CUBIGRAPH5K

Organizational Graph Generation for Structured Architectural Floor Plan Dataset

YUEHENG LU[1], RUNJIA TIAN[2], AO LI[3], XIAOSHI WANG[4] and GARCIA DEL CASTILLO LOPEZ JOSE LUIS[5]
[1,2,3,4,5]*Harvard Graduate School of Design*
[1,2,3,4,5]*{lu|runjia_tian|aoli|xwang4|jgarciadelcasti}@gsd.harvard.edu*

Abstract. In this paper, a novel synthetic workflow is presented for procedural generation of room relation graphs of floor plans from structured architectural datasets. Different from classical floor plan generation models, which are based on strong heuristics or low-level pixel operations, our method relies on parsing vectorized floor plans to generate their intended organizational graph for further graph-based deep learning. This research work presents the schema for the organizational graphs, describes the generation algorithms, and analyzes its time/space complexity. As a demonstration, a new dataset called CubiGraph5K is presented. This dataset is a collection of graph representations generated by the proposed algorithms, using the floor plans in the popular CubiCasa5K dataset as inputs. The aim of this contribution is to provide a matching dataset that could be used to train neural networks on enhanced floor plan parsing, analysis and generation in future research.

Keywords. Graph Theory; Algorithm; Architecture Design Dataset; Organizational Graph.

1. Introduction

Graph theory plays a significant role in the design and analysis of building layouts. Architects have been used bubble diagrams-a schematic diagram of organizational graph-to represent the spatial relationship of architectural spaces as early as 1938 (Emmons & Paul 2017), and various definitions have been proposed by researchers for such organizational structures (Baglivo & Graver, 1983; Roth & Hashimshony, 1988).

A major part of the creative work for an architect is to translate a high-level organizational graph into a concrete dimensioned architectural floor plan. The automation of this translation process has been studied intensively, but traditional approaches mainly rely on rasterized architectural drawing datasets, i.e. bitmaps of floor plans. This research creates a cornerstone for future studies by developing algorithms to generate a dataset of the graph representations of architectural

PROJECTIONS, Proceedings of the 26th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA) 2021, Volume 1, 81-90.

layouts based on existing drawing documents. The proposed algorithms can be further applied to three-dimensional Building Information Model (BIM) datasets by adding vertical relations to the edge mapping.

Our algorithms are successfully implemented on CubiCasa5K, a large-scale architectural floor plan dataset containing 5000 samples, which are also manually annotated into over 80 architectural element categories (Kalervo et al. 2019). For each floor plan, both raster images and a vectorized model are stored. Annotations are also included as metadata on the vector group objects. As a result, we present CubiGraph5K, a dataset of graph representations of structured floor plans, published open-source as part of this work.

In this paper, we describe the structure of the proposed organizational graphs, present the graph-generation algorithms, explain its application to the proposed floor plan dataset, and discuss the results obtained in the process.

2. Previous Work

The CubiCasa5K dataset was first used by its authors to analyze floor plans of residential buildings: the dataset was first explained, and a multi-task model was then introduced and applied on the dataset to predict the labels of architectural elements for a given floor plan image. The authors conclude that the performance of any machine learning model is dependent on the design of the dataset it is trained on, suggesting that future floor plan generation related machine learning research could benefit from the contributions of their work.

The traditional methods used to automate floor plan generation include rule-based (Levin, 1964) or optimizations algorithms (Roth & Hashimshony, 1988). More recently, organizational graphs have been used to train neural networks (Scarselli et al., 2009). CNNs and convolutional autoencoders have also been used to predict the location of rooms and walls inside of a floor plan boundary (Wu et al. 2019). Graph Convolutional Networks (GCN) have also shown potential in handling 2D images whose components are related to each other as a graph. Research work has been performed by using GCN along with other types of networks to generate images from a given scene graph which illustrates the spatial relation among different objects in the scene (Johnson et al., 2018).

