaces are linear spaces, the boundary operator can be written as a boundary matrix $\partial_k = \left(\left[\sigma_i^{k-1}, \sigma_j^k\right]\right)$, where:

$$\left[\sigma_i^{k-1}, \sigma_j^k\right] = \begin{cases} +1 & +\sigma_i^{k-1} \in \partial_k \sigma_j^k \\ -1 & -\sigma_i^{k-1} \in \partial_k \sigma_j^k \\ 0 & \sigma_i^{k-1} \notin \partial_k \sigma_j^k \end{cases}$$

The k-chains and their boundary operators are illustrated in Figure 23.

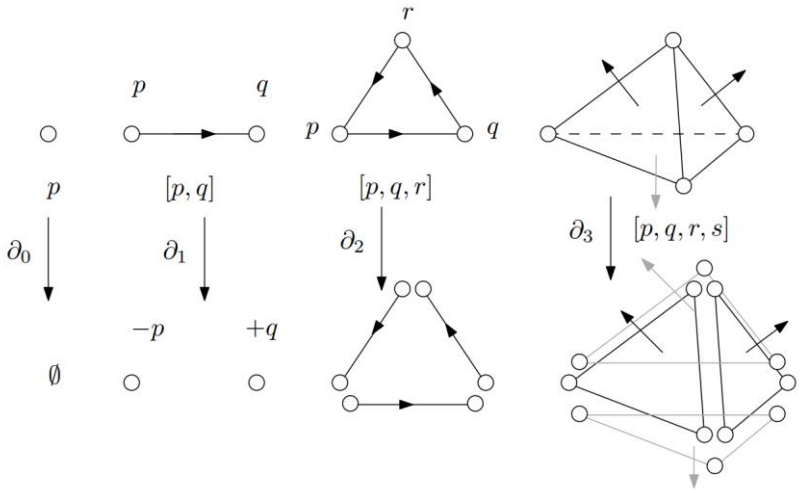


Figure 23 A illustration of 0,1,2,3-chain, and the responding boundary operators.

Also, the co-chain spaces of the vertex, edge, and face spaces are the functions defined on these chains. These co-chain spaces of the planar graph are also called a k-form. A coboundary operator or k-differential form is defined as:

$$\delta_k \omega \coloneqq \omega \circ \partial_{k+1}, \omega \in C^k(\Sigma, Z)$$

a function mapping a k-form to its coboundary, which is a (k+1)-form. In the same way, the coboundary operators also can be written as matrixes. The illustration of 1-cochain is shown in Figure 24.

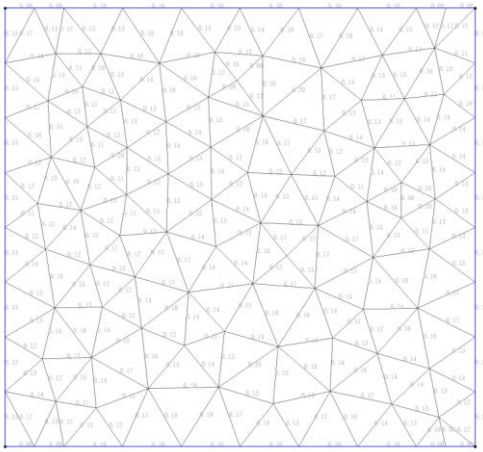


Figure 24 A illustration of a 1-cochain, which is a function defined on edge space.

Besides boundary and coboundary operators, Hodge starts operators are also linear operators that convert the feature spaces. Let $\mathcal{M}$be a n-dimensional Riemannian manifold. We can locally find oriented orthonormal basis $\left\{\frac{\partial}{\partial x_i}\middle| i = 1,2,\dots,n\right\}$ of vector fields and let $\{dx_i | i = 1,2,\dots,n\}$ be the

dual 1-form basis. A Hodge star operator $\star := \Omega^k(M) \mapsto \Omega^{n-k}(M)$ is a linear operator mapping k-form to its dual spaces with n-k dimensions is defined as:

$$\star\left(dx_{i_1} \wedge dx_{i_2} \wedge \ldots \wedge dx_{i_k}\right) = (-1)^{\sigma} dx_{i_{k+1}} \wedge dx_{i_{k+2}} \wedge \ldots \wedge dx_{i_n},$$

where $\sigma = (i_1, i_2, \ldots, i_n)$ is a permutation of the dual 1-form basis.

With the above three operators in discrete differential geometry, the author proposed a novel graph convolution network structure for the planar graph and its dual graph in the room layout diagram generation task.

Considering the predicted topology and geometry information in the bubble diagram and the room layout diagram, the author proposes a diagram composed of the different chain spaces and the related operators to illustration the conversion process, which is shown in Figure 25. In the proposed formulation of the room diagram generation task, the task needs to classify the dual edges and predict the locations of dual vertices. Thus, these two chains need to be transferred so that the features defined in their spaces can be aggregated. As is shown in Figure 25, boundary operator ∂ maps the prime face to the prime edges in the bubble diagram and the dual edge are one-to-one corresponded to the dual edges in its dual graph according to the duality construction in the section 0. Meanwhile the boundary operator of dual edges is exactly the transpose of the ∂.

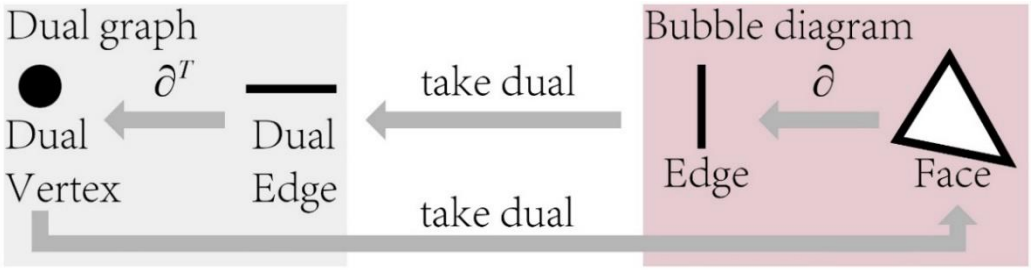


Figure 25 A brief illustation of the operators mapping the features difined on different spaces of the bubble diagram graph and its dual graph.

Since the features on the vertex, edge, and face spaces of a graph can be considered as the functions defined on such spaces, the features also could be considered as the cochains of the planar graph. Based on the upper chain conversions diagram, this process could conduct the conversions of the cochains respectively. As is shown in Figure 26, the dual vertex features V^l are mapped to the dual edge space, add it with the dual edge features E^l:

$$E' = E^l + dV^l,$$

and finally, map the features back to dual vertex space V^{l+1}:

$$V^{l+1} = \sigma_0\left(d^T \sigma_1\left(E' W_1^l\right) W_0^l\right),$$

in which $d \in \{-1,0,1\}^{|E|\times|V|}$ is the combinatorial differential matrix of the dual graph, mapping features on the dual vertex space to the dual edge space:

$$d_{i,j} = \begin{cases} -1 & if\ v_j\ is\ the\ startpoint\ of\ e_i, \\ +1 & if\ v_j\ is\ the\ endpoint\ of\ e_i, \\ 0 & otherwise. \end{cases}$$

Coboundary matrix d is exactly equal to ∂^T, and $\star_0^{-1}$ and $\star_1$ are two learnable Hodge star operators mapping prime faces and dual edges to them dual spaces.

The proposed insight of the network structure is replacing the symmetric normalized Laplacian operator L_N in vanilla graph convolution structure, with a Laplacian decomposition in discrete differential geometry (Smirnov and Solomon 2021, Desbrun *et al.* 2005):

$$\mathcal{L} = \star_0^{-1} d^T \star_1 d.$$

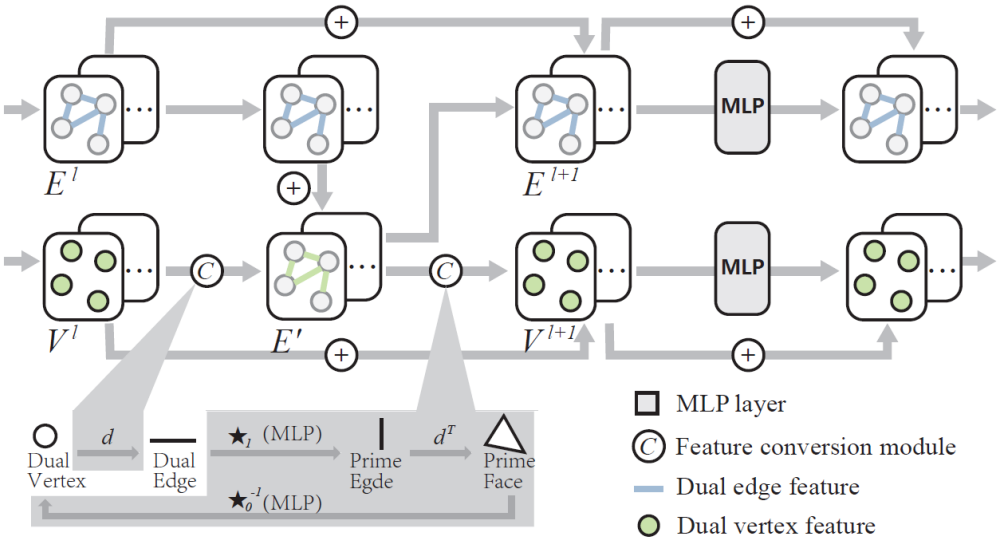


Figure 26 A novel structure for feature fusion in the floorplan generation problem.

3.2.3 Summary

This section primarily explains how to derive the convolution definition for graphs in the upper section, from the general graph convolution definition. Due to varying weights on the edges in line graphs and plane graphs, distinct network structures are designed to learn these weights, resulting in a better approximation of the decomposition of the Laplacian matrix. The specific points of innovation are as follows:

- Transformer structure concatenated with regular terms are used to as the learnable weights matrix for line graph for vertices aggregation.

- Boundary operators are used for topology embedding for planar graphs, inspired by the definition of discrete exterior differential operators.

3.3 Floorplan detection

Computer-aided design (CAD) is the use of computers to generate digital 2D or 3D illustrations of a product, aiding the creation, modification, analysis, or optimization process during designing and manufacturing. This technology has been widely used in modern architecture, engineering, and construction (AEC) industries. The CAD drawings usually convey accurate geometry, rich semantics, and domain-specific knowledge of a product design, with basic geometric primitives, such as segments, arcs, circles, and ellipses.

Spotting and recognizing symbols from CAD drawings is the first step towards understanding their content, which is crucial to many real-world industrial applications. For example, building modeling with semantic has a growing demand in various architecture engineering areas such as pipe arrangement, construction inspection, and equipment maintenance. A floor plan usually contains complete details of a store in an orthogonal top-down view.

Traditional symbol spotting usually deals with instance symbols representing countable things, like tables, sofas, and beds. Following the idea by Kirillov *et al.* (2019), Fan *et al.* (2021) extended the definition by recognizing the semantic of uncountable stuff and named it ***panoptic symbol spotting***. Therefore, all components in a CAD drawing are covered in one task altogether. For example, the wall represented by a group of parallel lines was properly handled by Fan *et al.* (2021), which however was treated as background.

Large-scale dataset of high-quality annotations is the fundamental ingredient to recent advances in supervised methods with deep learning, e.g., ImageNet for image classification, COCO for image detection, and ShapeNet for 3D shape analysis. Existing datasets for symbol spotting on floorplans, i.e., SESYD and FPLAN-POLY, are either synthetic, or inaccurate, both with only a few hundred samples. Fan built the first large-scale real-world FloorPlanCAD dataset of over 10,000 floorplans in the form of vector graphics and provided line-grained panoptic annotations.

CAD drawings are composed of domain-specific items, which are usually represented by abstract symbols. Human perception of CAD drawings is usually a multi-modal cross-context reference process requiring strong domain-related knowledge. Meanwhile, the large intra-class variance and small inter-class dissimilarity of symbols make it a more challenging task for computers.

Representing a CAD drawing as a graph of primitives is an intuitive way to retain the property of vector graphics and has been proven effective for the semantic symbol spotting task. In this work, the author presents a novel graph attention network GAT-CADNet to solve the panoptic symbol spotting problem. The network achieved state-of-the-art performance and the proposed main contributions to this research are:

- The author formulates the instance symbol spotting task as a subgraph detection problem and solve it by predicting the adjacency matrix.

- The author explicitly encodes the relative relationship among vertices, using a relative spatial encoding (RSE) module, to enhance the vertex attention.

- The author treats the vertex attention as edge encoding for predicting the adjacency matrix and design a cascaded edge encoding (CEE) module to aggregate vertex attentions from multiple GAT stages.

3.3.1 Problem definition

In this subsection, the author borrows the notions from the related work in the field of computer vision and give a clear definition of the floorplan detection problem.

As mentioned in section 1.3.1, the requirements of floorplan detection in real applications are to predict the labels for each geometric primitive in an as-built floorplan CAD drawing and to cluster the primitives by the architectural elements. As the annotations and the layer structures in CAD drawing contains a lot of semantic information, floorplan detection is a multi-modal task.

Theoretically, all semantic and instance information is well-stored in a perfect CAD drawing. However, the problem is more complicated. The data from the proposed partner company are not in ideal condition. Firstly, in some samples, all annotations are hidden for protecting privacy. Secondly, since the architectural drawing review process in China only focuses on the architecture project itself, there is no clear standard naming method to normalize the CAD drawing files in real projects. Projects designed by different companies may have different naming methods. Some projects are subcontracted several times which may have chaos in the naming process. Further, instead of using full names, the annotations are sometimes replaced with meaningless code names in practice.

On the other hand, some projects with lower standards or built-in early time may only have the CAD drawings conversed from papery floorplan drawings. Thus, the information used as input is only the geometric primitives in the CAD drawings.

In the field of computer vision, classification usually refers to the task of predicting a proper label for a given sample. Further, semantic segmentation is the task that predicts labels for all pixels in an input sample. Instance segmentation is a detection task, which predicts the clusters of each countable instance in a sample and predicts the label of each instance. None of these three popular tasks fit the proposed floorplan detection task.

Here, the author introduces the notion from the field of optical character recognition (OCR), namely panoptic symbol spotting (Fan *et al.* 2022), which not only predicts the labels but also the instance IDs. As is shown in Figure 27, the recognition targets in panoptic symbol spotting (c) are composed of two parts, the countable instances (a) and the uncountable stuff (b). In the context of the floorplan detection task, the countable instances refer to recurring architectural elements, which could be

separated individually, such as windows, doors, and furniture. And the uncountable stuff usually refers to the architectural elements that cannot be separated into independent parts, such as walls and handrails.

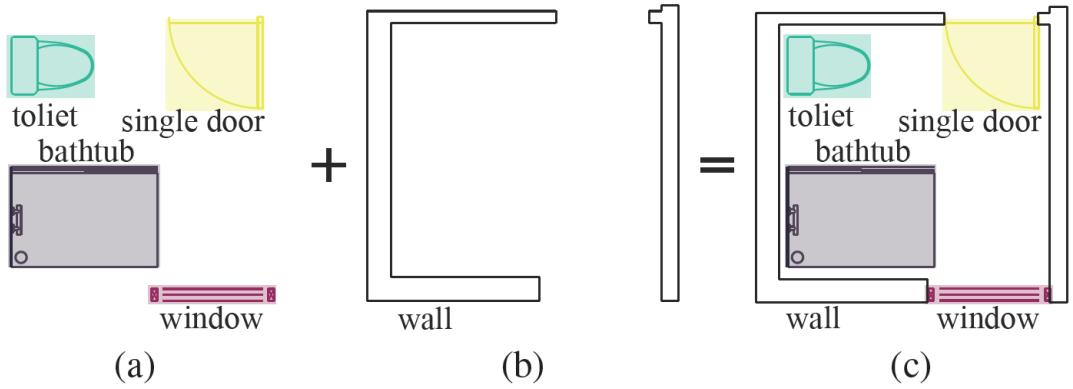


Figure 27 The author borrows the defination of panoptic symbol spotting problem (c) in the field of computer vision, which detects not only the countable instance (a) but also the uncountable stuff (b).

With the above panoptic symbol spotting problem definition and the demands in practice, a clear definition of the floorplan detection problem is given. The floorplan detection problem is defined as: given an as-built floorplan CAD drawing, taking the geometric primitives as input, predicting all labels of the primitives, and clustering the countable architectural elements.

3.3.2 Graph construction and feature definition

Due to the reasons from various aspects, the raw CAD drawings cannot be directly fed to the neural networks, this section introduces the data pre-processing step before the training step.

Firstly, the sizes of floorplans vary in a large range with the different types of architecture. Usually, the floorplan of a public architecture, *i.e.*, a shopping mall or gallery, may be much larger than the floorplans of a residual building. Meanwhile, the number of geometric primitives is a positive correlation with the size of the floorplan. The floorplans in the dataset should be split to unify the size of the samples.

In this task, the author uses the heterogeneous line graph to represent the as-built floorplan CAD drawing samples, and the graph is constructed following the method in section 3.1.4.1. Then the definitions are provided for the initial features of the vertex and edge spaces.

Vertex features. As is mentioned in Figure 11, all geometric primitives are linear, i.e., line segments, arcs, and ellipses. However, it is not proper to directly use the endpoints as the initial vertex features. Since the directions of these linear primitives, the polarity of the features should be eliminated. In other words, the polarity that comes with the order of the endpoints should be eliminated. A commonly used approach, in which both directions are used as input and eliminates the polarity

with symmetric operations, such as addition, is not suitable for this task, since this approach will double the size of the vertices features spaces, which may exceed the memory. Thus, the author proposes another approach to define the linear primitives without directions.

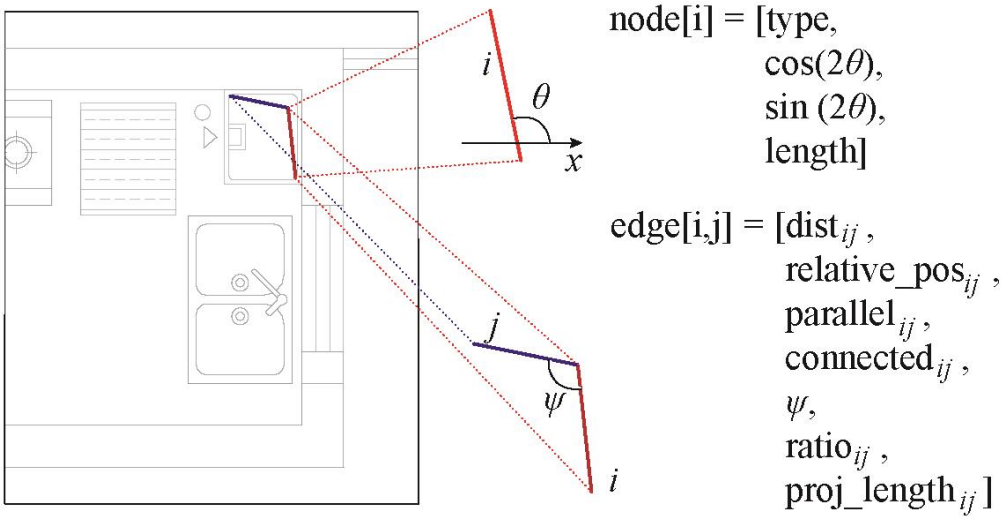


Figure 28 The initial feature defination of the line graph of the floorplan before detection.

Since the circle could be considered as the special case of an ellipse, all primitives in CAD drawings are classified with three type vertices. For the line segments, segments are embedded as:

$$\boldsymbol{v_{seg}} = [cos(2\theta_i), sin(2\theta_i), l_i],$$

where $\theta_i \in [0, \pi)$ is the clockwise angle from the x positive axis to $\boldsymbol{v_{seg}}$, and l_i measures the length of $\boldsymbol{v_{seg}}$. The proposed direction features are continuous when θ jumps between 0 and π. Here we double the angle θ to eliminate the direction of the line segment. The intrinsic feature of segments is taken, i.e., length, for feature embedding.

By referring the representation of arcs in SVG format, arcs are described as:

$$\boldsymbol{v_{arc}} = [cos(2\theta_i), sin(2\theta_i), l_i, r_x, flag_{large}, flag_{sweep}],$$

where $\theta_i \in [0, \pi)$ is the clockwise angle from the x positive axis to the chord of $\boldsymbol{v_{arc}}$, l_i measures the length of the chord of v_{arc}. Because an arc is part of an ellipse, we use r_x to measure the angle between the long axis and the x positive axis. Following the standard format of SVG files, $flag_{large}$ and $flag_{sweep}$ are used for determining arcs in four possible cases, i.e., the combination of superior arc and inferior arc, the clockwise and counterclockwise direction.

For ellipses and circles, we use:

$$\boldsymbol{v_{elli}} = [cos(2\theta_i), sin(2\theta_i), r_{long}, r_{short}],$$

where $\theta_i \in [0, \pi)$ is the clockwise angle from the x positive axis to the long axis of $\boldsymbol{v_{elli}}$, r_{long} and r_{short} refer to the length of the long axis and the short axis respectively. And circles are considered as the degradation of ellipse, in which the θ_i is set as 0, and r_{long} is set equally to r_{short}. Overall, the proposed approaches could precisely define each geometric primitive. The vertex features are continuously defined in three different feature space. Also, the absolute coordinate of segment center is not used for positional embedding since they do not have the translation invariant.

Edge features. The edge features $\boldsymbol{e_{ij}}$ is defined as:

$$\boldsymbol{e_{ij}} = [\boldsymbol{\delta_{ij}}, \psi_{ij}, r_{ij}, \parallel_{ij}, \perp_{ij}, \neg_{ij}, T_{ij}].$$

The feature of one edge describing the relationship of two geometric primitives is composed of two parts, relative positional relations, and regularity geometric constraints.

In the first part, parameter δ_{ij} refers to the coordinate difference of two centers of primitives. Parameter ψ_{ij} is the angle formed by two adjacent primitives in the graph. Parameter r_{ij} is the length ratio between v_i and v_j. These three parameters describe the relative positional relations of two adjacent primitives.