A more recent example of graph-related research is the floor plan generating GAN model "House-GAN" (Nauata et al., 2020). This work shows that bubble diagrams, as a graph embodying organizational information, can be used as inputs of GAN models, facilitating the generation of floor plans. Similar work has been proposed in this direction (Ruizhen et al., 2020). One shortcoming of House-GAN is that the organizational graphs for training had to be manually generated by human annotators, which is a rather labor-intensive operation difficult to scale for larger datasets. In this research, we focus on the automatic generation of such graphs.

3. Method

3.1. GRAPH EXTRACTING ALGORITHM

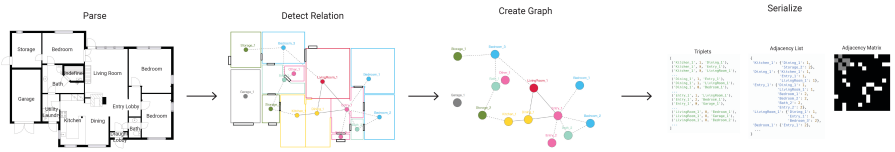

Figure 1. Synthetic Workflow.

Our goal is to extract geometric information from structured floor plan data, and generate a graph representation that describes the room relations in such a floor plan. The graph-generating algorithm consists of two steps: 1) machine-parsing the structured floor plan data, to retrieve geometric information of related elements (i.e. rooms, doors); 2) discriminating the relationships between rooms pairwisely to construct the graph. We first present the format of the input data followed by the relationship definition between two rooms, then explain the algorithm in detail followed by the possible output formats, and finally discuss the resulting algorithm complexity.

3.1.1. Input: Structured Floor Plan Data

The input of the graph-generating algorithm is structured floor plan data represented in vector format, like for example Scalable Vector Graphics (SVG), Drawing Exchange Format (DXF), etc. As a minimal requirement, each floor plan should include tags for different types of architectural elements and each element should include precise euclidean coordinates of their inner boundary.

3.1.2. Room Relations

In architectural practice, designers use bubble diagrams to study the relationship of rooms. Many kinds of relationships can be illustrated using these diagrams, such as program proximity, spatial connectivity, flow of humans, etc. In this paper, since our study focuses on residential floor plans, we present three possible relations between each room pair to describe connectivity: adjacent, door-connect and not-connect.

Adjacent. When no physical partition between two rooms is present, these two rooms are considered adjacent. Geometrically, the two polygons representing adjacent rooms should touch each other by more than a single point.

Door-connect. When two rooms can be and can only be traversed through a door, they are considered as door-connect. Geometrically, the two polygons representing the two door-connect rooms should touch the same door polygon by more than a single point. If two rooms are both adjacent and door-connect, they are considered adjacent, since direct connectivity is prioritized here. Door-connect relation does not have transitivity.

Not-connect. Rooms that are not adjacent or door-connect are considered

not-connect. In another word, people cannot go from one room to the other without entering a third room or a space that is not visible in the floor plan.

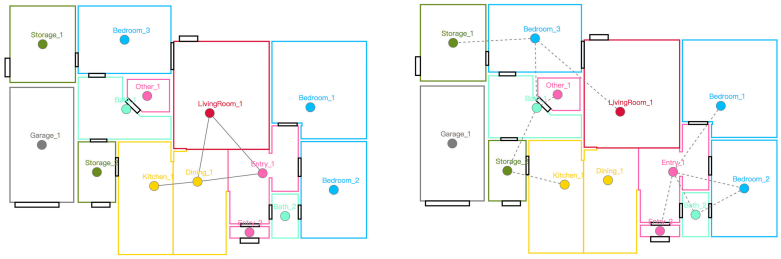

Figure 2. Adjacent Relationship and Door-connect Relationship.

3.1.3. Algorithm for Single Floor Plan

Our proposed graph-generation algorithm features two steps: 1) parsing the floor plan data to obtain the geometry of rooms and doors, and 2) calculating the geometric relationships for each room pair to generate the graph.

Parsing. As discussed previously, this algorithm relies on the floor plan data to contain annotated and classified room and door boundaries. Our implementation has been tested with input SVG files, an extended form of Extensible Markup Language (XML), where each room and the door between them is explicitly labelled and includes Euclidean coordinates of the curves of their inner boundaries.

Algorithm 1: Room Relation Calculation

Given P_{room}, P_{door}
$A_{door} \leftarrow$ empty list to store adjacent doors of each room.
while $r, d \leftarrow$ *each pair of room and door* **do**
  if r *intersect with* d **then**
    Record d as r's adjacent door in A_{door}.
  end
end
$R \leftarrow$ empty list to store pairwise relationship of rooms as adjacency list.
while $r_1, r_2 \leftarrow$ *each pair of rooms* **do**
  if r_1 *intersect with* r_2 **then**
    Record r_1 and r_2 in each other's adjacent rooms with relationship **ADJACENT**.
  else
    if r_1 *and* r_2 *share same adjacent door according to* A_{door} **then**
      Record r_1 and r_2 in each other's adjacent rooms with relationship **DOOR_CONNECT**.
    end
  end
end
Return R.

Algorithm 2: Dataset Iteration

input : A sequence of floor plans D such that for all $P \in D$, P represents a XML file that represents one floor plan data structure
output: A sequence of graph data structure of the same size as D
initialize G arbitarily;
for $i \leftarrow 0$ **to** $|D|$ **do**
  $P \leftarrow D[i]$;
  $(P_{ROOM}, P_{DOOR}) \leftarrow PARSE(P)$;
  $G_{j,k} \leftarrow ROOMRELATION(P_{ROOM}, P_{DOOR})$;
end
return G;

Figure 3. Pseudo Code for Room Relation Calculation and Dataset Iteration Algorithm.

Calculating Room Relation. We iterate the dataset and calculate pairwise room-room and room-door relationships in each floor plan to generate the graph. For disambiguation, we prioritize room-room relationship over room-door relationship. After the two rounds of iteration, the relation between all pairs of rooms in the floor plan is recorded to construct the graph representation. Pseudo code for room relation calculation is detailed in Figure 3.

We implement a robust algorithm for relationship calculation by dilating the polygons by a small distance D and calculating the intersection of the two targeting

polygons, and set a threshold T for detecting adjacency. Some examples of inaccurate data labeling that can be accommodated by our algorithm will be discussed in Section 3.3.3.

3.1.4. Output: Data Structure for Room Relation Graph

We propose three types of representations for the output graph: triplets, adjacent list and adjacency matrix (Figure 4). The time complexity and space complexity of all methods are similar, but they can be applied to various use cases, and the advantage of each representation will depend on the network structure that is applied for performing geometric deep learning on the relational graph dataset.

Figure 4. Plan 41: Graph, Triplet, Adjacency List and Adjacency Matrix as Rasterized Image.

Triplet. This is a common representation used in Message Passing Networks (MPN) (Gilmer, J, et al., 2017). It is a generic representation for all types of weighted graphs. It uses a triplet (i, w, j) to represent an edge with weight w between node i and j.

Adjacency List. This representation features a structured room-adjacencies set of relations, where each room is associated with an object composed of all the room labels that it is connected to, and their connectivity type.

Adjacency Matrix. The output of this representation features a pixelated square image, where the value of each pixel $w_{ij} = w_{ji} = w(i, j)$ represents the category of connection between two connected rooms. This representation is mainly used for graphs that are very dense.

3.1.5. Workflow for a Dataset

In this work, we propose an algorithm to generate the floor plan graph representation for a given architectural dataset in pseudo polynomial time, defining the following mathematical formulation of an architectural floor plan dataset:

Room: We represent R_j as a polygon composed of a array of two-dimensional points: $R_j = \{p_k | k = 0, \dots c_j\}$, where c_j is the total number of points in the geometry layout of room .