The second part of the edge feature represents the regular geometry constraints with the rest of the parameters. As is mentioned in section 3.1.2, during architectural CAD drawings production, topology constraints, e.g., coincidence and orthogonality, play an important role as well as the geometric primitives. As these constraints are usually the relationship between two primitives, undirected graphs are used to describe the CAD drawings and the edge features to embed the relationship between every two primitives. With the consideration of instances are usually drawn with strong regularity in CAD drawings, we add $\parallel_{ij}$,$\perp_{ij}$, $\neg_{ij}$, T_{ij} as extra features to enhance the edge feature. These four parameters are set as 0 or 1 to indicate parallelism, orthogonality, coincidence, and tangentially. Most other complicated regular geometry constraints can be represented with the combination of such indicator parameters.

Although this relationship could be inferred, only with the node feature, the neural network module could just reach an approximation of the transition function. Usually, feeding the network with explicit regularity terms on edge feature space would make network reach better convergence.

3.3.3 Architecture of the CAD-GAT network

In this section, the formulation of the proposed solution to the floorplan detection problem are introduced as well as the architecture of the structure of the proposed CAD-GAT network.

Problem formulation. In the panoptic symbol spotting problem, the label prediction part could be considered as a vertex prediction task to classify all vertices in the line graph of the floorplan CAD drawings. However, there is no proper task defined on graphs to cluster the countable architectural elements. An intuitive idea is predicting an offset vector for each primitive vertex in the CAD drawings, and figuring out the instance information with classic cluster algorithms. A naïve

approach to offset vector supervision is using the vector from the center of the primitive to the center of the instance (Fan *et al.* 2022). However, the shortage is obvious, since some instances may have closed centers which may cause errors in the clustering process.

In this research, the clustering problem is formulated as a subgraph detection problem. In the proposed graph-based representation, the architectural elements are exactly the subgraphs of the whole line graph of the complete floorplan CAD drawing sample. Both the countable instances and the uncountable stuff could be formulated as the subgraphs of the line graph. Further, the target instance segmentation results could be considered as the union set of all subgraphs corresponding to the instances and stuff. This union set is also a graph, in which all instance clusters are not connected. Thus, to figure out all instances in the floorplan, we only have to predict the adjacent matrix of the set graph mentioned above.

So far, the subgraph detection problem is formulated as an adjacent matrix prediction problem. Although the adjacent matrix is composed of 0 and 1, the adjacent matrix prediction problem is converted as a 0-1 classification problem. With the predicted adjacent matrix, all instances and stuff can be figured out by finding all unconnected subgraphs.

This approach perfectly solves the shortage in the method using classic cluster algorithms, since the formulation of the instance segmentation is based on the adjacent matrix, which is purely a topology method and does not need to consider the parameters for clustering algorithms.

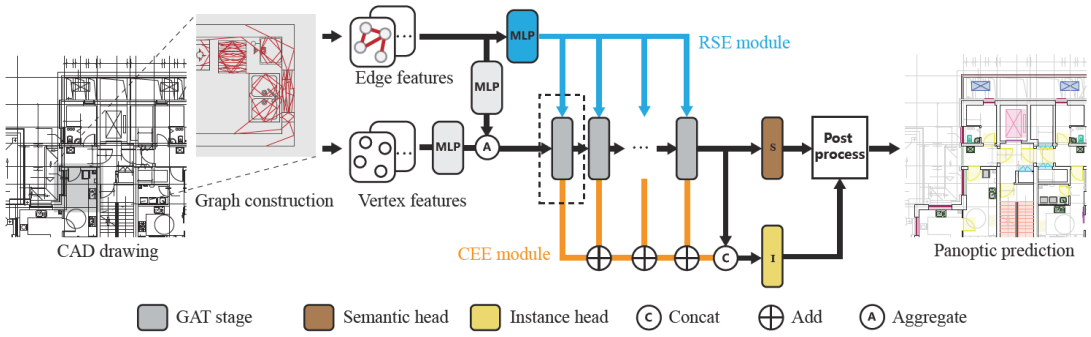


Figure 29 The network architecture of the CAD-GATnet for floorplan detection.

Overall architecture. Based on the above formulation, novel method is proposed, namely CAD-GATnet, which can solve the panoptic symbol spotting the problem in one neural network. As is shown in Figure 30, the overall architecture of the CAD-GATnet has a main trunk which is composed of several graph attention (GAT) layers in section 45 and Figure 22 as the backbone and two branch modules namely relative spatial encoding (RSE) module and cascaded edge encoding (CEE) module.

The raw floorplan CAD drawing data are processed into the line graph $\mathcal{G}$ with the initial vertex feature V and the edge feature E with the method in section 3.1.4.1 and section 0 respectively.

The three different feature spaces in the initial vertex feature V are firstly mapped into the same space R^{128} with three different shared Multi-Layer Perceptron (MLP) layers. Meanwhile, the initial edge feature E are also be mapped into R^{128} with another MLP layer. The RSE module embeds the edge feature and feeds them into each GAT layer and the CEE module collected the attention scores in each GAT layer for adjacent matrix prediction. The operations of the vertex feature and edge feature are illustrated in section 45. The embedded vertex features are fed to a semantic classification head to predict the labels for each primitive, which contains a fully connected layer mapping 128-dimensional vertex to the R^{class} , and a Softmax layer:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

to normalize the probability distribution of the label predictions. Meanwhile the edge features from the CEE module are also fed into another 0-1 instance classification head mapping the edge features into R^1, and a Sigmoid layer:

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1} = 1 - S(-x).$$

mapping the results to $[0,1]$. A post-process stage is added to obtain the final panoptic prediction results, which is composed of two parts. The first part is a sample method traversing the graph of the predicted adjacent matrix to figure out all unconnected subgraphs, which are the instances in the line graph. The second part combines the labels predictions, and the instance segmentation results to obtain the results.

The loss functions in the upper two heads are Cross Entropy, which is frequently used loss function in classification problems. The Cross Entropy expression of 0-1 classification in the instance head is shown as:

$$Loss = \frac{1}{N}\sum_i L_i = \frac{1}{N}\sum_i -[y_i \cdot log(p_i) + (1-y_i) \cdot log(1-p_i)],$$

where y_i represents the label of the primitive i, 1 for positive and 0 for negative, and p_i represents the predicted probability of the primitive i.

And the expression of multi-tag classification is shown as:

$$\boldsymbol{Loss = \frac{1}{N}\sum_i L_i = -\frac{1}{N}\sum_i \sum_{c=1}^{M} y_{ic}\, log(p_{ic}),}$$

where M is the constant number of the classes and y_{ic} represents the label of the primitive i of class c, 1 for positive and 0 for negative, and p_{ic} represents the predicted probability of the primitive i of class c.

It is worth mentioning that the cases of the entries in the predicted adjacent matrix can be classified into four types, according to the semantics labels and the instance relationship of two primitives. In the first case, two primitives belong to the same instance and have the same semantic label, whose entry should be 1. In the second case, two primitives belong to different instances and have different semantic labels, whose entry should be 0. In the third case, two primitives belong to different instances but have the same semantic label, whose entry should also be 0. In the last cases, two primitives belong to the same instance but have different semantic labels, which do not exist in real floorplan CAD drawings.

The distribution of the above three cases has a big difference in quantitative. Especially, the third cases have the least number but affect the results of the instance segmentation a lot. Thus, different weights are aligned for the upper three cases in this loss calculation. As is shown in Table 1, the weights for the first cases are set as 1, since they are the most seen cases in the dataset. The weights for the second cases are set as 2, which are less than the first cases. The weights for the third cases are set as 20, which are least cases in the dataset.

Weights	Same instance	Different instance
Same semantic label	1	20
Different semantic label		2

Table 1. Weights for different cases of the entries in the predicted adjacent matrix.

RSE module. When processing point cloud (Zhao *et al.* 2021) or natural language (Vaswani *et al.* 2017), researchers often enhance the vertex features with positional embedding to make the network invariant to translation and aware of the distance. According to the positional features and the embedding way, there are four common ways to obtain the positional embedding.

Positional features are classified into two types, the absolute positional embedding, and the relative positional embedding. The absolute positional embedding uses the absolute positions of each vertex which is frequently used in cases whose vertices' positions are fixed globally. The Relative positional embedding uses the relative positions of each vertex, which is usually used in cases whose vertices' positions are relatively fixed locally. Unlike the absolute positions, the relative positions encode the relationship between two vertices which is defined on edge spaces. Compared with absolute positional embedding, relative positional embedding is translation invariance which is more suitable for larger sequences and graphs with lots of vertices, since in these cases the attention mechanism usually focuses on local context.

Embedding ways also have two types, sinusoidal functions embedding and learnable functions embedding. The sinusoidal functions embedding alternating the value of a sine function and a cosine function, in which the dot product of the two positional embeddings is undirected and only relies on the offset distance. The learnable function embedding usually uses the learnable nonlinear functions encoding the initial position features. According to the related experiments, the sinusoidal functions embedding is suitable for 1-d sequences in natural language processing tasks, while the learnable

functions embedding shows better performance for 2-d domains in computer vision tasks.

Considering the cases in the line graph of the floorplan CAD drawing, we choose the suitable positional embedding method in the floorplan detection task. Since the context of the primitives in the floorplan CAD drawing is usually local, the relative positional embedding is used in the proposed network. As the primitives are drawn in a 2-d canvas, the learnable functions are used to encode the relative positions.

After adding the position embedding, the initial edge features are passes through another MLP layer to encode the relative spatial relations among vertices:

$$R = MLP(E)$$

where $E \in R^{N\times N\times 7}$ is the edge features by expanding $|E|$ edges to $N \times N$. As the GAT layers go deeper, the kernel function has a larger receptive field and filters the more global context. Compared with the global context, the local context may be more important in the prediction. Thus, the RSE encoding $R \in R^{N\times N\times H}$ is then fed to every stage of the main GAT branch to enhance the local features, where H is the number of heads in the GAT layers.

CEE module. Recall that vertex attentions A^s can be viewed as relational intensity among vertices, which is a good choice for predicting the adjacency matrix. By imitating the pyramid structure in image processing, the author combines the attention scores from local to global for adjacent matrix prediction. Therefore, attention scores are cascaded from all GAT stages $\{A^s\}$ as implicit edge encoding to capture local and global vertex connectivity:

$$C = \sum_{s=1}^{S} A^s.$$

Each valid edge encoding c_{ij} in $C \in R^{N\times N\times H}$ is then concatenated with vertex features of its two endpoints from the last GAT stage to form the final edge feature:

$$\tilde{\boldsymbol{e}}_{\boldsymbol{ij}} = \text{Concat}\left(\boldsymbol{c}_{\boldsymbol{ij}}, \boldsymbol{v}_{\boldsymbol{i}}^{\boldsymbol{S}}, \boldsymbol{v}_{\boldsymbol{j}}^{\boldsymbol{S}}\right).$$

3.3.4 Summary

Different from raster images, CAD drawings are vector graphics consisting of geometric primitives such as segments, arcs, and circles. By treating each CAD drawing as a graph, the author proposes a novel graph attention network GAT-CADNet to solve the panoptic symbol spotting problem: vertex features derived from the GAT branch are mapped to semantic labels, while their attention scores are cascaded and mapped to instance prediction. The proposed key contributions are three-fold:

- The instance symbol spotting task is formulated as a subgraph detection problem and solved by predicting the adjacency matrix.

- A relative spatial encoding (RSE) module explicitly encodes the relative positional and geometric relation among vertices to enhance vertex attention.

- A cascaded edge encoding (CEE) module extracts vertex attentions from multiple stages of GAT and treats them as edge encoding to predict the adjacency matrix.

The proposed GAT-CADNet is intuitive yet effective and manages to solve the panoptic symbol spotting the problem in one consolidated network. In this work, the author presents an intuitive, yet effective architecture named GAT-CADNet for panoptic symbol spotting on CAD drawings. It formulates the instance symbol spotting task as an adjacency matrix prediction problem. The relative spatial encoding module explicitly encodes the relative relationships among vertices to enhance their attention. The cascaded edge encoding module extracts vertex attentions from multiple GAT stages capturing both local and global connectivity information. With the help of the RSE and CEE modules, the proposed GAT-CADNet surpasses other approaches by a large margin.

3.4 Floorplan generation

Artificial intelligence generated content (AIGC) techniques have rapidly developed in recent years. Using AI to create content has many applications, such as floorplan generation in architectural design and video game scene synthesizing (Liggett 2000). In the architecture industry, this technology can be used in floorplan design, which is usually a time-consuming iterative process in traditional architectural design. To efficiently obtain a satisfactory room layout, architects must have professional skills and a deep understanding of customers' intentions. Due to the cost, only a small fraction of buildings whose floorplans are customized for customers.

This task can be divided into room layout prediction and other downstream tasks, *e.g.*, windows prediction, and doors prediction. In architecture design, the bubble diagram is a frequently used constraint to describe the customer's requirements (Merrell *et al.* 2010), which illustrates the numbers and types of rooms, as well as their adjacency relationships. In addition to the topology information, the relative room locations are also illustrated with the vertex positions in the bubble diagram.

A floorplan that fails customers' requirements may be unacceptable in practice. Missing rooms or the wrong room types would cause the incomplete function of the house. The wrong topology may lead to confusion about room accessibility. A generated floorplan with wrong room adjacency may lead to unreachable rooms, which is unreasonable. Thus, the consistent floorplan topology constrained by the given bubble diagram is a vital key to generating feasible floorplans. In some cases, like renovation, the fixed house boundary is also a strict constraint. The floorplan generation tackling the bubble diagram and boundary constraints is studied in this research.

Approaches to solving this task originate from iterative optimization in early time. With the recent breakthrough in natural image generation, deep learning methods, *i.e.*, generative adversarial networks (GANs) and variational autoencoders (VAEs) are naturally used on floorplan generation and get realistic results. According to different formulations of the problem, approaches can be

roughly classified into two categories. One approach represents floorplan as images and predicts room masks at pixel level. An alternate approach predicts the location and size of room boxes by treating floorplans as graphs. Most of the methods generating floorplans at the pixel level miss some topology requirements in the bubble diagram. Besides, the generative networks may lack supervision on topology.

To tackle the above issues, the author proposes a novel framework, namely DualGraph2Plan, which has three modules, i.e., topology subdivision, geometry optimization, and boundary registration. To begin with, a dual graph is constructed from the given bubble diagram. In the topology subdivision module, a novel graph neural network is proposed to predict the topology by classifying the dual edges of the given bubble diagram according to the number of subdivisions. In the geometry optimization module, a network sharing the same feature fusion backbone is used to predict and optimize the coordinates of the dual vertices to obtain the target floorplan. In the boundary registration module, the outer loop of the dual graph is registered with the given house boundary.

The proposed networks are trained and tested with the RPLAN dataset (Wu *et al.* 2019[99]). A chart of the proposed results and other methods is set in the experiment section to illustrate the quality comparison. Results in both quantity metrics and subjective study show that the proposed framework surpasses state-of-the-art methods. The proposed main contributions are highlighted as follows:

- The author decouples the constrained floorplan generation task into topology prediction and geometry optimization problems and solve them on the dual graph separately.
- The author proposes an effective feature fusion backbone between vertices and edges for the planar graphs embedded in the 2D Euclidean space.
- The author proposes a novel hybrid orthogonal optimization module to predict the shape of the room layout and optimize the shape fully satisfy the topology constraints.
- The author proposes a boundary registration module for the extra house boundary constraints.

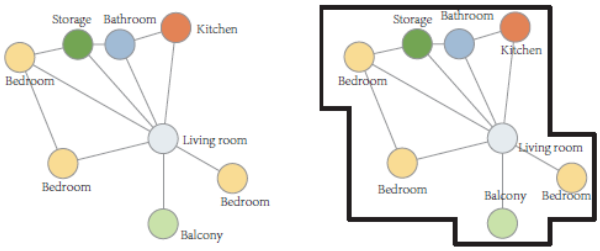
3.4.1 Problem definition

In this section, the author proposes a clear definition and formulation for the floorplan generation task.

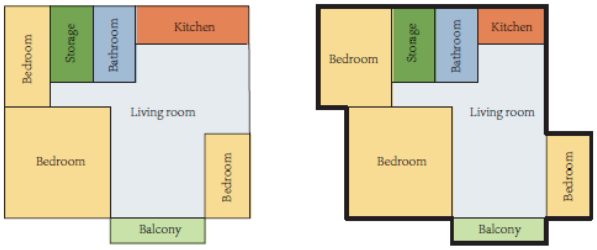
Usually, floorplan design is not a free-generation task. As is mentioned in the background, most traditional architectural floorplan generation starts with the ideas of designers or the requirements of customers. Thus, the input information is the bubble diagrams describing the room types, room positions, and room adjacency of the ideas and the requirements of the expected room layout diagrams. Besides the bubble diagram, in some cases, other fixed conditions also constrain the room layout generation. As is shown in (a) of Figure 30, in this research, the generation task is treated as bubble diagram constrained generation problem (left) and the bubble diagram and boundary

constrained problem (right). The first form is the basic problem in architectural floorplan design, while the second form depicts the commonly seen scene of house renovation in which the boundary is also a fixed constraint.

The most essential step in floorplan design is concreting the input ideas and requirements to a room layout diagram with clear boundaries, in which the room functions, room sizes, space relations, and use flow are roughly determined. So, the output results of this task are the generated room layout diagrams. As is shown in (b) of Figure 30, the topology and boundary walls should be strictly satisfied during the generation.



(a) Customer input bubble diagram (left) and boundary (right) constraints.



(b) Floorplans generated with given constraints.

Figure 30 The definition of the floorplan generation in this research.

As is introduced in section 0, most current deep learning methods formulate the room layout generation problem as a raster image generation problem with a vectorization optimization to obtain the target room layout. However, this formulation is inappropriate since the raster image generated with the decoder structure of the image generative network lacks supervision on topology. Therefore, the target room layout diagram is represented as a graph with 2-d orthogonal embeddings, whose vertices, edges, and faces are the wall corners, wall segments, and rooms respectively.

In summary, the floorplan generation task is formed as: given a bubble diagram $\mathcal{G}_{\mathcal{B}}$ and the fixed house boundary $\mathcal{B}$, generate the room layout diagram $\mathcal{G}_{\mathcal{L}}$. The visualized results of $\mathcal{G}_{\mathcal{L}}$ should not only have the realized shapes as the real floorplans, but also have the same room types, room numbers, and room adjacency as described in $\mathcal{G}_{\mathcal{B}}$.

3.4.1.1 Problem formulation

There are many graph-based deep learning methods, including box-level generation methods (Hu

et al. 2020) and breadth-first graph traverse generation methods (Sun *et al.* 2022), which generate the topology and the geometry of the target room layout graph at the same time. These methods usually have limitations on satisfying the strict constraints. Box-level generation methods may generate room layouts in which rooms are outside the fixed boundary in the bubble diagram and boundary-constrained floorplan generation. And the breadth-first graph traverse generation methods lack supervision on the topology, which generates a vertex with coordinates and updates its adjacency at one time in the sequence of the breadth-first traverse tree of the room layout graph.

Tackling the current issues, the author proposes a novel method for floorplan generation. Noticing that the topology is implied in the input diagram, based on the duality mentioned in section 0, the topology prediction and the geometry prediction of the target room layout diagram are decoupled.

Topology prediction. Based on the upper constructed duality, a subdivision graph neural network is designed to predict the topology of the room layout. The idea in the section is to take the reverse direction of the duality construction in the upper section. A vertex representing f^{∞} is added into vertex set $\mathcal{V}_{\mathcal{B}}$ of the bubble diagram $\mathcal{B}$ and add edges connecting this vertex with all vertices on the boundary of the infinity face in the edge set $\mathcal{E}_{\mathcal{B}}$ of $\mathcal{B}$ to obtain the $\mathcal{G}_{\mathcal{P}}$ (c in Figure 20). This operation keeps the planar property of graphs. The dual graph of $\mathcal{G}_{\mathcal{P}}$ is converted to the complete dual graph $\mathcal{G}_{d}$ (b in Figure 20). The topology of the room layout can be figured out by taking the inverse operation of edge collapse, i.e., subdivision.

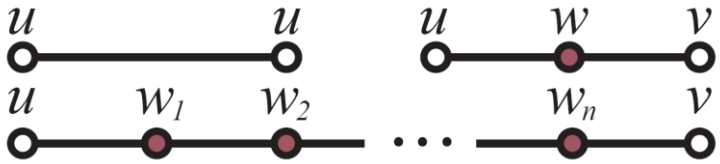


Figure 31 An illustation of the notion of the graph subdivision.

The notion of subdivision refers to an expansion of a graph resulting from the subdivision of its edges. As is shown in Figure 31, one subdivision of an edge with endpoints $\{u, v\}$ yields a graph containing one new vertex w, and with an edge set replacing the original edge by two new edges, $\{u, w\}$ and $\{w, v\}$ (Trudeau 2013).

In the proposed research, the topology is predicted by classifying each edge in $\mathcal{G}_{\mathcal{P}}$ with the number of subdivisions. Since taking dual graphs and subdivisions of a planar graph do not destroy planar property, the proposed predicted topology will always be planar graphs.

Geometry prediction. In the geometry prediction, the 2-d embedding of graph is approximated with the predicted topology. Since all walls are assumed to be line segments, the author only predicts the 2-d coordinates of the vertices on the canvas and connect them according to the predicted topology of the dual graph.

For the training set, all coordinates are normalized into $(0,1)$ by zooming the floorplan to the unite

square. Thus, the geometry prediction is formed as a regression problem. Although the networks in the proposed decoupling method show better performance on the topology consistently, it still lacks efficient supervision on the geometry prediction. So, the author compliments an orthogonal optimization step to make the final generated room layout fully satisfy the strict constraints.

Pipeline. As is shown in Figure 32, the pipeline of the proposed DualGraph2Plan generates a room layout from a given bubble diagram. The orange branch represents the topology prediction process. The bubble diagram is first converted to the complete bubble diagram. The dual of the complete bubble diagram is used for predicting the subdivision.

The green branch represents the geometry prediction process. The 2-d coordinates are predicted with the dual graph updated with the predicted subdivision. The orthogonal optimization is composed of an orthogonal representation further optimizing the topology, a rectangular separation converting the orthogonal representation to convex units, an integral optimization to obtain the compact layout, and quadratic programming to optimize the predicted coordinates to satisfy the topology constraints. The final room layout is visualized with a simple rendering process.