Floor Plan $P_i = \{R_j | j = 0, 1, \dots, n_i\}$ where ni represents the number of Rooms in each floor plan and $\max(n_0, \dots n_i) = n$. Therefore Floorplan P_i is composed of at most n Rooms.

Annotated Floor Plan Dataset $D = \{P = D_i | f \quad \text{or} \quad i = 0, 1, \dots, m\}$, where P is an element in the Dataset and m is the total number of Floorplans in the Dataset.

Graph Dataset $G = \{G_i | f \quad \text{or} \quad i = 0, 1, \ldots, m\}$ where G_i represents the Graph in Floor Plan D_i.

Our dataset-iteration algorithm is detailed in Fig 3. The parsing function for each floor plan is trivial. The ROOMRELATION function calls the room relation calculation algorithm defined in Section 3.1.3.

3.1.6. Algorithm Complexity Analysis

We formulate the graph parsing into the following mathematical problem: from Section 3.1.5, we assume there are m rooms in the data set in total, and maximum n rooms and d doors in the floor plan, each represented by the a set of two-dimensional points $R_i = \{p_j\}$, and the point set has maximum cardinal c, where $|R_i| \leq c$, for $i = 0, 1, \ldots, n$. Then the time complexity for computing the relational graph for each plan will be $O\left(m\left(nd + \frac{n^2}{2}\right)c^2\right)$. It is noticeable that in most architecture design dataset, n will not be arbitrarily large, as the smallest using units in architecture that counts as a room should not be arbitrarily small. Therefore in most cases, it does not contribute to the overall time complexity, thus the time complexity would be reduced to $O(mdc^2)$. Our algorithm runs in a pseudo-polynomial time of total number of floor plans. Note that normally the number of doors and rooms are on the same scale, since rooms in a floor plan usually form a tree-like structure. As a result the complexity can be further simplified as an expression of m and c: $O(mc^2)$.

3.2. CASE STUDY

In order to prove the validity of the algorithm, we applied it to all floor plans in the CubiCasa5K dataset (Kalervo et al. 2019), a collection of annotated architectural floor plans in both raster and vector format. As a result, we present CubiGraph5K, a dataset of graph representations of architectural floor plans that mirrors the structure of CubiCasa5K, and which could be used as stand-alone or in tandem with the original for machine learning applications.

3.2.1. Single Floor Plan

To generate the CubiGraph5K dataset, an object model was implemented representing the key classes for elements available in the source dataset: Plan, Room and Door. These classes represent concretions of the high-level graph, node and edge elements they represent in the graph structure, and feature properties and methods that allow them to be queried as part of the proposed algorithms.

For simplicity, all Room types found in the original dataset were mapped into a set of basic types: LivingRoom, Bedroom, Kitchen, Dining, Bath, Storage, Entry, Garage, Outdoor and Other. If multiple instantiations of the same Room type are present in a Plan, correlative 1-based indices are assigned as ID.

We use Plan 41 in the dataset as an example to show case the creation of single floor plan.

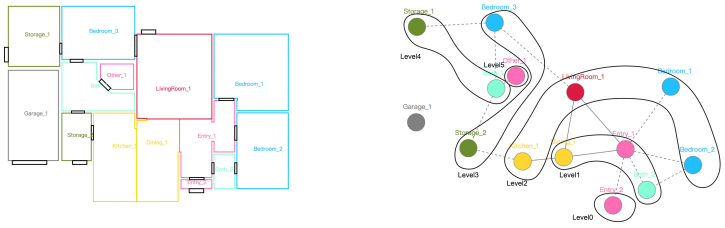

Figure 5. Plan 41 and Diameter of Plan 41.

Rooms. Calling plan.room_names would return a list of all room labels, with unique correlative IDs concatenated.

Number of Instances for Each Room Type. The attribute plan.room_type_count will return a dictionary with key, value pairs representing room types and their corresponding number of occurrences in the plan.