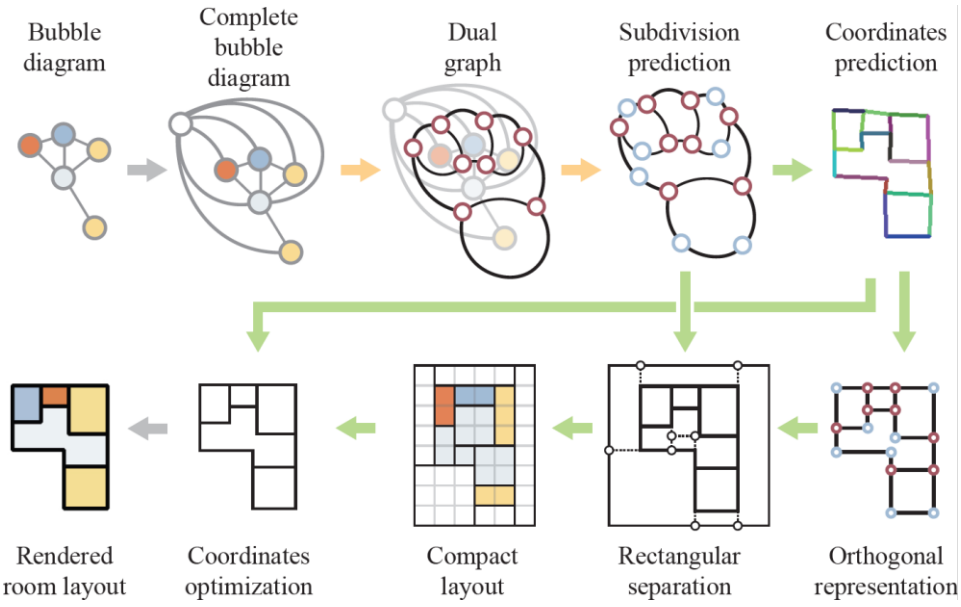


Figure 32 The pipeline of the floorplan generation with the given bubble diagram as constraints.

3.4.1.2 Feature definition

Before the subdivision prediction, the initial features on the vertex space and the edge space of $\mathcal{G}_p$ are defined through the constructed duality. The vertex feature $\mathcal{V}_d$ is naturally defined, $v_d^i \in R^{14}$ as:

$$v_d^i = [x, y, t],$$

where x, y are coordinates of room locations and t is a 12-dimensional one-hot vector indicating

room type. A constant point (0,0) is used to represent the location of v_d^{∞},. In order to encode the combinatorial embedding of the oriented manifold, the author uses the half-edge structure to describe the given bubble diagram. The direction of each half-edge is determined with the counterclockwise orientation of the faces in the bubble diagram. Since the edge e_d^i and its dual edge e_p^i is one-to-one corresponding, with the definition on $\mathcal{V}_d$, the initial edge feature can be defined on $\mathcal{E}_p$. $e_p^i \in R^{30}$ is defined as:

$$\boldsymbol{e_p^i = e_d^i = [v_d^j, v_d^k, k]},$$

where v_d^i, v_d^j are vertex features of the starting and ending vertices of e_d^i, e_p^i and e_d^i are dual-prime relations and k is a 2-dimensional one-hot vector that indicates whether the dual edge is on the boundary. Since the degrees of the dual vertices imply the shape of the wall corners, the initial features on dual vertex space, $\boldsymbol{v_p^i} \in R^3$, is defined as:

$$\boldsymbol{v_p^i} = [d, k],$$

where d is the degree of v_p^i and k is a 2-dimensional one-hot vector indicating the boundary vertex or the inner vertex.

By taking the two above feature spaces as inputs, a novel graph neural network, namely Subdivision GNN, is used to predict the topology of the room layout. Unlike pixel-based or box-based deep learning methods, Subdivision GNN explicitly integrates the topology of the bubble diagram with the network structure.

3.4.2 Overall architecture

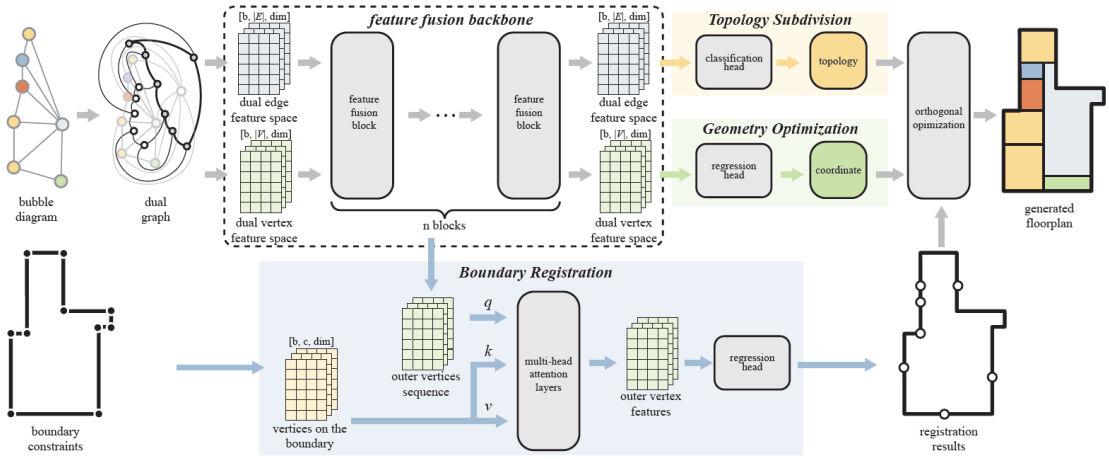


Figure 33 The network structure of the Dualgraph2Plan for floorplan generaion, which is composed with three modules, Topology subdivision (yellow branch), Geometry Optimization (green branch), and Boundary registration(blue branch).

Based on the formulation of the constrained floorplan generation task, the author introduces the overall architecture of the DualGraph2Plan in this section. As illustrated in Figure 33, the overall architecture of the DualGraph2Plan is composed of three modules, a topology subdivision module (the yellow branch), a geometry optimization module (the green branch), and a boundary registration module (the blue branch).

The bubble diagram constrained floorplan generation task can be solved with the first two modules. The topology subdivision module and the geometry optimization module share the same backbone structure composed of several feature fusion layers, which is introduced in detail in section 0. These two modules have different head structures. After embedding all features with the backbone, the vertex and edge features of the dual graph are separately fed to the classification head and the regression head in the topology subdivision module and the geometry optimization module respectively.

The bubble diagram and boundary constrained floorplan generation task can be solved in the upper two basic modules with an additional boundary registration module, in which the outer loop of the dual graph is registered to the given boundary with a vanilla transformer structure.

3.4.3 Subdivision GNN

Based on the duality constructed in section 0, the author designs a subdivision graph neural network (Subdivision GNN) to predict the topology of the room layout. The idea is to take the reverse direction of the duality construction method.

The architecture of the proposed Subdivision GNN is shown in Figure 33 (orange branch). Two different shared Multilayer Perceptron Layers (MLP) embed the vertex and edge features of $\mathcal{G}_{\mathcal{P}}$ into R^{128}. The backbone module is composed of several feature fusion blocks, which are designed for coupling the features defined on two different spaces. By taking the two above feature spaces as inputs, a novel graph neural network, namely Subdivision GNN, is used to predict the topology of the room layout.

Unlike pixel-based or box-based deep learning methods, Subdivision GNN explicitly integrates the topology of the bubble diagram with the network structure. The architecture of the proposed Subdivision GNN is shown in the orange branch of Figure 33. Two different shared Multilayer Perceptron Layers (MLP) embed the vertex and edge features of the dual graph $\mathcal{G}_d$ into R^{128}.

The backbone module is composed of several feature fusion blocks, which are designed for coupling the features defined on two different spaces. The structure of one feature fusion block is shown in Figure 26. Inspired by Smirnov and Solomon (2021) and Desbrun *et al.* (2005), the symmetrically normalized Laplacian in the vanilla GCN structure is replaced with a factorization of the Laplacian in discrete exterior calculus in the proposed neural network for feature fusion:

$$\mathcal{L} = \star_0^{-1} d^T \star_1 d,$$

In this factorization, $d \in \{-1,0,1\}^{|E_p|\times|V_p|}$ is the combinatorial differential matrix mapping features on the $\mathcal{V}_p$ to $\mathcal{E}_p$.

d and d^T are exactly equal to the boundary operator of $\mathcal{F}_d$ and $\mathcal{E}_p$, which contain the topology. $\star_0^{-1}$ and $\star_1$ are two learnable Hodge star operators mapping $\mathcal{F}_d$ and $\mathcal{E}_p$ to their dual spaces $\mathcal{V}_p$ and $\mathcal{E}_d$. In one feature fusion block, the features V^l on $\mathcal{V}_p$ are mapped to $\mathcal{E}_p$, add it with the edge features E^l, and finally map the features back to the next layer feature V^{l+1} defined on $\mathcal{V}_p$. For better convergence, skip connections and feed-forward layers are added in the feature fusion blocks.

At last, a classification head is set to predict the subdivisions of each dual edge. The cross entropy is taken as the loss for subdivision prediction.

$$Loss_{sub} = \frac{1}{N}\sum_i L_i = -\frac{1}{N}\sum_i \sum_{c=1}^{M} y_{ic}\, log(p_{ic})$$

Compared with other deep learning methods, the results of the proposed network have two good properties. Firstly, as is mentioned in duality construction, the topology of predicted graphs is planar. Secondly, since the faces in the proposed results one-to-one correspond to the vertices in the bubble diagram, the topology in the proposed room layout keeps consistent with the one in the given bubble diagram.

3.4.4 Orthogonal and planar optimization

With the above neural network, the topology of the room layout is obtained. For convenience, $\bar{\mathcal{G}}$ is used to represent $\mathcal{G}_p$ updated with the predicted topology. Drawing the room layout on the canvas still requires knowing the 2D embedding (Hocking and Young 2004) $\varphi: \bar{\mathcal{G}} \mapsto R^{\mathbb{2}}$. According to the proposed three assumptions in room layout diagram generation, this mapping needs to satisfy two constraints. One is the planar constraints in which φ is an injective continuous map, and $\bar{\mathcal{G}}$ is a homeomorphism onto $\varphi(\bar{\mathcal{G}})$. Another is orthogonal constraints, in which all embedded edges of $\bar{\mathcal{G}}$ are either vertical or horizontal. Although the correct combinatorial embedding is known, constructing a certain φ with realistic room layout results is still a non-trivial problem.

Before construction, the author first demonstrates the existence of the mapping φ. Clearly, if a graph has an orthogonal drawing, then its maximum degree Δ is less equal than four. Every planar graph with $\Delta \leq 4$ has an orthogonal drawing but may need ***bends***, in Chap.8 of Planar *graph drawing* (Vol. 12) (Nishizeki and Rahman 2004). A constructive proof can be found in a frequently used algorithm in very-large-scale integration (VLSI) applications. Tamassia (1987, 1997) presented this algorithm that finds an orthogonal drawing of a given plane graph with the minimum number of bends in time $\mathcal{O}(n^2 \log n)$ and $\mathcal{O}\left(n^{7/4\sqrt{\log n}}\right)$ respectively.

These algorithms could generate topology-consistent room layouts. However, the shortage is also obvious. This method can only generate the topological equivalence class of the ground truth. On

the other hand, the neural network could approximate the mapping restricted on vertex space $\varphi|_{\bar{\mathcal{V}}}: \bar{\mathcal{V}} \mapsto R^{\mathbb{2}}$, which generates more realistic shapes but may violate the orthogonal and planar constraints, see Figure 34 for illustration.

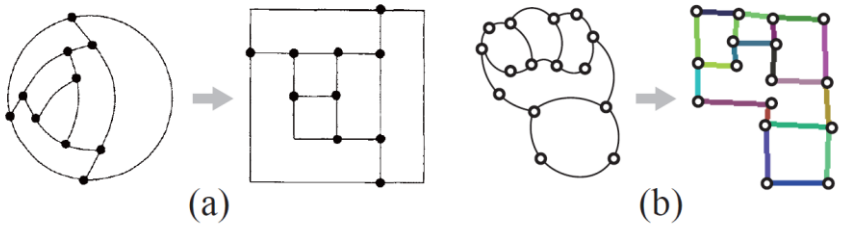


Figure 34 Limitations of vaious floorplan generation methods.

In this research, the author proposed the proposed hybrid method, combining the advantages of a coordinate prediction neural network and the bend-optimal drawing algorithm, which takes the combinatorial embedding and the bubble diagram as input and outputs realistic room layouts satisfying all constraints. To the proposed knowledge, the proposed method is the first to achieve such goals.

The architecture of the coordinate's prediction neural network in the proposed method is illustrated in Figure 33 (green branch). In this network, the dual graph is updated with the predicted topology in Subdivision GNN. The initial features definition and backbone structure are the same as the ones in subdivision GNN. The encoded vertex features of $\bar{\mathcal{V}}$ are fed to a regression head to predict the coordinates. All predicted coordinates are restricted from 0 to 1. The loss in this network is Mean Squared Error:

$$Loss_{reg} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2,$$

in which n is the number of the coordinates, y_i is the ground truth coordinates and $\hat{y_i}$ is the predicted coordinates.

The pipeline of the proposed improved bend-optimal drawing algorithm is shown in Figure 35. The proposed method starts with the predicted coordinates and topology in the upper neural networks. The first step is to obtain the representation of the predicted results by rounding all angles to $\{90°, 180°, 270°\}$. After that, the author adds bends and add adjust the corners so that the representation is an orthogonal representation. With the orthogonal representation, a flow network is built to get the permutations of horizontal and vertical coordinates which satisfy the planar constraints. Finally, the permutations as well as the orthogonal representation are taken as constraints to minimize the sum of the Euclidean distance between the prediction coordinates and target coordinates. The final room layout is obtained after rendering. The details of the steps are illustrated in the following part of this subsection.

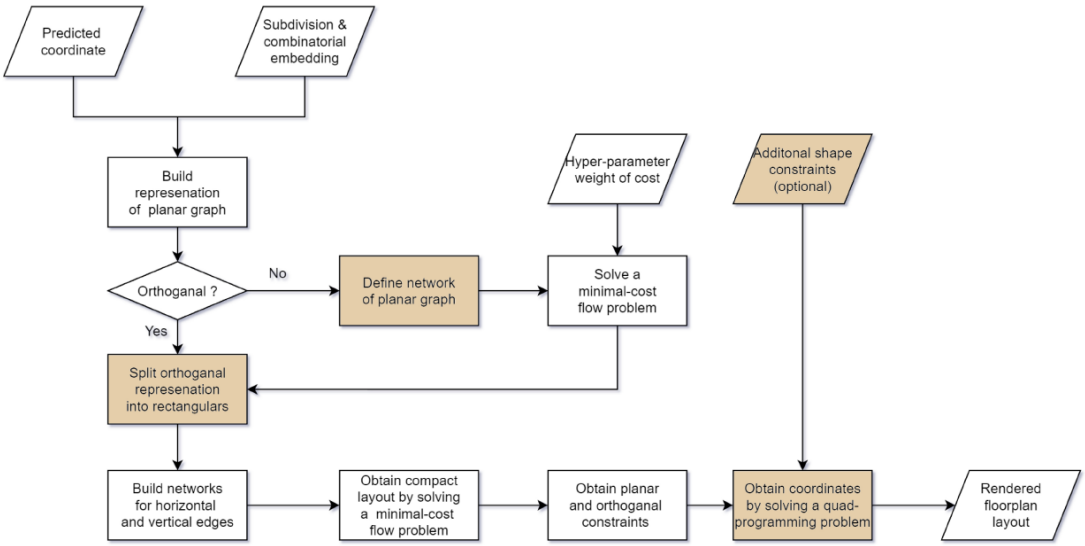


Figure 35 A detail workflow of the post-process step in the Dualgrapg2Plan. The innovation steps are highlighted in the diagram.

3.4.4.1 Orthogonal representation

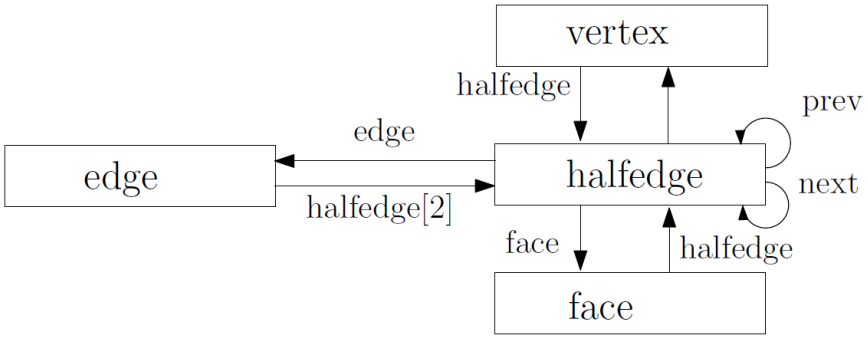


Figure 36 A diagram illustrating the half-edge structure for planar graphs.

Before the optimization, the notion of the orthogonal representation is briefly introduced adhere. In the orthogonal representation, the half-edge structure (Figure 36) is used to describe an oriented planar graph. All faces have the same direction and are represented as lists of their adjacent half-edges. An angle formed by adjacent half-edges incident to a vertex is called a ***vertex-angle***. Clearly, in orthogonal drawing, all ***vertex-angles*** belong to$\{90°, 180°, 270°\}$. The following two facts is obvious:

$$\sum_{i \in V_n} A_i = 360\ ,$$

$$\sum_{i \in F_m} C_i = \begin{cases} 180 * (p-2) & if\ F_m\ is\ a\ room\ , \\ 180 * (p+2) & if\ F_m\ is\ f^{\infty}. \end{cases}$$

in which, A_i are all ***vertex-angles*** around a same vertex v_n, C_i are all ***vertex-angles*** belong to a same Faces F_m, and p is the number of the half-edges of the Faces F_m. In other words, the sum of all angles around each vertex equals 360° and the sum of all angles of each room should be 180*(p-2) (or 180*(p+2) if the face is the outer area). An orthogonal representation $\mathcal{H}$ of a graph is a set of circularly ordered lists for each face. Each element r of a list is a triple (e, s, a), in which e is an edge, $s \in \{1,3\}^n$ is a bit string implies a bend sequence (bit 1 is 90° bend, bit 3 indicates a 270° bend), and $a \in \{1,2,3\}$ indicates the degree $\in \{90°, 180°, 270°\}$ of ***vertex-angle*** formed by e_i and e_{i+1}. In the following context, integer {1,2,3} are used to indicate the degrees instead of $\{90°, 180°, 270°\}$. More details about orthogonal representation can be found in Chapter 8.2.1 of the book by Nishizeki and Rahman (2004).

The predicted results of neural networks are transformed to the same form of the orthogonal representation, by rounding all angles and setting all bit strings as empty sets. The author judges whether a representation is an orthogonal representation or not by the conditions. If the representation violates upper conditions, the workflow goes to the horizontal branch to optimize the representation to orthogonal.

3.4.4.2 Orthogonal optimization

The basic idea of orthogonal optimization is converting the optimization to a minimal cost flow problem. The author first builds a new graph and define a flow on it. As is shown in Figure 37, vertices and faces are taken as nodes in this new graph. The edges in this new graph consist of two parts. One is e_{vf} connecting every vertex to its face, the black edges in the left part of Figure 37. Another is e_{ff} connecting every two adjacent faces, the orange edges in the left part of Figure 37. Both two types of edges in the new graph are bi-directional. As is shown in the right part of Figure 37, the flow defined on e_{vf} indicates the increase (e_1, e_3, e_5) and decrease (e_2, e_4, e_6) of vertex-angles, according to the edge direction of e_{vf}.

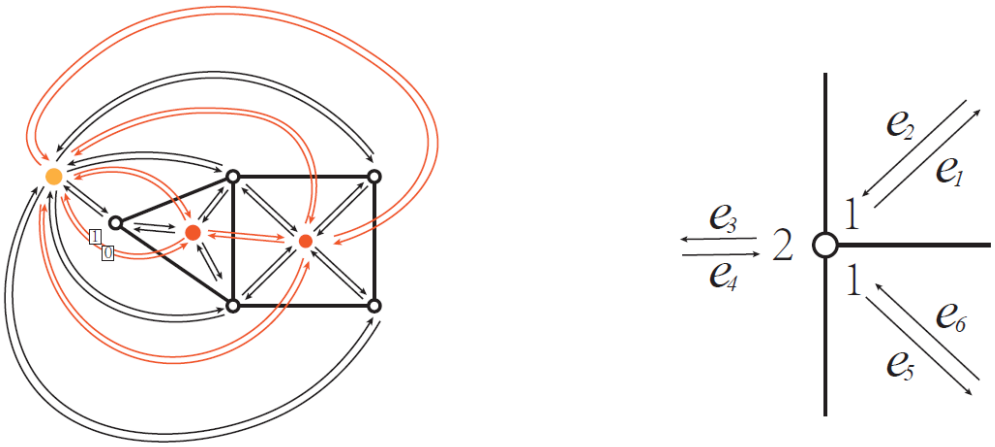


Figure 37 An illustraion of the flow network construction in the orthogonal representation optimization step. The method (left) builds graph for orthogonal representation optimization. The local graph (right)

of a vertex.

The upper bound of the flow on e_{vf} must ensure the changed angle in $\{1,2,3\}$. The right part of Figure 37 is taken as an example for illustrating the ranges on e_{vf}:

$$\begin{cases} e_1 \in \{0,1,2\} \\ e_3 \in \{0,1\} \\ e_4 \in \{0,1\} \\ e_5 \in \{0,1,2\} \\ e_2 = e_6 = 0 \\ 1 + 2 + 1 + (e_1 + e_3 + e_5) - (e_2 + e_4 + e_6) = 4 \end{cases} .$$

And the flow defined on e_{ff} is a non-negative integer, indicating the number of $90°$ bends on each edge:

$$e_{ff} \in N .$$

With the proposed defined network on the graph, the orthogonal optimization is converted to a minimal cost flow problem. All flows are non-negative integers and the orthogonal representation conditions are used to set the supplies of each node. Compared with the algorithms by Tamassia (1987,1997), the author creatively sets e_{vf} as bidirectional to approximate the angle in the prediction of the coordinate neural network.

In the proposed method, weights on e_{vf} and e_{ff} are two hyper-parameters used for manually balancing the approximation to coordinates and minimal bends.

3.4.4.3 Rectangular division

So far, the orthogonal representation $\mathcal{H}$ of the room layout is obtained as well as the predicted coordinates for shape reference. The orthogonal constraints for coordinates can be obtained from the orthogonal representation. Since all predicted coordinates are restricted in $(0,1)^2$, the upper left corner vertex with minimal distance to the origin must have the pattern as (a) in Figure 38. It could be easily proven with contradiction. Based on the upper conclusion, the horizontal and vertical positions of each edge could be figured out with optimizations. It is difficult to enumerate all possible permutations of coordinates in both vertical and horizontal directions to satisfy the planar constraints. The author creatively separates all concave rooms into rectangles to obtain the target permutations. This task is also not easy. On one hand, separation with only orthogonal representation may have a larger deviation from the predicted coordinates. On the other hand, separation with only predicted coordinates may cause errors because the coordinate neural network may not satisfy the planar constraints, see (b) and (c) in Figure 38.

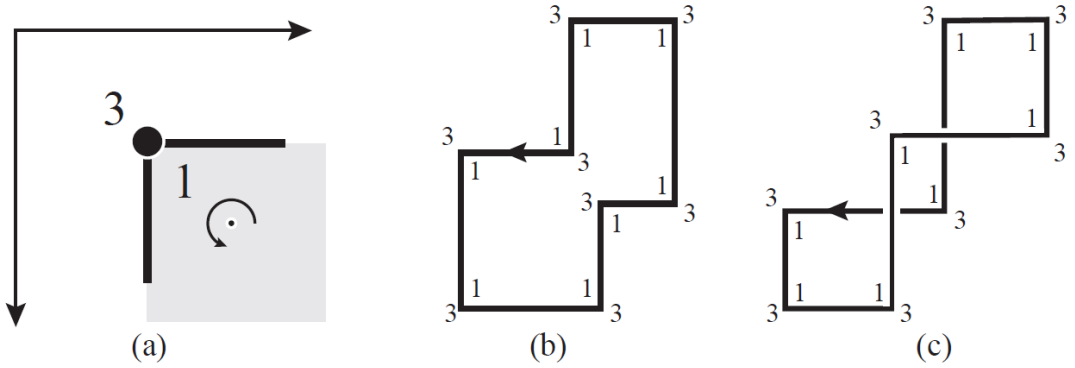


Figure 38 (a)The way to obtained the horizontal direction of the orthogonal representation with the orientation of the dual graph. (b) and (c) have same orthogonal representation but different 2D embedding.

In this research, the author tackles this problem with a novel algorithm to separate the concave orthogonal shapes into rectangles based on both the orthogonal representation and the predicted coordinates. As is shown in Figure 39, one dummy point (hollow points in Figure 39) separates a concave shape into two fragments. Dummy points are added with the proposed algorithm until all fragments are convex shapes.

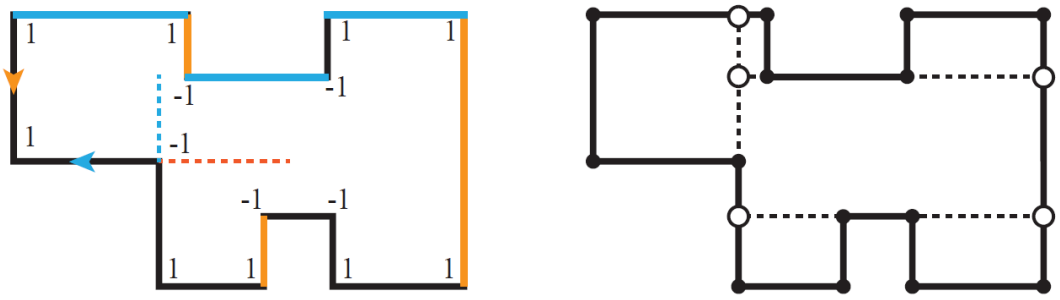


Figure 39 The diagram illustrated the rectangle separetion process.

The algorithm of how to add one dummy point is illustrated as follows:

Algorithm: Add one dummy point.

Require: Orthogonal representation $\mathcal{H}$, predicted coordinates.

Select one concave face F_0 in $\mathcal{H}$.

Label all $\{1,2,3\}$ ***vertex-angles*** with $\{1,0,-1\}$.

Select any vertex in F_0 with a label -1 as the start point.

1. Go alone in the orientation order (orange in Figure 39).

For v_j, j **in** $\{i, i+1, \ldots, 0, \ldots, i-1\}$ **do**

If $\Sigma_i^j labels == 1$ **then**

Append the edge $[v_j, v_{j+1}]$ as one candidate edge in the orange set for the dummy point.

End if

End for

2. Go alone in the reverse direction order (blue in Figure 39).
   For v_j, j **in** $\{i, i-1, \dots, 0, \dots, i+1\}$ **do**
   If $\Sigma_i^j labels == 1$ **then**
   Append the edge $[v_j, v_{j-1}]$ as one candidate edge in the orange set for the dummy point.
   End if
   End for

Calculate the intersection D_k of the edge $[v_{i-1}, v_i]$ and $[v_{i+1}, v_i]$ with all candidate edges in the orange and blue set respectively.

Calculate the distance d_1 of D_k and v_i and the distance d_2 of D_k and the corresponding candidate edge.

The D_k with minimal $d_1 + \lambda * d_2$ will be the dummy point.

Use this dummy point to separate F_0 and update $\mathcal{H}$.

Since a considerable proportion of room layouts whose boundaries are not rectangles, the author adds an external rectangular frame and separate the outer area shown in Figure 40. The author copies the upper left corner vertex s and add a dummy point t on the upper edge of the external frame to cut the ring into a disc. Then the ring separation problem is transformed into a solved problem. Finally, s, t are glued with s', t' respectively. Until now all rooms are split into rectangles.

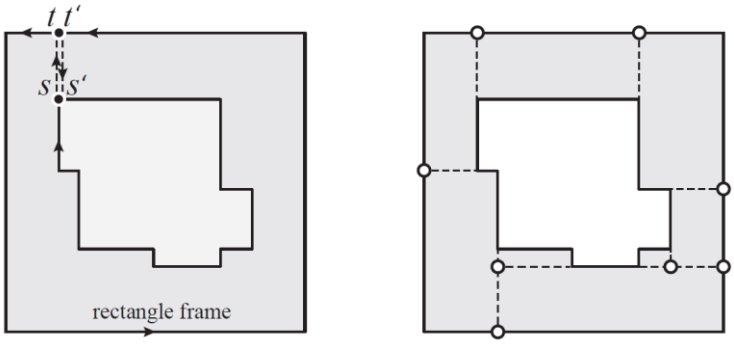


Figure 40 The trick used to convert boundary to a rectangle.

3.4.4.4 Coordinates optimization

In order to figure out the permutation of the coordinates, we make use of the rectangular orthogonal representation to calculate a compact room layout. As is shown in Figure 41, the length of every edge is minimized also by solving a minimal-cost flow problem, such that orthogonal and planar constraints are satisfied. Since the vertical and horizontal are independent, the length of two directions can be parallel solved. Vertical edges are taken as examples for illustration. First, a graph is built, whose nodes are each rectangle plus a left source node and a right sink node. Second, two nodes are connected if two rectangles are horizontally adjacent, and the direction is from left to right. Finally, an additional edge is added from the sink node to the source node. The flow defined on edges is non-negative integers whose lower bounds are 1 and upper bounds are infinity. Every edge has the same weight. The minimized cost and the compact layouts are illustrated in the right part of Figure 41. With the compact layout, all constraints can be obtained and feed the following

coordinate optimization.

With the upper information, the approximate coordinates that satisfy orthogonal and planar constraints can be figured out. The final coordinates can be figured out with such following quadratic-program problem:

$$\min_{x_i} \ \sum \| \|\boldsymbol{c_i} - \boldsymbol{x_i}\|^2$$

$$s.t. \quad O^C \quad and \quad P^C \quad are \quad satisfied.$$

where c_i is the predicted coordinate, x_i is the target coordinate, O^C and P^C are orthogonal constraints and planar constraints. This coordinates optimization can be solved with the least squares method. Other optional constraints, such as the requirements of room sizes or specific locations of walls, can be added to the coordinate's optimization at this step. So far, the final generated room layout is obtained, whose bubble diagram is 100 percent isomorphism to the given bubble diagram.

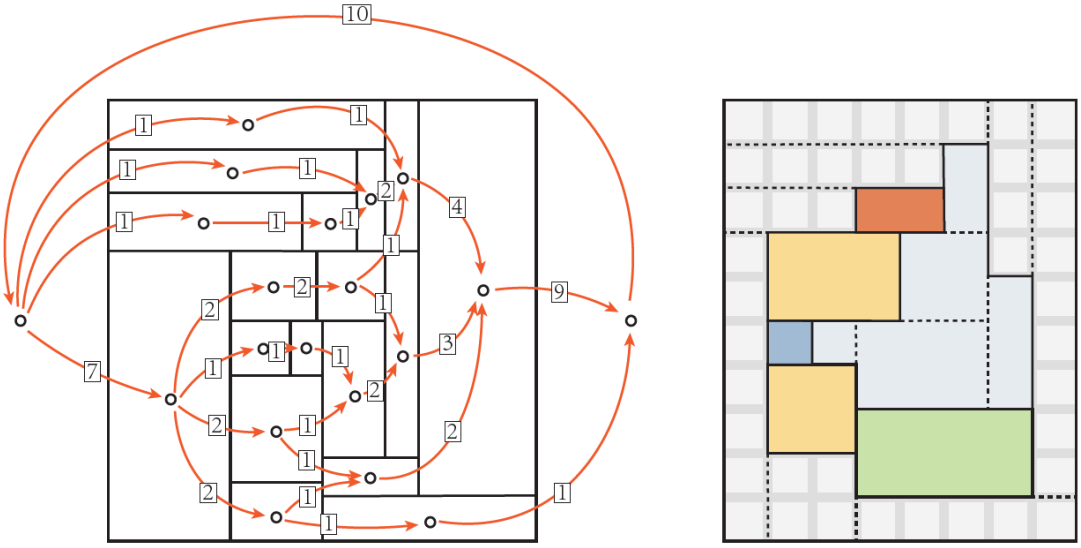


Figure 41 The flow network (left) defined to obtained the compact layout of the orthogonal representation (right).

3.4.5 Boundary constrained generation

Apart from the bubble diagram, the boundary of a house is another common and strict constraint that needs to be complied with in most practical applications. Since the author decoupled the topology and geometry in previous sections, the proposed framework can handle these extra constraints easily. The author only needs to register the outer loop of the dual graph to the fixed boundary and predict the inner part without breaking the room topology. The given boundary and

outer loop of the dual graph are aligned in the boundary registration module. The structure of the registration network is illustrated in Figure 33.

The bubble diagram and boundary-constrained floorplan generation task are similar to the previous problem with an additional fixed boundary constraint. Correspondingly, the boundary registration module in the DualGraph2Plan is introduced to satisfy the boundary constraint.

The given boundary of the building is a circle which is a homeomorphism onto the boundary of the infinite face in the dual graph of the complete bubble diagram. Thus, the topology of the boundary is fixed, and the author only needs to determine the positions of the vertices on the boundary of the infinite face in the dual graph.

Meanwhile, these positions are restricted on the given boundary. Unlike predicting the 2-d coordinates in the bubble diagram constrained generation, the given boundary is a 1-dimensional geometry primitive (closed polyline) whose topology is a homeomorphism onto S^1. The position of the outer vertices in the dual graph can be located with the parameters of arc length parameterization of the given boundary.

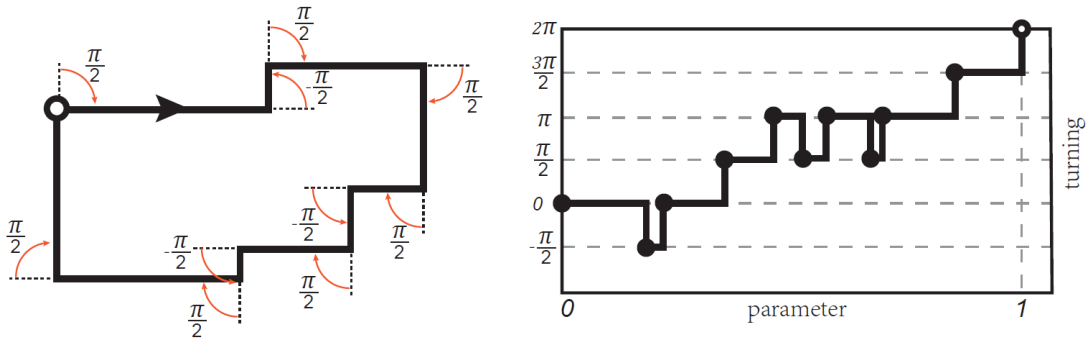


Figure 42 The represeation of the fixed house boundary constraints.

In the boundary registration module, the author also uses the deep learning method to predict parameters of the outer vertices on the given boundary, which is a registration process of the given boundary and the boundary of the infinite face in the dual graph. The neural network used in the module is the transformer structure.

As is shown in the left part of Figure 42, the input fixed boundary is represented by the vertices sequence in the clockwise order. The initial feature of the vertices in the sequence is defined as:

$$v_b = [x, y, p, \tau],$$

in which x, y are the 2-d coordinates of the vertices on the fixed boundary in the normalized square canvas, p is the normalized arc length parameter of each vertex, and τ is the turning angle of each vertex. The last two features of v_b is shown in the right part of Figure 42. To obtain the normalized arc length parameter of each vertex, the author splits the fixed boundary with the top left vertex and

map the boundary to $[0,1)$.

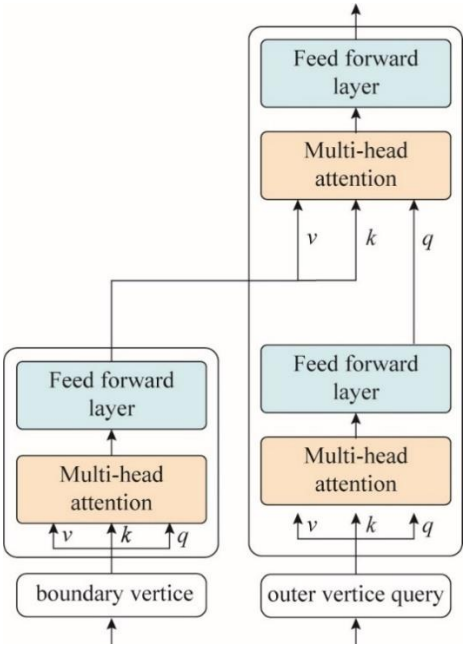


Figure 43 The vanilla tranformer network structure in the boundary registration module.

The transformer structure is shown in Figure 43. The author mimics the decoder part in the vanilla transformer network to design the proposed arc length parameter prediction structure in the boundary registration module. The structure of the boundary registration module is shown as the blue branch in Figure 33. The vertex feature of the dual graph is first encoded with the feature fusion backbone. The embedding feature of the vertices on the boundary of the infinite face in the dual graph is selected as a sequence in the clockwise order and fed to the decoder layer of the transformer as the *query*. The initial feature of the vertices on the given boundary is mapped to R^{128} and also fed to the transformer as the *value* and *key*.

The prediction head in this module is a regression head. In this head, the decoded vertices on the boundary of the infinite face in the dual graph are mapped to a real number with a fully connected layer, which represents the arc lengths. Since the sum of normalized arc lengths of all segments of the boundary equals one, the output real numbers are normalized with a SoftMax layer:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

With the predicted parameters of the boundary vertices, the author only has to predict the topology and the geometry of the inner part of the floorplan, which can be solved with the topology subdivision module and the geometry optimization module respectively with a mask filtering outer vertices and edges.

3.4.6 Summary

Using deep neural networks to assist floorplan generation has gained increasing interest in the community. In practice, a generated floorplan conflicting with customer preference may lead to room usage and accessibility being out of order. However, existing methods may produce floorplans that seem fine but do not fulfill customers' requirements or are even infeasible.

In this research, by taking the dual graph of the input bubble diagram, the author directly generates floor plan drawings at the graph level, with a proposed novel method with three modules. By taking the dual graph of the given bubble diagram and a fixed boundary as input constraints, the author proposes a novel framework with three modules, which could generate floorplans with correct topology and realistic geometry. The proposed key contributions are listed as following:

- A novel generation pipeline, in which the topology and geometry are decoupled and predicted on the dual graph to ensure correct topology and realistic geometry.

- A novel feature fusion backbone is proposed using the combinatorial differential matrix of the dual graph to aggregate features defined on different graph entities effectively.

- An integral-programming based post process algorithm for orthogonality.

- A boundary registration module is proposed for the additional boundary constraints.

The proposed results reach an equivalent realistic level as those generated with state-of-the-art techniques but with much higher topological similarity. The proposed dual graph-based approach surpasses current methods in the public dataset and subjective comparison. Besides, the proposed method also has good extensibility to the bubble diagram and boundary constraints.

4 Experiments

4.1 Floorplan detection

In this section, experiments are conducted to test the symbol recognition method and clean wall polygons baseline. This section contains two independent experiments to test the heterogeneous GNN network and the U-net-based baselines.

4.1.1 Dataset and implementation

Dataset. Although there are several floorplan CAD drawings datasets, few are close to the real situation in the industry. Dataset SESYD by Delalandre *et al.* (2010) and FPLAN-POLY by Rusinol *et al.* (2010) are vector graphic CAD drawings. But they are synthesis floorplan images and have a limited number of samples. RPLAN by Wu *et al.* (2019) and Lifull dataset have a large number of samples. However, their data are mainly the room layout diagram raster images of residential buildings. After the proposed investigation, the author finally determined to use the FloorplanCAD by Fan *et al.* (2021) dataset as the proposed train and test set in the floorplan detection task.

In FloorplanCAD (Fan *et al.* 2021), there are over 15,000 as-built CAD drawings with line-grained annotations, covering various types of buildings, e.g., residential buildings, towers, schools, hospitals, shopping malls, and office buildings. The distribution of the FloorplanCAD (Fan *et al.* 2021) dataset is shown in Figure 44. There are a total of 31 classes of primitives in the dataset. Four uncountable stuff classes are parking, wall, curtain wall, and handrail. The rest classes are countable instance classes. According to the statistic of the different classes of primitives in FloorplanCAD (Fan *et al.* 2021), the primitives of wall and chairs has the most numbers, since the wall and chairs are the basic elements in architectural structure and furniture.

In most cases, a CAD drawing of public architecture contains too many primitives for the memory of hardware. Therefore, all floorplan CAD drawings are split in the dataset into $10\,m \times 10\,m$ squares. Besides, the author also omits the other types of information except for geometric primitives.

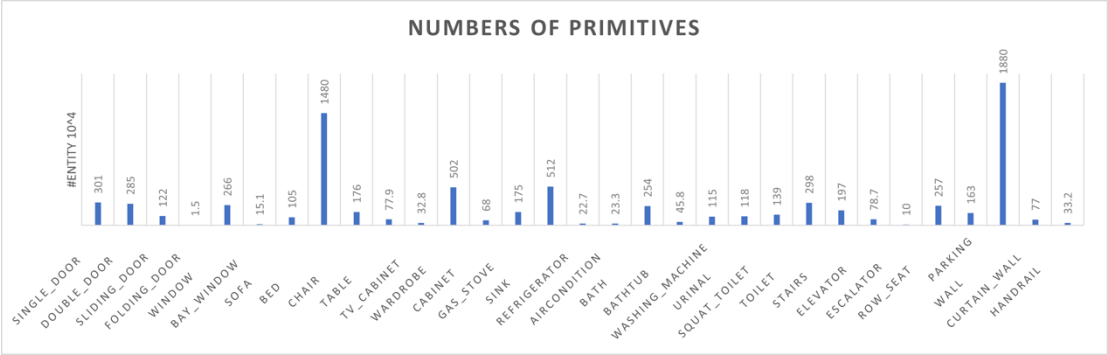


Figure 44 The distribution of each class of primitives in FloorplanCAD dataset.

Implementation. The author trains the proposed network with one 12 GB memory Nvidia GTX 2080Ti GPU with 8 samples for each batch. The author chooses the ADAM optimizer (Kingm and Ba 2014) with β1 = 0.9 and β2 = 0.999 and lr = 0.0001. In order to avoid over-fitting and stabilization, a weight decay rate of 0.0005 and gamma with 0.7 for every 20 epochs are used in experiments. Other CNN-based wall polygon generation baseline networks are trained on the same devices with MMsegmentation developed by MMlab (Chen *et al.* 2019).

4.1.2 Metrics

Mimicking the metrics of the panoptic symbol spotting task on raster images, the metrics for this task on CAD drawings are introduced in this section.

In this research, to compare different methods fairly, the author chooses the metrics which are most accepted in related research. Following the work by Fan *et al.* (2021), the author uses panoptic quality (PQ), recognition quality (RQ), and segmentation quality (SQ) to metric the performance of the networks on floorplan detection with CAD drawing data:

$$PQ = RQ \times SQ = \frac{\sum_{(s_p,s_g)\in TP} IoU(s_p,s_g)}{|TP|+\frac{1}{2}|FP|+\frac{1}{2}|FN|},$$

$$RQ = \frac{|TP|}{|TP|+\frac{1}{2}|FP|+\frac{1}{2}|FN|},$$

$$SQ = \frac{\sum_{(s_p,s_g)\in TP} IoU(s_p,s_g)}{|TP|},$$

where RQ is the F_1 score measuring the recognition quality.

Truth Class / Predicted Class	Positive	Negative
True	True Positive (TP)	True Negative (TN)
False	False Positive (FP)	False Negative (FN)

Table 2 The illustration of the confusion matrix.

$|TP|$, $|FP|$, $|TN|$, and $|FN|$ are the entry numbers in the confusion matrix shown in Table 2. The F_1 score is a commonly used metric in multi-tag classification task:

$$F_1 = \frac{2PR}{P+R},$$

$$P = \frac{|TP|}{|TP|+|FP|},$$

$$R = \frac{|TP|}{|TP|+|FN|},$$

where P and R are the precision and recall of the predicted results. SQ is the segmentation quality computed by averaging IoU of matched symbols. s_p and s_g refer to the subgraph, i.e., the instance of architectural element, from prediction and ground truth. The IoU is primitive-grained intersection rate over onion of instance segmentation, whose original form is shown as:

$$IoU = \frac{detected\_instance \cap true\_instance}{detected\_instance \cup true\_instance}$$

And its form on primitive level is shown as:

$$IoU = \frac{|TP_{instance}|}{|TP_{instance}| + |FP_{instance}| + |FN_{instance}|}$$

4.1.3 Evaluations

4.1.3.1 Semantic segmentation

To compare with existing image segmentation methods, the CAD drawings are rendered as images with a line width of 2 pixels. The semantic of a primitive in the line graph $\mathcal{G}$ is then retrieved by sampling on the predicted mask with a majority voting strategy.

PanCADNet (Fan *et al.* 2021) is a GCN architecture for semantic symbol spotting and relies on image features from a CNN backbone. Table 3 compares the results of popular segmentation methods by Chen *et al.*(2017) and Wang *et al.*(2020) with different configurations.

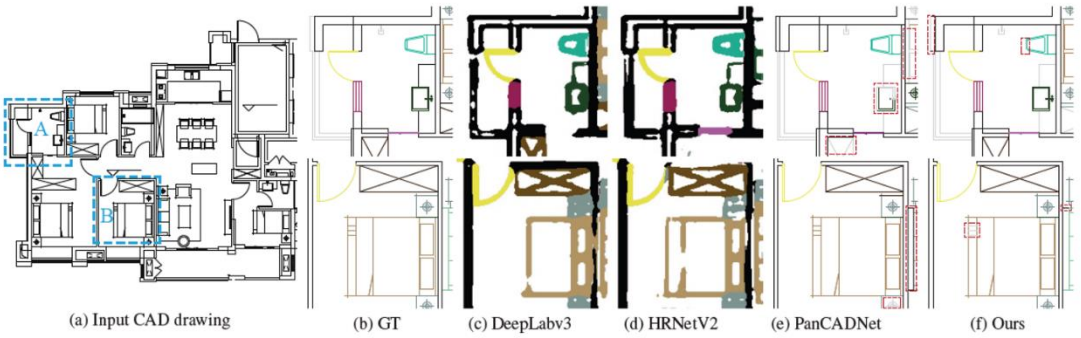


Figure 45 A qualitative comparison of DeepLabV3, HRNetV2, PanCADNet, and the proposed CAD-GATnet on semantic segementation task.

Qualitative comparison are shown in Figure 45, where DeepLabv3 (Cheng *et al.* 2018) and HRNetV2 (Wang *et al.* 2020) are with the W48 and R01 configuration in Table 3 respectively.

While the proposed GAT-CADNet is built on the graph entirely and requires geometric features only, it manages to outperform other image-based methods.

Methods	F1	length-weighted F1
HRNetsV2 W18	0.656	0.683
HRNetsV2 W48	0.666	0.693
DeepLabv3+R50	0.680	0.705
DeepLabv3+R101	0.688	0.714
PanCADNet	0.806	0.798
Ours	**0.850**	**0.823**

Table 3 The quantitative comparison of DeepLabV3, HRNetV2, PanCADNet, and the proposed CAD-GATnet on semantic segementation task.

4.1.3.2 Instance detection

As reported by Fan *et al.*(2021, 2022), traditional symbol spotting algorithms (Nguyen *et al.* 2008,2009, Rusinol *et al.* 2010) have lower generalization ability and are omitted in the comparison. By rendering CAD drawings into images, the proposed GAT-CADNet is compared with various image detection methods, including the two-stage Faster-RCNN (Ren *et al.*2015), the one-stage YOLOv3 (Redmon *et al.*2016), and the more recent FCOS (Tian *et al.* 2019). Note that the instance head in PanCADNet (Fan *et al.* 2021) is from Faster-RCNN and is not listed here.

Methods	AP50	AP75	mAP
Faster R-CNN [31]	0.693	0.631	0.568
YOLOv3 [30]	0.656	0.431	0.395
FCOS [38]	0.648	0.572	0.525
Ours	**0.735**	**0.680**	**0.690**

Table 4 A quantitative comparison of Faster R-CNN, YOLOV2, FCOS, and the proposed CAD-GATnet on instance segmentation task.

The image-based detection methods (Ren *et al.*2015) predict bounding boxes directly, while the author predicts instance labels for each geometric primitive. For a fair comparison, the bounding box of each instance symbol are calculated and averaged connection intensity are used as the confidence score. Quantitative comparisons are listed in Table 4 and the proposed GAT-CADNet surpasses other methods by a large margin.

One thing noteworthy is that the proposed average precision (AP) does not drop dramatically when increasing the IoU threshold and has a much higher mAP score. Since CNNs rely on local patch texture for recognition and may ignore features at the border, it is not a surprise that their box predictions are less accurate due to the low texture in CAD drawings. Such a phenomenon can be observed in Figure 46, where the proposed primitive-level prediction has clearer bounding boxes.

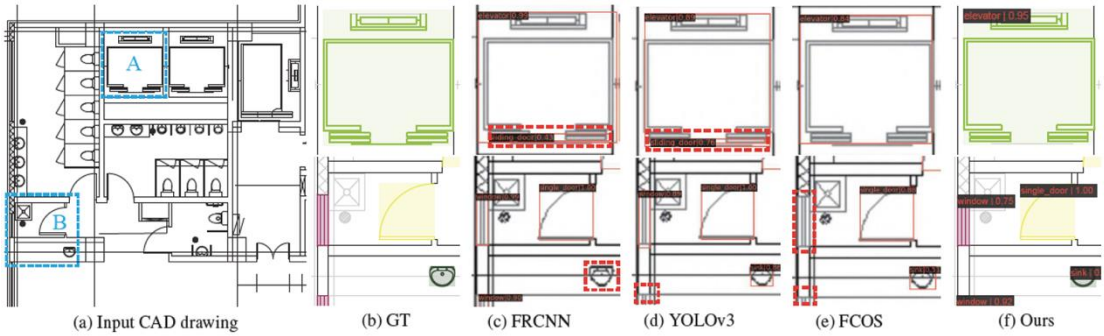


Figure 46 A quantitative comparison of Faster R-CNN, YOLOV2, FCOS, and the proposed CAD-GATnet on instance segmentation task.

4.1.3.3 Panoptic symbol spotting

Converting CAD drawings into images and applying panoptic segmentation algorithms to them is a straightforward approach. However, as demonstrated in the aforementioned comparison sections, the image-based methods are less capable of recognizing abstract symbols at a geometric primitive level.

PanCADNet (Fan *et al.*2021) provides a CNN-GCN architecture for panoptic symbol spotting. It constructs a graph on the CAD drawing first, then fetches CNN multi-layer features to each vertex

and uses a simple GCN structure for recognition. Since PanCADNet (Fan *et al.*2021) adopts Faster-RCNN as its backbone and detection head, there is no surprise that it has a much lower recognition quality than the proposed model. In addition, it does not encode inter-vertex relation explicitly and even has lower recognition and segmentation than the proposed baseline model.

class	Baseline			Basline + RSE			Baseline + CEE			Basline + RSE + CEE		
	RQ	SQ	PQ	RQ	SQ	PQ	RQ	SQ	PQ	RQ	SQ	PQ
single door	0.78	0.91	0.71	0.84	0.93	0.78	0.88	0.93	0.82	**0.91**	**0.95**	**0.86**
double door	0.82	0.91	0.75	0.86	0.93	0.80	0.84	0.93	0.79	**0.89**	**0.94**	**0.83**
sliding door	0.89	0.94	0.83	0.90	0.94	0.85	0.90	0.95	0.85	**0.94**	**0.96**	**0.91**
folding door	0.34	0.85	0.29	**0.46**	0.90	**0.42**	0.39	**0.91**	0.35	0.45	0.89	0.40
revolving door	**0.00**	**0.00**	**0.00**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
shutter door	**0.00**	**0.00**	**0.00**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
window	0.69	0.81	0.56	0.71	0.82	0.58	0.74	0.81	0.60	**0.79**	**0.84**	**0.66**
bay window	**0.00**	**0.00**	**0.00**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
shutter window	0.69	0.82	0.56	0.75	0.85	0.64	0.74	0.84	0.62	**0.76**	**0.87**	**0.66**
opening symbol	**0.00**	**0.00**	**0.00**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
sofa	0.40	0.81	0.32	0.47	0.89	0.42	0.36	0.91	0.33	**0.61**	**0.96**	**0.59**
bed	0.68	0.90	0.61	0.68	0.88	0.60	0.64	0.91	0.58	**0.78**	**0.91**	**0.72**
chair	0.38	0.84	0.32	0.58	0.85	0.49	0.66	**0.94**	0.62	**0.84**	0.93	**0.78**
table	0.36	0.88	0.32	0.31	0.88	0.27	0.40	0.93	0.37	**0.57**	**0.94**	**0.53**
TV cabinet	0.45	0.84	0.38	0.54	0.83	0.45	0.39	0.87	0.34	**0.73**	**0.94**	**0.69**
wardrobe	0.75	0.81	0.61	0.67	0.81	0.55	0.71	0.87	0.62	**0.84**	**0.92**	**0.77**
cabinet	0.20	0.79	0.16	0.21	0.82	0.18	0.18	0.80	0.14	**0.44**	**0.85**	**0.37**
gas stove	0.92	0.96	0.88	0.81	0.95	0.77	0.92	0.96	0.88	**0.94**	**0.98**	**0.92**
sink	0.75	0.93	0.69	0.78	0.93	0.72	0.77	0.94	0.73	**0.81**	**0.95**	**0.77**
refrigerator	0.72	0.81	0.58	0.72	0.87	0.62	0.76	0.85	0.64	**0.88**	**0.93**	**0.82**
air conditioning	0.46	0.89	0.41	0.59	0.92	0.55	**0.68**	0.96	**0.66**	0.66	**0.97**	0.64
bath	0.24	0.77	0.19	0.31	0.77	0.24	0.36	**0.80**	0.29	**0.46**	0.79	**0.36**
bathtub	0.51	0.78	0.40	0.54	0.85	0.46	0.63	0.81	0.51	**0.68**	**0.88**	**0.59**
washing machine	0.74	0.85	0.63	0.61	0.86	0.53	0.69	0.86	0.59	**0.75**	**0.94**	**0.70**
urinal	0.91	0.97	0.89	0.92	0.97	0.89	0.92	0.98	0.90	**0.94**	**0.99**	**0.92**
squat toilet	0.88	0.90	0.79	0.91	0.93	0.85	0.77	0.93	0.72	**0.92**	**0.96**	**0.88**
toilet	0.88	0.96	0.84	0.89	0.97	0.87	0.91	0.96	0.88	**0.94**	**0.99**	**0.93**
stairs	0.53	0.83	0.44	0.62	0.86	0.53	0.68	0.86	0.58	**0.74**	**0.90**	**0.66**
elevator	0.76	0.92	0.70	0.82	0.93	0.76	0.81	0.93	0.75	**0.84**	**0.95**	**0.80**
escalator	0.17	0.73	0.12	0.23	0.74	0.17	**0.25**	0.74	**0.18**	0.22	**0.78**	0.17
row seat	0.26	0.90	0.23	0.46	0.92	0.42	0.33	0.92	0.30	**0.49**	**0.93**	**0.45**
parking	**0.83**	0.88	0.73	0.82	0.90	0.74	0.77	0.86	0.66	0.82	**0.90**	**0.74**
wall	0.66	0.71	0.47	0.72	0.74	0.53	0.68	0.72	0.48	**0.76**	**0.76**	**0.58**
curtain wall	0.28	0.77	0.22	0.33	0.74	0.24	0.21	0.75	0.15	**0.40**	**0.78**	**0.32**
handrail	0.12	0.65	0.08	0.14	0.72	0.10	0.10	0.69	0.07	**0.27**	**0.78**	**0.21**
total	0.69	0.87	0.60	0.73	0.90	0.65	0.75	0.90	0.67	**0.80**	**0.91**	**0.74**

Table 5 A quantitative results of the proposed CAD-GATnet on instance segmentation task of each classes. First three colunms are the results with different setting.

Figure 47 More visualization of the floorplan detection results with the proposed CAD-GATnet.

4.2 Floorplan generation

Qualitative and quantitative evaluations of the proposed neural-guided method are conducted for the constrained room layout drawings generation on the public dataset. The author also compares the proposed method with former SOTA methods (Nauata *et al.* 2021, Hu *et al.* 2020, Sun *et al.* 2022).

4.2.1 Dataset and Implementation

The author trains and tests the proposed method with the RPLAN dataset by Wu *et al.* (2019) which contains abundant residual buildings floor plans, including apartments and houses. 70\%-10\%-20\% of 80k samples in the dataset are set as training, validation, and testing split. Before training, the image samples in RPLAN are processed into graph-based data.

In the following experiments, the proposed networks share the same backbone structure with 8 coupling encoder blocks. The proposed graph-based models are light enough, so the author trains them on a single NVIDIA-RTX 3080 with 10 GB memory. The author uses the Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.99, lr = 0.0001$ and set the decay rate to 0.8 for every 20 epochs. The author trains the proposed networks for 100 epochs and take the best model on the validation split. All other open-source baselines are trained on the same device with default parameters in the released codes.

In the optimization stage, the minimal cost flow problem and the quad-programming problem are solved with *OR-Tools* developed by Google Inc. and an open-source kit *CVXOPT*.

4.2.2 Metrics

Several quantitative metrics are introduced in (Hu *et al.* 2020, Para *et al.* 2021, He *et al.* 2022). But the suitable metrics for floor plans are still discussed in the community. In this research, the author takes the most frequently used metrics in related research Fréchet Inception Distance (FID) (Heusel *et al.* 2017):

$$FID = \left\|\mu_r - \mu_g\right\|^2 + Tr\left(\Sigma_r + \Sigma_g - 2\left(\Sigma_r\Sigma_g\right)^{\frac{1}{2}}\right),$$

commonly used in natural image generation to evaluate the shape similarity between generated results and the ground truths image, where μ is the empirical mean, Σ is empirical covariance, subscripts r and g represent the real data and the generated data, and Tr takes the trace of the covariance matrix.

Another is Graph Editing Distance (GED) (Abu-Aisheh *et al.* 2015), the minimum number of inserting, deleting, and substituting operations to convert one graph to another, evaluating the topological similarity between the bubble diagram in the generated room layouts and the given

bubble diagram constraints.

For a fair comparison, the results of all methods are rendered with same the color setting and line weight. Since the FID score is highly correlated to the example number, the author randomly selects sufficient samples in the test split during evaluation, to keep the FID in the same order of magnitude as previous work. 20 volunteers are recruited to conduct a subjective comparison as a complementary metric.

4.2.3 Bubble diagram constrained generation

The room layouts are generated with the bubble diagram constraints by the proposed method. During training, with a given bubble diagram, the author classifies all edges of the dual graph by the number of subdivisions with Subdivision GNN. And then the ground truth of subdivision is used for the coordinate network training. During reasoning, the predicted topology is used to update the dual graph.

It is not a surprise that all the proposed generated room layouts have the same room topology as the input bubble diagrams. Some of the results are even better than the corresponding ground truth room layouts in the RPLAN dataset. As is shown in Figure 48, the author randomly selects several generated results with the bubble diagrams in the test split and illustrate them with a unified rendering setting.

A qualitative comparison of the proposed method and two open-source SOTA methods, HouseGAN++ [4]) and WallPlan (Sun *et al.* 2022) is illustrated in Figure 48. Although WallPlan (Sun *et al.* 2022) could complete this task with tricks, it does not generate new boundaries outside the RPLAN dataset. As is shown in Figure 48, the results of HouseGAN++[4]) are not so satisfying. Since the position of each room is not used in HouseGAN++ [4]), some of the cases show an obvious discrepancy with the given bubble diagrams as well as the ground truths. The proposed methods show better results in generating both diverse room shapes and more plausible room layouts.

In quantitative metrics shown in Table 6, HouseGAN++ [4]) has the worst FID score (52.37) in all methods, and the proposed FID (0.88) in this task surpasses HouseGAN++ [4] by a large margin. The WallPlan (Sun *et al.* 2022) retrievals existing boundaries from the dataset in this task, so its FID would be lower in theory. However, the FID of the proposed complete version method is still lower than WallPlan (Sun *et al.* 2022). Since the input bubble diagrams of HouseGAN++ [4]) are differently defined from other methods, its GED is significantly higher than other methods.

In order to evaluate the performance of the networks only, the author adds a version without the proposed coordinate's optimization in the post-processing stage. In this experiment, the predicted coordinates are only optimized with the orthogonal constraints, such that the performance of topology similarity only relies on the networks. As is shown in Table 6, even the proposed lite method surpasses other methods a lot on the GED score (0.167). Naturally, the GED of the proposed complete version method is exactly equal to 0 since the proposed generated room layouts have the exact same room topology as the input bubble diagrams.

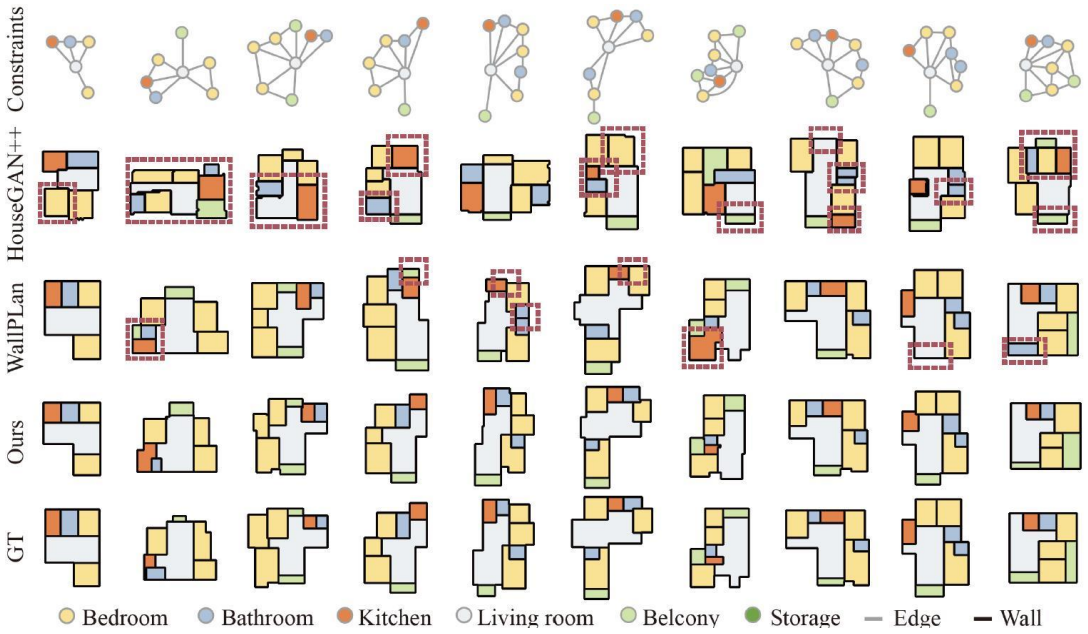


Figure 48 Bubble diagram constrained floorplan generation. Rooms are colored by room type. The author compare the proposed framework with HouseGAN++[4]) and WallPlan (Sun *et al.* 2022). Topology errors (wrong rooms and wrong adjacency) are marked with red dash lines.

Constraints	Method	FID ↓	GED ↓
Bubble diagram	*House-GAN++*	52.37	5.286
	WallPlan	**1.36**	0.618
	ours	1.47	**0.161**
Bubble diagram and boundary	*Graph2Plan*	1.53	0.733
	WallPlan	1.44	0.578
	ours	**1.26**	**0.191**

Table 6 Quantitative evaluation. The author randomly picks 9k samples in the test split in RPLAN (Wu *et al.* 2019) to calculate the FID between generated results with ground truth. The author calculates the mean GED between generated results and the given bubble diagrams on the whole test split.

Figure 49 More generated room layout diagrams with the given bubble diagrams in the test split of RPLAN dataset.

4.2.4 Bubble diagram and boundary-constrained generation

In this section, the proposed framework is tested for floorplan generation with the bubble diagram constraints as well as the fixed house boundary constraints. Similar to the training strategy in the upper task, the author uses ground truth for the boundary registration and the geometry optimization module during training and use predicted results in relevant processes during reasoning.

A qualitative comparison of the proposed framework, Graph2Plan (Hu *et al.*2020), and WallPlan (Sun *et al.* 2022) is illustrated in Figure 50. Compared with the constraints in the upper task, the

boundary constraint in this task is stricter.

As is shown in Figure 50, there is a slight difference between the visualization of all results. At the topological level, ours performs best in all participant methods. The author marks all topological errors with red dash lines in Figure 50. Both Graph2Plan (Hu *et al.*2020) and WallPlan (Sun *et al.* 2022) have some errors in missing rooms and wrong room adjacency. In contrast, the proposed results have the least topological errors.

In the FID comparison, all three methods show evenly matched scores. In the GED comparison, the proposed GED is significantly lower than the other two methods, see Table 6. Compared with FID, GED is a more direct metric reflecting the quality of the generated results under the bubble diagram constraints. Missing rooms and wrong room adjacency are the most common topology errors in other methods that lead to unaccepted floorplans. The proposed insight into decoupling topology and geometry prediction preserves the topology during the whole generation and ensures a much lower GED score.

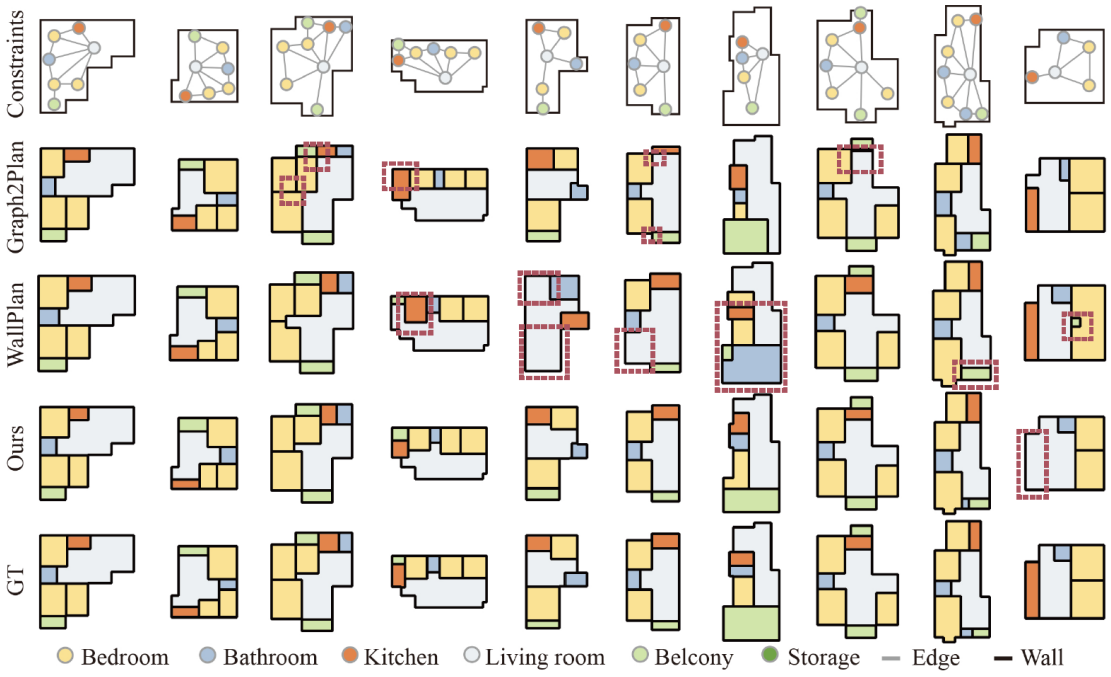


Figure 50 Bubble diagram and boundary constrained floorplan generation. Rooms are colored by type. The author compares the proposed framework with the other two baselines Graph2Plan (Hu *et al.*2020) and WallPlan (Sun *et al.* 2022). Although many results look acceptable, ours have a much more faithful topology to the input bubble diagram. All topology errors (wrong rooms and wrong adjacency) are marked with red dash lines.

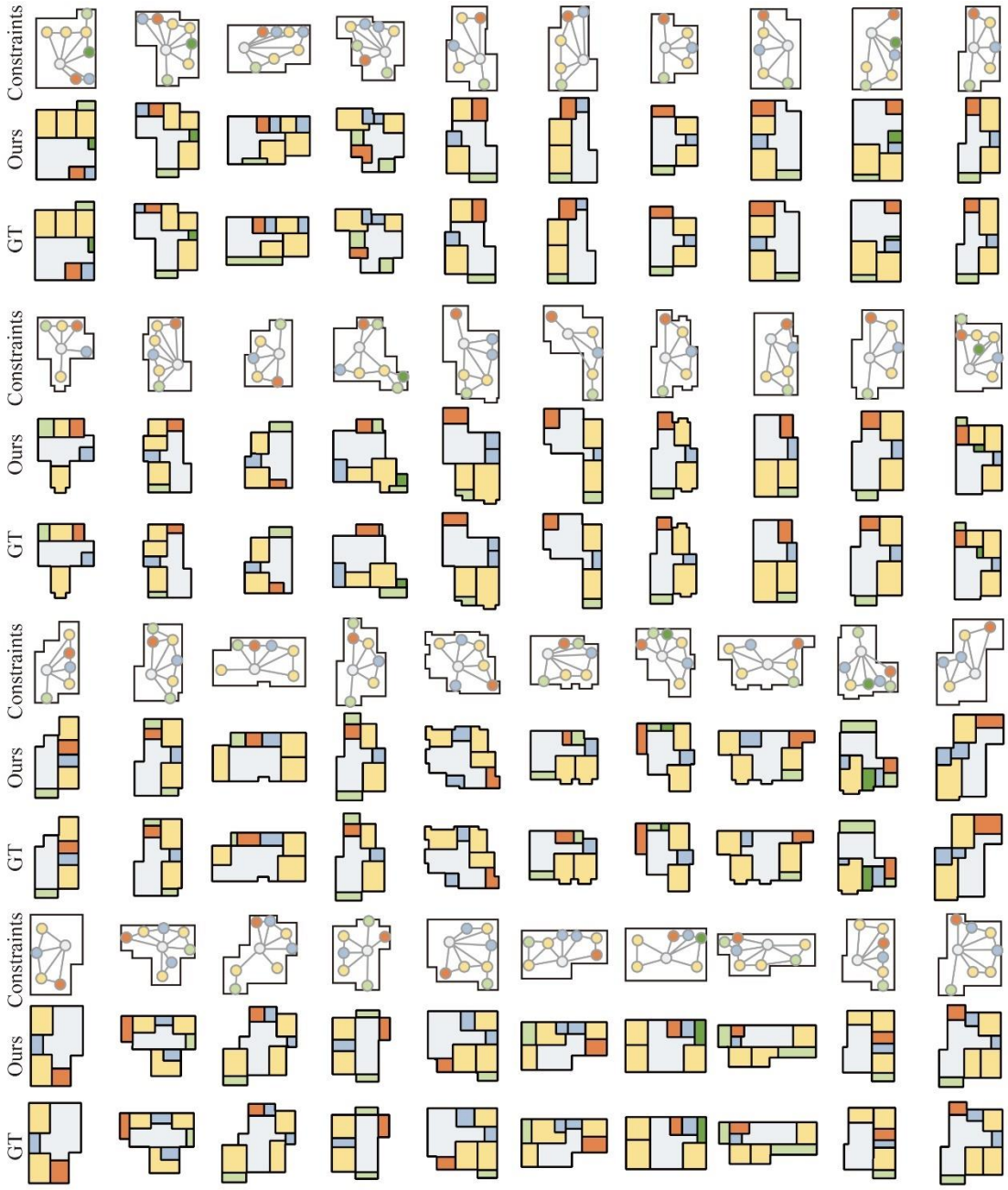


Figure 51 More generated room layout diagrams with the given bubble diagrams and house boundary in the test split of RPLAN dataset.

5 Discussion

5.1 Discussion of floorplan detection

Various controlled experiments are conducted to verify specific design decisions in the proposed GAT-CADNet architecture. Discussion about initial geometric feature selection and the number of GAT stages are also included.

5.1.1 Data augmentation and data unbalanced

As shown in Figure 44, the number of primitives varies in different classes. The number of beds and walls surpasses other classes by a scale in an incongruous way. From the proposed observation, the complicated patterns lead to the rise of bed primitives, while the high number of walls is because walls are a basic element in floorplan CAD drawings. To figure out this long-tail data problem, a natural thought is adding weight to the losses of different classes. However, this method doesn't work well on the proposed data, the PQ, RQ, and SQ with weighted loss are still on the same level and distribution but require more epochs to converge. The proposed explanation is that classes with few samples rarely occur in CAD drawings and the patterns vary hugely from one project to another which may easily bring about over-fitting on the training set.

Data augmentation is another commonly used method to overcome over-fitting. However, it is not easy to add data augmentation to CAD drawings. The reasons are threefold. Firstly, the rigid body transform on a primitive level is harmful, even if the noise is controlled in a small range, which may damage the geometric constraints in CAD drawings. Furthermore, it is also hard to add transform on the instance level and keep the context at the same time. Secondly, adding masks or random dropping on primitives sampling also has both advantages and disadvantages. For classes with complicated patterns, it would be helpful, but on the contrary cases, it is harmful. Third is that for some classes there are severe differences of distribution on the training set and testing set. For these cases, data augmentation helps little in coming up with the theory.

5.1.2 Ablations for the RSE and CEE module

The baseline architecture of the proposed model is the multi-stage GAT branch in the middle of Figure 29. Following the black arrows in Figure 29, it takes initial vertex and edge features and maps to the semantic and instance heads. The blue branch in Figure 29 is the RSE module that attaches relative spatial relation to the vertex attention in every GAT stage. Adding the RSE module to the baseline shows clear improvement in both recognition and segmentation quality by 4 and 5 percentage points respectively, as shown in the third row in Table 7.

It is evident that the explicitly encoded primitive spatial relations, e.g., parallelism and orthogonality, enhance vertex attention and thus yield better performance in panoptic recognition. The proposed CEE module is the orange branch in Figure 29, which views attention among vertices as affinity in

feature space and cascades them to predict the instance adjacency matrix.

Adding the CEE module to the baseline boosts the RQ metric up to 6 percentage points as shown in the fifth row in Figure 29. It proves that the CEE module is able to gather connections between vertices effectively and assist in collecting primitives of the same instance. If the author adds both RSE and CEE modules to the baseline, the proposed method achieves state-of-the-art performance, which exceeds PanCADNet (Fan *et al.* 2021) in RQ, SQ, and PQ metrics by 14.7, 7.6, and 18.4 percentage points respectively.

To further verify the cascaded structure in CEE, the author takes attention scores from only one GAT stage and test their performance. Specifically, the attention in the 2nd, 4th, 6th, and 8th GAT stages are fed to the instance head separately. Statistics listed in Table 7 (sixth to eighth row) show steady improvement in the RQ metric, indicating that higher-level information is gathered from the deeper GAT stage. The proposed cascaded structure can merge multi-stage local and global features for instance symbol spotting.

Model	RSE	CEE	RQ	SQ	PQ
PanCADNet	-	-	0.660	0.838	0.553
baseline			0.687	0.875	0.602
b. + RSE	✓		0.734	0.891	0.654
b. + CEE		✓	0.749	0.896	0.671
	✓	2nd	0.761	0.903	0.687
	✓	4th	0.768	0.903	0.694
	✓	6th	0.768	0.904	0.695
	✓	8th	0.786	0.908	0.714
Ours	✓	✓	**0.807**	**0.914**	**0.737**

Table 7 Ablation study of different network configurations. Numbers in the CEE column represents the nth GAT stage.

5.1.3 Time complexity

For large CAD drawings, the author splits them into fixed-size patches of $10m \times 10m$ to fit into the network, following the practice in PanCADNet (Fan *et al.* 2021) to curb the computational complexity.

Compared to the basic GAT (Velickovic *et al.* 2017) configuration, the author has two extra RSE and CEE modules. For an attention head in each stage, the time complexity of the linear mapping RSE module is $O(|E|F)$, where $|E|$ is the number of edges in the graph and F is the length of input features. The CEE module accumulates attention scores from the previous stage and the time complexity is $O(|E|)$.The complexity of an attention head is $O(|V|FF + |E|F)$ as described in GAT (Velickovic *et al.* 2017), where $|V|$ is the number of vertices in the graph.

Therefore, the complexity of the proposed GAT stage is also $O(|V|FF + |E|F)$. In the research, the dimension notation $N \times N$ is used to hide the implementation detail of the neighboring mask for simplicity and does not indicate $O(N^2)$ complexity. Since at most K edges are allowed for each vertex, $|E| \leq K|V|$, the computational complexity grows linearly as the number of primitives

increases.

5.1.4 Attention-based layer for regularity features

Theoretically, the parallel and orthogonal indicators are redundant if the author has the angle between two vertices. However, if the author drops the regularity term in the initial edge features, the RQ, SQ, and PQ metrics decrease to 0.58, 0.85, and 0.49 respectively. This suggests that the regularities in CAD drawings are essential to recognizing symbols and the proposed extra geometric regularity properties help the network to find a better solution.

The author also tests the effect on different numbers of GAT stages. The number of GAT stages is configured from 2 to 16 and the results are plotted in Figure 52. As the number of stages increases, the performance gets better. However, if the number of stages reaches 16, the proposed network does not benefit from it.

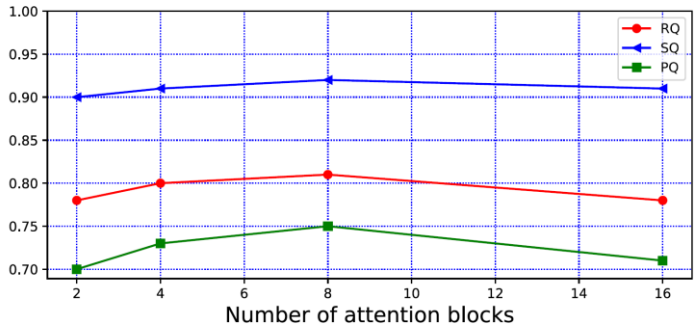


Figure 52 A diagram of the perfomance with different numbers of attention blocks in CAD-CADnet.

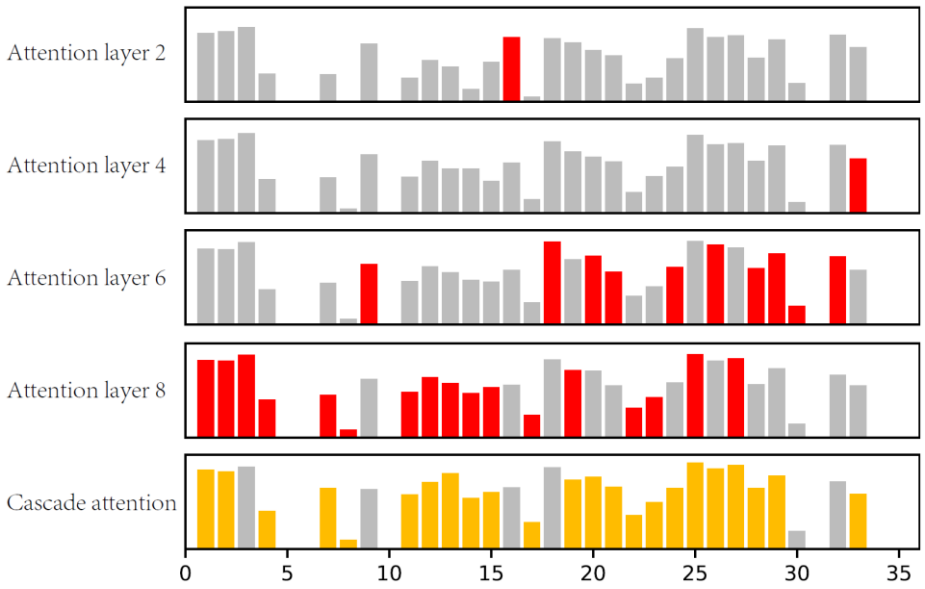


Figure 53 The metric of each classes with different attention layer numbers, and w/o the cascade attention.

Two more extra experiments are conducted to further prove the superiority of the proposed model. One is using a graph convolution network (GCN) as the baseline. Like GAT, GCN is also a widely used graph neural network. Thus, the author replaces GAT stages in the proposed model with GCN stages and take the normalized Laplacian matrix equivalently to one head attention score in the proposed RSE and CEE modules. Another is using vertices features only. Once the center coordinates are added to current vertices features, the spatial relationship can be figured out with any two given segments. To verify the necessity of explicitly encoded edge features in the proposed model, the author conducts another experiment with only vertices features. As shown in Table 8, neither of the two extra experiments reaches the performance of the proposed baseline, let alone the proposed best model.

Model	RQ	SQ	PQ
our baseline	0.687	0.875	0.602
GCN based	0.655	0.859	0.563
w/o edge features	0.599	0.850	0.509
ours	**0.807**	**0.914**	**0.737**

Table 8 Extra experiments. GCN based model and GAT without edge features show inferior results.

5.1.5 Limitations

The proposed CAD-CATnet treats the instance symbol spotting as a subgraph detection problem, with proposed RSE and CEE modules, surpassing existing state-of-the-art methods by a large margin. There are still limitations. The author selects some failed cases and illustrate them in Figure 54. In some cases, simple symbols could be missing or wrongly recognized with mistaken labels or larger variations in the graph, e.g., the proposed model misses all L shape tables (upper) and recognizes all windows as curtain walls (lower) by mistake. Future work would be focusing on failed cases and improving the robustness of the proposed model.

In this subsection, the author provides several typical failed cases and classified them into two situations, i.e., overlapping, and local ambiguity, as is shown in Figure 54. When two instances overlapped (b), (d), and (i), the author found that the confidence score would be decreased acutely compared with the cases with clear boundaries. The author infers the reason is that it is hard to set a threshold to figure out ideal geometric constraints of primitives in overlapped areas and bring more noise to edge feature spaces. There is another possibility that during adjacency prediction in the instance head, the overlapping cases are too few to contribute a valid percentage in instance loss. For the first reason the author will try a more robust way to extract geometric constraints features, and for the second, the author will consider adding weight on loss or introducing feasible data augmentation.

The local ambiguity cases refer to instances that have similar geometric patterns in the local field. There are several situations that may lead to ambiguity. Instances like tables, chairs, and sofas, are sometimes represented with very simple patterns (c), (j), and (k). There is little difference among some certain cases of the table and half-high closets (a) and (l). Also, the symbols are very similar between the curtain wall and handrail (f) and (g) as well as the situations of the air conditioner and

windows (e) and (h). Some patterns are even isomorphic on the graph, which requires a higher capacity for representation learning of the model. Part of the above cases could be distinguished when adding stronger priories by formulating architectural and civil construction knowledge. The author will also consider conducting networks with higher ability and make more use of global context in the scenes.

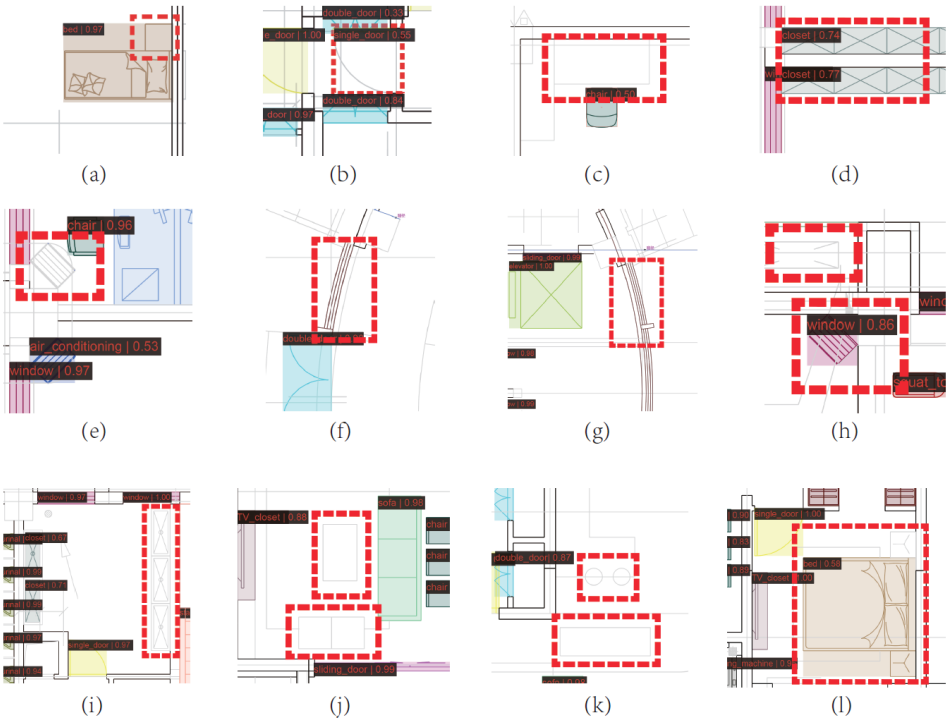


Figure 54 Failed cases of CAD-GATnet.

5.2 Discussion of floorplan generation

In this section, an extensive ablation study is performed to validate the proposed structure of the proposed Subdivision GNN network. The author also analyzes the time complexity of each step in the proposed method. At last, the author illustrates the failed cases and discuss the limitation of the proposed methods.

5.2.1 Perceptual study

Besides FID and GED, the author also recruits 20 volunteers to conduct a subjective comparison as a complementary metric. The participants include professional architects and students studying architecture design at university. The author follows a similar experiment setting of the subjective comparison in HouseGAN++ (Nauata *et al.* 2021).

The author randomly picks 200 samples from each method listed in Table 9, and the total 1200 samples are evenly mixed. Each volunteer will be assigned 30 samples. The participants are asked

to choose the better floorplan between the generated results and the corresponding ground truth with given bubble diagrams. A method scores (+1) for a win, (-1) for a loss, and (0) for a tie. The author suggests the participants consider the level of user requirements satisfaction and the reasonableness of the floor plans during scoring. To reduce the irrelevant influence, method and ground truth labels are hidden from the participants. The author calculates the average score as the metric of realism to measure the similarity between the ground truth and the floorplans generated with different methods.

As is shown in Table 9, ours is the only method with a positive score, which means the proposed results are slightly better than the ground truth in the subjective comparison. In the bubble diagram constrained generation task, the proposed score (0.08) surpasses HouseGAN++ (Nauata *et al.* 2021) (-0.66) by a great margin. The author deduces the reason that the proposed score could surpass the WallPlan (Sun *et al.* 2022[87]) (-0.10) is the proposed results can better satisfy the user requirements.

Constraints	Method	Mean score ↑	σ
Bubble diagram	*House-GAN++*	-0.66	0.86
	WallPlan	-0.10	0.79
	ours	**0.03**	0.69
Bubble diagram and boundary	*Graph2Plan*	-0.36	0.87
	WallPlan	-0.16	0.74
	ours	**-0.11**	**0.66**

Table 9 The author conducts a subjective comparison with 20 architects. σ is the standard deviation of each method. In the case of the similar realism scores, the lower the σ, the better the performance. Compared with other methods, the proposed results have the highest realism scores.

5.2.2 Ablations for the feature fusion backbone

In this section, the author conducts an ablation study to evaluate the efficiency of the proposed feature fusion backbone blocks as well as the generalization performance of the proposed method.

The author takes the graph subdivision task for backbone architecture comparison, since the precision, recall, and F_1 score could intuitively show the performance of each structure. In the comparison, the author chooses classic graph network structures Graph Convolutional Network (kipf *et al.* 2016) and Graph Attention Network (Velickovic *et al.* 2017) (GCN and GAT) as the proposed baselines. For GCN and GAT, the author takes dual edges as vertices and their intersecting relationship as edges to construct graphs. After feature encoding, the subdivisions are predicted with the same head module.

As is shown in Table 10, the proposed module has the best scores. The author infers that the representation ability on the combinatorial embedding of oriented planar graphs is improved with the proposed module so that features defined on dual vertex space and dual edge space could be fused.

Constraints	Network	Precision	Recall	F_1 score
Bubble diagram	*GCN*	0.824	0.836	0.828
	GAT	0.848	0.860	0.852
	Ours	**0.876**	**0.886**	**0.879**
Bubble diagram and Boundary	*GCN*	0.923	0.925	0.923
	GAT	0.956	0.957	0.956
	Ours	**0.975**	**0.976**	**0.975**

Table 10 Ablation study of different graph network architecture used in dual graph subdivision task. Networks with the proposed feature fusion module have the best performance on subdivision prediction.

5.2.3 Generalization performance

Meanwhile, the author also tests the generalization performance of the proposed method. The author conducts an experiment that uses bubble diagrams outside the RPLAN dataset. As is shown in Figure 56, the proposed method generates plausible room layouts and shows good performance on generalization.

The author conducts experiments with manually designed constraints outside the RPLAN dataset (Wu *et al.* 2019). As is shown in Figure 56, (a) is the bubble diagram and GT of a randomly chosen sample in RPLAN (Wu *et al.* 2019). (b) has the same graph as the bubble diagram but with different 2D embedding. Constraints in (c) are the given boundary and the differently embedded bubble diagram. In (d), the boundary is consistent, and the bubble diagram is a new one. The plausible generated floorplans in Figure 55 illustrate the generalization performance of the proposed framework. More generated results are shown in Figure 56.

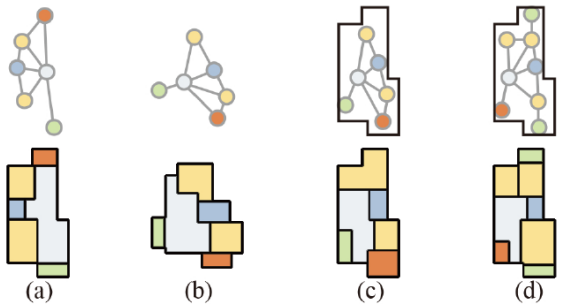


Figure 55 Ablation study of generalization performance. (a) constraints and GT. (b) same bubble diagram, different embedding. (c) same boundary, bubble diagram with different positions. (d) same boundary, bubble diagram with extra nodes and edges.

Figure 56 Generalization ability of Dualgraph2Plan.

5.2.4 Time complexity

In this section, the author analyzes the time complexity of operations in each step of the proposed neural-guided method and illustrate them in Table 11.

Step	Operation	Time complexity
Subdivision GNN	MLP	$\mathcal{O}(\lvert E\rvert d^2 + \lvert V\rvert d^2)$
	Feature fusion block	$\mathcal{O}(2\lvert E\rvert\lvert V\rvert d + \lvert E\rvert d^2 + \lvert V\rvert d^2)$
	Classification head	$\mathcal{O}(\lvert E\rvert d)$
Coordinate prediction	MLP	$\mathcal{O}(d^2)$
	Feature fusion block	$\mathcal{O}(2\lvert E\rvert\lvert V\rvert d + \lvert E\rvert d^2 + \lvert V\rvert d^2)$
	Regression head	$\mathcal{O}(\lvert V\rvert d)$
Coordinate optimization	Orthogonal optimization	$\mathcal{O}(f(6\lvert E\rvert)(\lvert F\rvert + \lvert V\rvert))$
	Rectangle division	$\mathcal{O}(\lvert F_R\rvert p^2)$
	Compact layout drawing	$\mathcal{O}(f(\lvert E_R\rvert)(\lvert F_R\rvert))$
	Coordinates optimization	$\mathcal{O}(\lvert V\rvert^2)$

Table 11 Time complexity analysis of the DualGraph2Plan.

In Table 11, $|V|$, $|E|$, and $|F|$ are the number of vertices, edges, and faces in the dual graph of the given bubble diagram, $d = 128$ is the dimension of the feature space in the networks, and p is the edge number of one face in the dual graph. $|E_R|$, $|F_R|$ are the number of edges and faces in the rectangular orthogonal representation of the dual graph and f is the flow in the defined network in the minimal cost flow problem.

Since the flows in the orthogonal optimization and the compact layout drawing are usually far below 128, the overall time complexity of the proposed method is $\mathcal{O}(2|E||V|d + |E|d^2 + |V|d^2)$, which keeps the same level with other classic graph neural network methods.

5.2.5 Limitations

In this section, the author proposes several types of room layouts, which theoretically cannot be generated with the proposed methods. At the same time, all generated results are browsed, and typical failed cases are extracted for illustration.

According to the three assumptions the author adopted in the duality construction, the author constructs several infeasible situations which violate the assumptions. These situations are shown in Figure 57. Case (a) has slash line walls, which conflict with the Manhattan Assumption. Case (b) violates the second assumption, in which the living room has holes instead of a simply connected polygon. The bubble diagram of case (c) is not a connected graph, which may lead to errors in the proposed optimization step. In case (d), the isolation wall of two bedrooms is discontinuous, which violates the third assumption. Besides, since the author sets the maximal subdivision number as 15, cases like (e) having edges with more than 15 turns cannot be generated with the proposed Subdivision GNN.

During browsing all generated room layouts in the test split, the author also finds limitations of the proposed method. The typical cases are extracted and illustrated in Figure 57. In (a), (b), (c), and (d), even though the topology is consistent, there are smaller rooms in the room layouts, which is unreasonable in architectural design. Besides, the boundaries of (e) and (f) are too complex, which may cause corner space in room layouts. Fortunately, this limitation could be solved by adding interactive steps to set more constraints during the coordinates' optimization.

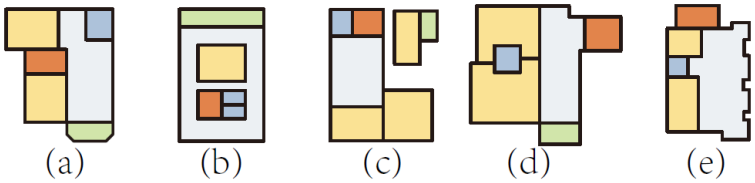


Figure 57 limitations and failed cases of Dualgraph2Plan.

6 Conclusions

6.1 Summary

In this dissertation, the author presents an intuitive yet effective method for floorplan CAD drawing representation. The corresponding network structures make full use of the geometry and topology information on the floorplan. The line graph successfully embeds the geometric primitives and the relationship among them, and the transformer-based structure can effectively extract semantic features. With further assumptions, the dual graph decouples the topology and geometry of the room layouts and predicts them with Subdivision GNN and neural-guided planar drawing method respectively. The feature fusion module could couple features on different spaces with the factorization of the Laplacian in discrete exterior calculus. The neural-guided drawing methods generate plausible orthogonal floor plan drawings with consistent topology.

The significance of this research is twofold. On one hand, the proposed method provides a novel representation and network structure for CAD drawings and reaches better performance in both floorplan detection and room layout generation tasks. The representation method and the network structure are applied and tested with floorplan detection and generation tasks. In the detection task, the performance of the method surpasses other current image or graph-based deep learning method and reach the highest metric scores in the result comparison. In the generation task, compared with other approaches, the proposed results achieve better FID levels with SOTA and reduce GED to 0. The method in this dissertation also achieves the best evaluation in subjective comparison.

On the other hand, the definition and formulation of the two tasks are closer to the scenarios in architectural design. In floorplan detection, primitive-grained labels and instances better fit the input information used for 3D reconstruction. In floorplan generation, the deviation of the geometry prediction is inevitable due to the principle of the neural network, which may lead to topology errors in the generation. In practice, topology errors may cause confusion in architectural usage, which is unacceptable. The proposed hybrid optimization can ensure that the topology is completely correct.

6.2 Future work

Although the representation method is validated to be efficient with the results in the two applications, there are still issues that can be improved with further research. The representation method can cover most cases with the assumptions for graph definition, while floorplans with more complicated topology, strong global context, or curve primitives still need further improved representation methods.

The improvement in the CAD drawing representation could be threefold. The first direction could be the capacity of the network scales since there are many public architectures, whose floorplans are complicated and have large areas. The larger and deeper the network is, the higher the representation ability the network would have for the oversized floorplans. Meanwhile, the corresponding network structure should be designed with good embedding performances on both

local geometric features and global context features.

The ability to represent and generate curved floorplans could be the second possible direction. From the author's perspective, the challenge of this direction could come from two aspects. Although curved floorplans sometimes are more flexible and have better quality than orthogonal floorplans, they are less common in the AEC industries. The limited number of curved floorplans may lead difficulty to establish a decent dataset with adequate samples. Another challenge is the option of the priories for the curves in floorplans. Whether using parametric or discretized expressions, there will be many problems.

The third direction would be the details in applications. As mentioned in Chapters above, details when deploying on specific applications remain lots of issues, such as data augmentation, optimization, post-processing, and metric design. Although some issues are faced in other generic tasks, they still need rethinking in the context of AEC. More reasonable solutions, definitions, or improvements may be figured out by combining deep learning technology and the actual needs of the construction field.

Besides the representation method, other deep learning-related tasks also have great potential in practice. In the proposed future work, the author would also attempt to integrate the subtasks to conduct end-to-end methods in detection, generation, and other CAD drawing-related tasks.

Reference

[1] Arnheim, R. (1997). *Visual thinking*. Univ of California Press.

[2] Zheng, Z., Li, J., Zhu, L., Li, H., Petzold, F., & Tan, P. (2022). GAT-CADNet: Graph Attention Network for Panoptic Symbol Spotting in CAD Drawings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 11747-11756).

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

[4] Nauata, N., Hosseini, S., Chang, K. H., Chu, H., Cheng, C. Y., & Furukawa, Y. (2021). House-gan++: Generative adversarial layout refinement network towards intelligent computational agent for professional architects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 13632-13641).

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

[6] Wang, X. Y., & Zhang, K. (2020). Generating layout designs from high-level specifications. *Automation in Construction*, *119*, 103288.

[7] Vidanapathirana, M., Wu, Q., Furukawa, Y., Chang, A. X., & Savva, M. (2021). Plan2scene: Converting floorplans to 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 10733-10742).

[8] Kim, Y. M., Mitra, N. J., Yan, D. M., & Guibas, L. (2012). Acquiring 3d indoor environments with variability and repetition. *ACM Transactions on Graphics (TOG)*, *31*(6), 1-11.

[9] Lim, J. O. I. E., Janssen, P. A. T. R. I. C. K., & Stouffs, R. U. D. I. (2018, May). Automated generation of BIM models from 2D CAD drawings. In *Proceedings of the 23rd CAADRIA Conference* (pp. 61-70).

[10] Xiao, Y., Chen, S., Ikeda, Y., & Hotta, K. (2020). Automatic recognition and segmentation of architectural elements from 2D drawings by a convolutional neural network. In *25th International Conference on Computer-Aided Architectural Design Research in Asia, CAADRIA 2020* (pp. 843-852). The Association for Computer-Aided Architectural Design Research in Asia (CAADRIA).

[11] Lewis, R., & Séquin, C. (1998). Generation of 3D building models from 2D architectural plans. *Computer-Aided Design*, *30*(10), 765-779.

[12] So, C., Baciu, G., & Sun, H. (1998, November). Reconstruction of 3D virtual buildings from 2D architectural floor plans. In *Proceedings of the ACM symposium on Virtual reality software and technology* (pp. 17-23).

[13] Or, S. H., Wong, K. H., Yu, Y. K., Chang, M. M. Y., & Kong, H. (2005). Highly automatic approach to architectural floorplan image understanding & model generation. *Pattern Recognition*, 25-32.

[14] Yin, X., Wonka, P., & Razdan, A. (2008). Generating 3d building models from architectural drawings: A survey. *IEEE computer graphics and applications*, *29*(1), 20-30.

[15] Nan, L., Xie, K., & Sharf, A. (2012). A search-classify approach for cluttered indoor scene understanding. *ACM Transactions on Graphics (TOG)*, *31*(6), 1-10.

[16] Adan, A., & Huber, D. (2011, May). 3D reconstruction of interior wall surfaces under occlusion and clutter. In *2011 International Conference on 3D Imaging, Modeling, Processing, Visualization*

and Transmission (pp. 275-281). IEEE.

[17] Avetisyan, A., Dahnert, M., Dai, A., Savva, M., Chang, A. X., & Nießner, M. (2019). Scan2cad: Learning cad model alignment in rgb-d scans. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition* (pp. 2614-2623).

[18] Bhunia, A. K., Chowdhury, P. N., Yang, Y., Hospedales, T. M., Xiang, T., & Song, Y. Z. (2021). Vectorization and rasterization: Self-supervised learning for sketch and handwriting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 5672-5681).

[19] Harary, F., & Norman, R. Z. (1960). Some properties of line digraphs. *Rendiconti del circolo matematico di palermo*, *9*, 161-168.

[20] Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., & Çöltekin, A. (2015). Applications of 3D city models: State of the art review. *ISPRS International Journal of Geo-Information*, *4*(4), 2842-2889.

[21] Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., & Vandergheynst, P. (2017). Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, *34*(4), 18-42.

[22] Brugere, I., Gallagher, B., & Berger-Wolf, T. Y. (2018). Network structure inference, a survey: Motivations, methods, and applications. *ACM Computing Surveys (CSUR)*, *51*(2), 1-39.

[23] Budroni, A., & Boehm, J. (2010). Automated 3D reconstruction of interiors from point clouds. *International Journal of Architectural Computing*, *8*(1), 55-73.

[24] Do Carmo, M. P., & Flaherty Francis, J. (1992). *Riemannian geometry* (Vol. 6). Boston: Birkhäuser.

[25] Budroni, A., & Böhm, J. (2009). Toward automatic reconstruction of interiors from laser data. *Proceedings of Virtual Reconstruction and Visualization of Complex Architectures (3D-Arch)*, *36*.

[26] Cai, H., Zheng, V. W., & Chang, K. C. C. (2018). A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE transactions on knowledge and data engineering*, *30*(9), 1616-1637.

[27] Chai, J., Chi, H. L., Wang, X., Wu, C., Jung, K. H., & Lee, J. M. (2016). Automatic as-built modeling for concurrent progress tracking of plant construction based on laser scanning. *Concurrent Engineering*, *24*(4), 369-380.

[28] Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, *30*.

[29] Chen, J., Cho, Y. K., & Kim, K. (2018). Region proposal mechanism for building element recognition for advanced scan-to-BIM process. In *Construction Research Congress 2018* (pp. 221-231).

[30] Chen, J., Kira, Z., & Cho, Y. K. (2019). Deep learning approach to point cloud scene understanding for automated scan to 3D reconstruction. *Journal of Computing in Civil Engineering*, *33*(4), 04019027.

[31] Chen, Q., Wu, Q., Tang, R., Wang, Y., Wang, S., & Tan, M. (2020). Intelligent home 3d: Automatic 3d-house design from linguistic descriptions only. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 12625-12634).

[32] Cho, C. Y., & Liu, X. (2017). An automated reconstruction approach of mechanical systems in building information modeling (BIM) using 2D drawings. In *Computing in Civil Engineering 2017* (pp. 236-244).

[33] Peng, C. H., Yang, Y. L., & Wonka, P. (2014). Computing layouts with deformable templates. *ACM Transactions on Graphics (TOG)*, *33*(4), 1-11.

[34] Couprie, C., Farabet, C., Najman, L., & LeCun, Y. (2013). Indoor semantic segmentation using depth information. *arXiv preprint arXiv:1301.3572*.

[35] Czerniawski, T., Sankaran, B., Nahangi, M., Haas, C., & Leite, F. (2018). 6D DBSCAN-based segmentation of building point clouds for planar object classification. *Automation in Construction*, *88*, 44-58.

[36] Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., & Nießner, M. (2017). Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5828-5839).

[37] Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., & Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

[38] Domínguez, B., García, Á. L., & Feito, F. R. (2012). Semiautomatic detection of floor topology from CAD architectural drawings. *Computer-Aided Design*, *44*(5), 367-378.

[39] Eisenstadt, V., Langenhan, C., & Althoff, K. D. (2019). Generation of Floor Plan Variations with Convolutional Neural Networks and Case-based Reasoning--An Approach for Unsupervised Adaptation of Room Configurations within a Framework for Support of Early Conceptual Design. In *eCAADe SIGraDi Conference, Porto*.

[40] He, F., Huang, Y., & Wang, H. (2022). iPLAN: interactive and procedural layout planning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 7793-7802).

[41] Sun, Y. (2012). *Mining heterogeneous information networks*. University of Illinois at Urbana-Champaign.

[42] Furukawa, Y., Curless, B., Seitz, S. M., & Szeliski, R. (2009, June). Manhattan-world stereo. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1422-1429). IEEE.

[43] Ganin, Y., Bartunov, S., Li, Y., Keller, E., & Saliceti, S. (2021). Computer-aided design as language. *Advances in Neural Information Processing Systems*, *34*, 5885-5897.

[44] Gao, H., & Ji, S. (2019, May). Graph u-nets. In *international conference on machine learning* (pp. 2083-2092). PMLR.

[45] Grover, A., & Leskovec, J. (2016, August). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 855-864).

[46] Fathi, H., Dai, F., & Lourakis, M. (2015). Automated as-built 3D reconstruction of civil infrastructure using computer vision: Achievements, opportunities, and challenges. *Advanced Engineering Informatics*, *29*(2), 149-161.

[47] Maron, H., Litany, O., Chechik, G., & Fetaya, E. (2020, November). On learning sets of symmetric elements. In *International conference on machine learning* (pp. 6734-6744). PMLR.

[48] Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in neural information processing systems*, *30*.

[49] Hamledari, H., McCabe, B., & Davari, S. (2017). Automated computer vision-based detection of components of under-construction indoor partitions. *Automation in Construction*, *74*, 78-94.

[50] Handa, A., Patraucean, V., Badrinarayanan, V., Stent, S., & Cipolla, R. (2016). Understanding real world indoor scenes with synthetic data. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4077-4085).

[51] Mo, H., Simo-Serra, E., Gao, C., Zou, C., & Wang, R. (2021). General virtual sketching framework for vector line art. *ACM Transactions on Graphics (TOG)*, *40*(4), 1-14.

[52] Guo, H., Liu, S., Pan, H., Liu, Y., Tong, X., & Guo, B. (2022). ComplexGen: CAD reconstruction by B-rep chain complex generation. *ACM Transactions on Graphics (TOG)*, *41*(4), 1-18.

[53] Hong, H., Guo, H., Lin, Y., Yang, X., Li, Z., & Ye, J. (2020, April). An attention-based graph neural network for heterogeneous structural learning. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 34, No. 04, pp. 4132-4139).

[54] Horna, S., Meneveaux, D., Damiand, G., & Bertrand, Y. (2009). Consistency constraints and 3D building reconstruction. *Computer-Aided Design*, *41*(1), 13-27.

[55] Hu, R., Huang, Z., Tang, Y., Van Kaick, O., Zhang, H., & Huang, H. (2020). Graph2plan: Learning floorplan generation from layout graphs. *ACM Transactions on Graphics (TOG)*, *39*(4), 118-1.

[56] Hu, Z., Dong, Y., Wang, K., & Sun, Y. (2020, April). Heterogeneous graph transformer. In *Proceedings of the web conference 2020* (pp. 2704-2710).

[57] Huang, W., & Zheng, H. (2018, October). Architectural drawings recognition and generation through machine learning. In *Proceedings of the 38th Annual Conference of the Association for Computer Aided Design in Architecture, Mexico City, Mexico* (pp. 18-20).

[58] Park, J. J., Florence, P., Straub, J., Newcombe, R., & Lovegrove, S. (2019). Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 165-174).

[59] Ren, J., Zhang, B., Wu, B., Huang, J., Fan, L., Ovsjanikov, M., & Wonka, P. (2021). Intuitive and efficient roof modeling for reconstruction and synthesis. *arXiv preprint arXiv:2109.07683*.

[60] Dosch, P., Tombre, K., Ah-Soon, C., & Masini, G. (2000). A complete system for the analysis of architectural drawings. *International Journal on Document Analysis and Recognition*, *3*(2), 102-116.

[61] Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., & Shah, M. (2022). Transformers in vision: A survey. *ACM computing surveys (CSUR)*, *54*(10s), 1-41.

[62] Kipf, T. N., & Welling, M. (2016). Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

[63] Laefer, D. F., & Truong-Hong, L. (2017). Toward automatic generation of 3D steel structures for building information modelling. *Automation in Construction*, *74*, 66-77.

[64] Lahav, A., & Tal, A. (2020). Meshwalker: Deep mesh understanding by random walks. *ACM Transactions on Graphics (TOG)*, *39*(6), 1-13.

[65] Lambourne, J. G., Willis, K. D., Jayaraman, P. K., Sanghi, A., Meltzer, P., & Shayani, H. (2021). Brepnet: A topological message passing system for solid models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 12773-12782).

[66] Li, L., Zou, C., Zheng, Y., Su, Q., Fu, H., & Tai, C. L. (2018). Sketch-R2CNN: An attentive network for vector sketch recognition. *arXiv preprint arXiv:1811.08170*.

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

[68] Liu, C., Wu, J., & Furukawa, Y. (2018). Floornet: A unified framework for floorplan reconstruction from 3d scans. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 201-217).

[69] Liu, H. T. D., Kim, V. G., Chaudhuri, S., Aigerman, N., & Jacobson, A. (2020). Neural subdivision. *arXiv preprint arXiv:2005.01819*.

[70] Lu, Q., & Lee, S. (2017). Image-based technologies for constructing as-is building information

models for existing buildings. *Journal of Computing in Civil Engineering*, *31*(4), 04017005.

[71] Lu, T., Tai, C. L., Su, F., & Cai, S. (2005). A new recognition model for electronic architectural drawings. *Computer-Aided Design*, *37*(10), 1053-1069.

[72] Lu, T., Yang, H., Yang, R., & Cai, S. (2007). Automatic analysis and integration of architectural drawings. *International Journal of Document Analysis and Recognition (IJDAR)*, *9*, 31-47.

[73] Ma, J. W., Czerniawski, T., & Leite, F. (2020). Semantic segmentation of point clouds of building interiors with deep learning: Augmenting training datasets with synthetic BIM-based point clouds. *Automation in Construction*, *113*, 103144.

[74] Merrell, P., Schkufza, E., & Koltun, V. (2010). Computer-generated residential building layouts. In *ACM SIGGRAPH Asia 2010 researchs* (pp. 1-12).

[75] Mura, C., Mattausch, O., Villanueva, A. J., Gobbetti, E., & Pajarola, R. (2014). Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts. *Computers & Graphics*, *44*, 20-32.

[76] Murillo, A. C., Košecká, J., Guerrero, J. J., & Sagüés, C. (2008). Visual door detection integrating appearance and shape cues. *Robotics and Autonomous Systems*, *56*(6), 512-521.

[77] Nauata, N., Chang, K. H., Cheng, C. Y., Mori, G., & Furukawa, Y. (2020). House-gan: Relational generative adversarial networks for graph-constrained house layout generation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16* (pp. 162-177). Springer International Publishing.

[78] Neuhausen, M., & König, M. (2018). Automatic window detection in facade images. *Automation in Construction*, *96*, 527-539.

[79] NOZ-SALINAS, R. M., Aguirre, E., García-Silvente, M., & ALEZ, A. G. (2004). Door-detection using computer vision and fuzzy logic.

[80] Perozzi, B., Al-Rfou, R., & Skiena, S. (2014, August). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 701-710).

[81] Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 652-660).

[82] Volk, R., Stengel, J., & Schultmann, F. (2014). Building Information Modeling (BIM) for existing buildings—Literature review and future needs. *Automation in construction*, *38*, 109-127.

[83] Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., & Welling, M. (2018). Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15* (pp. 593-607). Springer International Publishing.

[84] Seff, A., Ovadia, Y., Zhou, W., & Adams, R. P. (2020). Sketchgraphs: A large-scale dataset for modeling relational geometry in computer-aided design. *arXiv preprint arXiv:2007.08506*.

[85] Seyis, S. (2019). Pros and cons of using building information modeling in the AEC industry. *Journal of Construction Engineering and Management*, *145*(8), 04019046.

[86] Smirnov, D., & Solomon, J. (2021). HodgeNet: Learning spectral geometry on triangle meshes. *ACM Transactions on Graphics (TOG)*, *40*(4), 1-11.

[87] Sun, J., Wu, W., Liu, L., Min, W., Zhang, G., & Zheng, L. (2022). WallPlan: synthesizing floorplans by learning to generate wall graphs. *ACM Transactions on Graphics (TOG)*, *41*(4), 1-14.

[88] Czerniawski, T., & Leite, F. (2020). Automated digital modeling of existing buildings: A review of

visual object recognition methods. *Automation in Construction*, *113*, 103131.

[89] Thomas, H., Qi, C. R., Deschaud, J. E., Marcotegui, B., Goulette, F., & Guibas, L. J. (2019). Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 6411-6420).

[90] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

[91] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. *arXiv preprint arXiv:1710.10903*.

[92] Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer networks. *Advances in neural information processing systems*, *28*.

[93] Pătrăucean, V., Armeni, I., Nahangi, M., Yeung, J., Brilakis, I., & Haas, C. (2015). State of research in automatic as-built modelling. *Advanced Engineering Informatics*, *29*(2), 162-171.

[94] Volk, R., Luu, T. H., Mueller-Roemer, J. S., Sevilmis, N., & Schultmann, F. (2018). Deconstruction project planning of existing buildings based on automated acquisition and reconstruction of building information. *Automation in construction*, *91*, 226-245.

[95] Para, W., Guerrero, P., Kelly, T., Guibas, L. J., & Wonka, P. (2021). Generative layout modeling using constraint graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 6690-6700).

[96] Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., & Yu, P. S. (2019, May). Heterogeneous graph attention network. In *The world wide web conference* (pp. 2022-2032).

[97] Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., & Solomon, J. M. (2019). Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, *38*(5), 1-12.

[98] Willis, K. D., Pu, Y., Luo, J., Chu, H., Du, T., Lambourne, J. G., ... & Matusik, W. (2021). Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences. *ACM Transactions on Graphics (TOG)*, *40*(4), 1-24.

[99] Wu, W., Fu, X. M., Tang, R., Wang, Y., Qi, Y. H., & Liu, L. (2019). Data-driven interior plan generation for residential buildings. *ACM Transactions on Graphics (TOG)*, *38*(6), 1-12.

[100] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, *32*(1), 4-24.

[101] Wu, W., Fan, L., Liu, L., & Wonka, P. (2018, May). Miqp-based layout design for building interiors. In *Computer Graphics Forum* (Vol. 37, No. 2, pp. 511-521).

[102] Yang, Y. L., Wang, J., Vouga, E., & Wonka, P. (2013). Urban pattern: Layout design by hierarchical domain splitting. *ACM Transactions on Graphics (TOG)*, *32*(6), 1-12.

[103] Zeng, Z., Li, X., Yu, Y. K., & Fu, C. W. (2019). 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* (pp. 9096-9104).

[104] Zhang, F., Nauata, N., & Furukawa, Y. (2020). 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* (pp. 2798-2807).

[105] Zhao, H., Jiang, L., Jia, J., Torr, P. H., & Koltun, V. (2021). Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 16259-16268).

[106] Zhu, J., Zhang, H., & Wen, Y. (2014). A new reconstruction method for 3D buildings from 2D vector floor plan. *Computer-aided design and applications*, *11*(6), 704-714.

[107] Garland, M., & Heckbert, P. S. (1997, August). Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (pp. 209-216).

[108] Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

[109] Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

[110] Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., ... & Ahmed, A. (2020). Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, *33*, 17283-17297.

[111] Smirnov, D., & Solomon, J. (2021). HodgeNet: Learning spectral geometry on triangle meshes. *ACM Transactions on Graphics (TOG)*, *40*(4), 1-11.

[112] Desbrun, M., Hirani, A. N., Leok, M., & Marsden, J. E. (2005). Discrete exterior calculus. *arXiv preprint math/0508341*.

[113] Hocking, J. G., & Young, G. S. (1988). *Topology*. Courier Corporation.

[114] Nishizeki, T., & Rahman, M. S. (2004). *Planar graph drawing* (Vol. 12). World Scientific.

[115] Tamassia, R. (1987). On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing*, *16*(3), 421-444.

[116] Garg, A., & Tamassia, R. (1997). A new minimum cost flow algorithm with applications to graph drawing. In *Graph Drawing: Symposium on Graph Drawing, GD'96 Berkeley, California, USA, September 18–20, 1996 Proceedings 4* (pp. 201-216). Springer Berlin Heidelberg.

[117] Kirillov, A., He, K., Girshick, R., Rother, C., & Dollár, P. (2019). Panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 9404-9413).

[118] Liggett, R. S. (2000). Automated facilities layout: past, present and future. *Automation in construction*, *9*(2), 197-215.

[119] Arvin, S. A., & House, D. H. (2002). Modeling architectural design objectives in physically based space planning. *Automation in Construction*, *11*(2), 213-225.

[120] Medjdoub, B., & Yannou, B. (2000). Separating topology and geometry in space planning. *Computer-aided design*, *32*(1), 39-61.

[121] Michalek, J., Choudhary, R., & Papalambros, P. (2002). Architectural layout design optimization. *Engineering optimization*, *34*(5), 461-484.

[122] Liu, H., Yang, Y. L., AlHalawani, S., & Mitra, N. J. (2013). Constraint-aware interior layout exploration for pre-cast concrete-based buildings. *The Visual Computer*, *29*, 663-673.

[123] Harada, M., Witkin, A., & Baraff, D. (1995, September). Interactive physically-based manipulation of discrete/continuous models. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (pp. 199-208).

[124] Yeap, K. H., & Sarrafzadeh, M. (1993). Floor-planning by graph dualization: 2-concave rectilinear modules. *SIAM Journal on Computing*, *22*(3), 500-526.

[125] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, *63*(11), 139-144.

[126] Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*.

[127] Zhang, H., Goodfellow, I., Metaxas, D., & Odena, A. (2019, May). Self-attention generative

adversarial networks. In *International conference on machine learning* (pp. 7354-7363). PMLR.

[128] Brock, A., Donahue, J., & Simonyan, K. (2018). Large scale GAN training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*.

[129] Karras, T., Laine, S., & Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 4401-4410).

[130] Li, J., Yang, J., Hertzmann, A., Zhang, J., & Xu, T. (2020). Layoutgan: Synthesizing graphic layouts with vector-wireframe adversarial networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *43*(7), 2388-2399.

[131] Zhang, M., Cui, Z., Neumann, M., & Chen, Y. (2018, April). An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 32, No. 1).

[132] Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., & Leskovec, J. (2018). Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, *31*.

[133] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2008). The graph neural network model. *IEEE transactions on neural networks*, *20*(1), 61-80.

[134] Dosch, P., Tombre, K., Ah-Soon, C., & Masini, G. (2000). A complete system for the analysis of architectural drawings. *International Journal on Document Analysis and Recognition*, *3*(2), 102-116.

[135] Nguyen, T. O., Tabbone, S., & Terrades, O. R. (2008, September). Symbol descriptor based on shape context and vector model of information retrieval. In *2008 The Eighth IAPR International Workshop on Document Analysis Systems* (pp. 191-197). IEEE.

[136] Nguyen, T. O., Tabbone, S., & Boucher, A. (2009, July). A symbol spotting approach based on the vector model and a visual vocabulary. In *2009 10th International Conference on Document Analysis and Recognition* (pp. 708-712). IEEE.

[137] Rusinol, M., Lladós, J., & Sánchez, G. (2010). Symbol spotting in vectorized technical drawings through a lookup table of region strings. *Pattern Analysis and Applications*, *13*, 321-331.

[138] Medjdoub, B., & Yannou, B. (2000). Separating topology and geometry in space planning. *Computer-aided design*, *32*(1), 39-61.

[139] Zhi, G. S., Lo, S. M., & Fang, Z. (2003). A graph-based algorithm for extracting units and loops from architectural floor plans for a building evacuation model. *Computer-Aided Design*, *35*(1), 1-14.

[140] Dutta, A., Lladós, J., Bunke, H., & Pal, U. (2013, August). Near convex region adjacency graph and approximate neighborhood string matching for symbol spotting in graphical documents. In *2013 12th International Conference on Document Analysis and Recognition* (pp. 1078-1082). IEEE.

[141] Dutta, A., Lladós, J., & Pal, U. (2013). A symbol spotting approach in graphical documents by hashing serialized graphs. *Pattern Recognition*, *46*(3), 752-768.

[142] Domínguez, B., García, Á. L., & Feito, F. R. (2012). Semiautomatic detection of floor topology from CAD architectural drawings. *Computer-Aided Design*, *44*(5), 367-378.

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

[144] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic

segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431-3440).

[145] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 2961-2969).

[146] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, *28*.

[147] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

[148] Sun, Z., Wang, C., Zhang, L., & Zhang, L. (2012). Free hand-drawn sketch segmentation. In *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part I 12* (pp. 626-639). Springer Berlin Heidelberg.

[149] Li, L., Fu, H., & Tai, C. L. (2018). Fast sketch segmentation and labeling with deep learning. *IEEE computer graphics and applications*, *39*(2), 38-51.

[150] Huang, Z., Fu, H., & Lau, R. W. (2014). Data-driven segmentation and labeling of freehand sketches. *ACM Transactions on Graphics (TOG)*, *33*(6), 1-10.

[151] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., ... & Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI open*, *1*, 57-81.

[152] Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, *29*.

[153] Atwood, J., & Towsley, D. (2016). Diffusion-convolutional neural networks. *Advances in neural information processing systems*, *29*.

[154] Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., & Solomon, J. M. (2019). Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, *38*(5), 1-12.

[155] Horna, S., Meneveaux, D., Damiand, G., & Bertrand, Y. (2009). Consistency constraints and 3D building reconstruction. *Computer-Aided Design*, *41*(1), 13-27.

[156] Yang, L., Zhuang, J., Fu, H., Wei, X., Zhou, K., & Zheng, Y. (2020). SketchGNN: Semantic Sketch Segmentation with Graph Neural Networks. *arXiv preprint arXiv:2003.00678*.

[157] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

[158] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... & Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 10012-10022).

[159] Guo, Q., Qiu, X., Liu, P., Shao, Y., Xue, X., & Zhang, Z. (2019). Star-transformer. *arXiv preprint arXiv:1902.09113*.

[160] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16* (pp. 213-229). Springer International Publishing.

[161] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017, July). Neural message passing for quantum chemistry. In *International conference on machine learning* (pp. 1263-1272). PMLR.

[162] Fan, Z., Chen, T., Wang, P., & Wang, Z. (2022). Cadtransformer: Panoptic symbol spotting

transformer for cad drawings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 10986-10996).

[163] Trudeau, R. J. (2013). *Introduction to graph theory*. Courier Corporation.

[164] Rusiñol, M., Borràs, A., & Lladós, J. (2010). Relational indexing of vectorial primitives for symbol spotting in line-drawing images. *Pattern Recognition Letters*, *31*(3), 188-201.

[165] Delalandre, M., Valveny, E., Pridmore, T., & Karatzas, D. (2010). Generation of synthetic documents for performance evaluation of symbol recognition & spotting systems. *International Journal on Document Analysis and Recognition (IJDAR)*, *13*(3), 187-207.

[166] Lifull home's dataset, https://www.nii.ac.jp/dsc/idr/lifull

[167] Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., ... & Lin, D. (2019). MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*.

[168] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https ://github.com/open-mmlab/mmsegmentation

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

[170] Chen, L. C., Papandreou, G., Schroff, F., & Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*.

[171] Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., ... & Xiao, B. (2020). Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, *43*(10), 3349-3364.

[172] Chen, L. C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 801-818).

[173] Rezvanifar, A., Cote, M., & Albu, A. B. (2020). Symbol spotting on digital architectural floor plans using a deep learning-based framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (pp. 568-569).

[174] Tian, Z., Shen, C., Chen, H., & He, T. (2019). Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 9627-9636).

[175] Google, ortools, https://developers.google.com/optimization

[176] J. D. Martin Andersen, L. Vandenberghe, cvxopt, https:// https://cvxopt.org/

[177] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, *30*.

[178] Abu-Aisheh, Z., Raveaux, R., Ramel, J. Y., & Martineau, P. (2015, January). An exact graph edit distance algorithm for solving pattern recognition problems. In *4th International Conference on Pattern Recognition Applications and Methods 2015*.

[179] Deng, L., & Yu, D. (2014). Deep learning: methods and applications. *Foundations and trends® in signal processing*, *7*(3–4), 197-387.

[180] Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, *2*(1), 1-127.

[181] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 1798-1828.

[182] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436-444.

[183] Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, *36*(4), 193-202.

[184] Werbos, P. (1974). Beyond regression: New tools for prediction and analysis in the behavioral sciences. *PhD thesis, Committee on Applied Mathematics, Harvard University, Cambridge, MA*.

[185] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1*(4), 541-551.

[186] Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13* (pp. 818-833). Springer International Publishing.

[187] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

[188] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[189] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84-90.

[190] Dilokthanakul, N., Mediano, P. A., Garnelo, M., Lee, M. C., Salimbeni, H., Arulkumaran, K., & Shanahan, M. (2016). Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*.

[191] Hsu, W. N., Zhang, Y., & Glass, J. (2017, December). Unsupervised domain adaptation for robust speech recognition via variational autoencoder-based data augmentation. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)* (pp. 16-23). IEEE.

[192] Ehsan Abbasnejad, M., Dick, A., & van den Hengel, A. (2017). Infinite variational autoencoder for semi-supervised learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5888-5897).

[193] Xu, W., Sun, H., Deng, C., & Tan, Y. (2017, February). Variational autoencoder for semi-supervised text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 31, No. 1).

[194] Kanungo, T., Haralick, R. M., & Dori, D. (1995, August). Understanding engineering drawings: A survey. In *Proceedings of First IARP Workshop on Graphics Recognition* (pp. 217-228).

[195] Filipski, A. J., & Flandrena, R. (1992). Automated conversion of engineering drawings to CAD form. *Proceedings of the IEEE*, *80*(7), 1195-1209.

[196] Truong-Hong, L., Laefer, D. F., Hinks, T., & Carr, H. (2012). Flying voxel method with Delaunay triangulation criterion for façade/feature detection for computation. *Journal of Computing in Civil Engineering*, *26*(6), 691-707.

[197] Noack, R. (2001). *Converting CAD drawings to product models* (Doctoral dissertation, Institutionen för fastigheter och byggande).

[198] Rahbar, M., Mahdavinejad, M., Markazi, A. H., & Bemanian, M. (2022). Architectural layout design through deep learning and agent-based modeling: A hybrid approach. *Journal of Building Engineering*, *47*, 103822.

[199] Intwala, A. (2020). Image to CAD: feature extraction and translation of raster image of CAD drawing to DXF CAD format. In *Computer Vision and Image Processing: 4th International*

Conference, CVIP 2019, Jaipur, India, September 27–29, 2019, Revised Selected Researchs, Part I 4 (pp. 205-215). Springer Singapore.

[200] Dori, D., & Tombre, K. (1995). From engineering drawings to 3D CAD models: are the author ready now?. *Computer-Aided Design*, *27*(4), 243-254.

[201] Barki, H., Fadli, F., Shaat, A., Boguslawski, P., & Mahdjoubi, L. (2015). BIM models generation from 2D CAD drawings and 3D scans: an analysis of challenges and opportunities for AEC practitioners. *Building Information Modelling (BIM) in Design, Construction and Operations*, *149*, 369-380.

[202] Yang, B., Liu, B., Zhu, D., Zhang, B., Wang, Z., & Lei, K. (2020). Semiautomatic structural BIM-model generation methodology using CAD construction drawings. *Journal of Computing in Civil Engineering*, *34*(3), 04020006.

[203] Ernst, J., & Roddis, W. M. (1994). Checking of CAD drawings for fabrication issues. In *Analysis and Computation* (pp. 248-253). ASCE.

[204] Li, C., Pan, H., Bousseau, A., & Mitra, N. J. (2022). Free2CAD: parsing freehand drawings into CAD commands. *ACM Transactions on Graphics (TOG)*, *41*(4), 1-16.

[205] Vosniakos, G. C. (1992). Knowledge-based interpretation of CAD-drawing annotation for mechanical-engineering components. *Computer-aided design*, *24*(10), 547-555.

[206] Koutamanis, A., & Mitossi, V. (1992, January). Automated recognition of architectural drawings. In *1992 11th IAPR International Conference on Pattern Recognition* (Vol. 1, pp. 660-663). IEEE Computer Society.

[207] Zhi, G. S., Lo, S. M., & Fang, Z. (2003). A graph-based algorithm for extracting units and loops from architectural floor plans for a building evacuation model. *Computer-Aided Design*, *35*(1), 1-14.

[208] Huang, H. C., Lo, S. M., Zhi, G. S., & Yuen, R. K. K. (2008). Graph theory-based approach for automatic recognition of CAD data. *Engineering Applications of Artificial Intelligence*, *21*(7), 1073-1079.

[209] Domínguez, B., García, Á. L., & Feito, F. R. (2012). Semiautomatic detection of floor topology from CAD architectural drawings. *Computer-Aided Design*, *44*(5), 367-378.

[210] Prabhu, B. S., & Pande, S. S. (1999). Intelligent interpretation of CADD drawings. *Computers & Graphics*, *23*(1), 25-44.

[211] Villena Toro, J., & Tarkian, M. (2022, August). Automated and Customized CAD Drawings by Utilizing Machine Learning Algorithms: A Case Study. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (Vol. 86236, p. V03BT03A040). American Society of Mechanical Engineers.

[212] Tutte, W. T., & Tutte, W. T. (2001). *Graph theory* (Vol. 21). Cambridge university press.

[213] West, D. B. (2001). *Introduction to graph theory* (Vol. 2). Upper Saddle River: Prentice hall.

[214] Yao, B., Chen, H., Cheng, C. K., & Graham, R. (2003). Floorplan representations: Complexity and connections. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, *8*(1), 55-80.

[215] Guo, P. N., Cheng, C. K., & Yoshimura, T. (1999, June). An O-tree representation of non-slicing floorplan and its applications. In *Proceedings of the 36th annual ACM/IEEE Design Automation Conference* (pp. 268-273).

[216] Hong, X., Huang, G., Cai, Y., Gu, J., Dong, S., Cheng, C. K., & Gu, J. (2000, November). Corner block list: An effective and efficient topological representation of non-slicing floorplan.

In *IEEE/ACM International Conference on Computer Aided Design. ICCAD-2000. IEEE/ACM Digest of Technical Researchs (Cat. No. 00CH37140)* (pp. 8-12). IEEE.

[217] Knuth, D. E. (1972). The art of computer programming, Vols. l-3.

[218] Otten, R. H. (1982, June). Automatic floorplan design. In *19th Design Automation Conference* (pp. 261-267). IEEE.

[219] Grason, J. (1970). A dual linear graph representation for space-filling location problems of the floor plan type. *Emerging methods in environmental design and planning*.

[220] Schwarz, A., Berry, D. M., & Shaviv, E. (1994). On the use of the automated building design system. *Computer-Aided Design*, *26*(10), 747-762.

[221] Schwarz, A., Berry, D. M., & Shaviv, E. (1994). Representing and solving the automated building design problem. *Computer-aided design*, *26*(9), 689-698.

[222] Levin, P. H. (1964). Use of graphs to decide the optimum layout of buildings. *The Architects' Journal*, *7*, 809-815.

[223] Roth, J., Hashimshony, R., & Wachman, A. (1982). Turning a graph into a rectangular floor plan. *Building and Environment*, *17*(3), 163-173.

[224] Rahbar, M., Mahdavinejad, M., Bemanian, M., Davaie Markazi, A. H., & Hovestadt, L. (2019). Generating synthetic space allocation probability layouts based on trained conditional-GANs. *Applied Artificial Intelligence*, *33*(8), 689-705.

Figure content

Table content