Diameter of a Floor Plan. As the space gets larger, one may be interested in the complexity of a floor plan. We define the 'diameter' of a floor plan as the number of edges passed between two farthest rooms. Figure 5 illustrates the definition of plan diameter using Plan 41 as an example (in this case, $diameter = 5$). The metric of diameter finds its meaning in design as well, it can reflect the spatial progression in the floor plan, indicating the layering of the space, which can, to some extent, impact the design of circulation, acoustics or even energy flow.

Shortest Path. Since the floor plan is represented as a graph, graph search algorithms such as Breadth First Search can be easily applied to get the shortest path(s) between two rooms. One thing worth mentioning is that both adjacent and door-connect are treated the same as opposed to not-connect in the calculation. With our tool, user can get all possible shortest path(s) between two rooms.

3.2.2. Multiple Floor Plans

The single plan methods and attributes provide flexible subroutines to iterate over the multiple plans and the entire dataset, so that users can explore its statistics and filter plans with specific characters. The diameters of floor plans in CubiGraph5K concentrate around 3, 4, 5 and the most common room types are Studio, 1B1B, 2B1B and 3B2B (See Figure 6). Occasionally, rooms in the floor plan are not connected to any other rooms, which makes up 2.1% of the total rooms. As one may expect, 54.6% of the "islands" are Garage and Storage (outdoor).

To maximize the convenience for the users, we also provide utility functions on the whole dataset, which are also documented in the repository. The open-source nature of the project also gives users the flexibility to extend the existing functions library. Using these attributes and methods as subroutines, users can further develop their own functions or classes to achieve specific goals with the dataset.

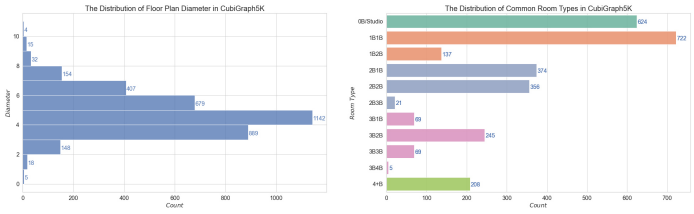

Figure 6. Dataset Statistics: Diameter Distribution (left), Room Type Distribution (right).

3.3. EVALUATION

To estimate the validity of the algorithm and its implementation, an initial exploratory analysis of the CubiGraph5K dataset was conducted. First, the contributed graph dataset was compared to the original CubiCasa5K dataset in terms of number of rooms per each room type. Second, the resulting graph representation of floor plans was examined by human experts in order to verify its correctness. To better facilitate this process, a visualization tool was developed.

3.3.1. Dataset Verification

In Table 1, The statistics of each room type in both datasets show that the proposed algorithm captures most of the major room types in the CubiCasa5K dataset. The statistics of the original dataset are estimated from Fig.3. in the original paper. The minor differences between CubiCasa5K and results generated by our parsing algorithm appear to be caused by different definitions of room types when grouping all original 62 types into the 10 major types in our CubiGraph5K implementation. Since the graph generation algorithm only works for single-story floor plans, the total number of rooms in CubiGraph5K shrinks by 42.8%.

Table 1. Statistics Comparison between CubiCasa5K and CubiGraph5K.

Room Type	# of Room in CubiCasa5k	# of Room Parsed	# of Room in CubiGraph5K
LivingRoom	11347	4611	3001
Bedroom	7353	8274	4853
Kitchen	4109	4525	3125
Dining	941	949	502
Bath	6475	7241	4252
Storage	6214	6347	3303
Entry	5393	5843	3524
Garage	729	687	326
Outdoor	6976	7805	4272
Other	12750	14732	7731
All	62287	61014	34889

3.3.2. Graph Visualization and Verification

In order to easily visualize and verify the generated graph results, a tool for visually overlaying the nodes and weights of a graph on its associated floor plan was developed. To manually verify the correctness of a room relation graph, the output as adjacency list and the visualized graph will be generated and laid out side-by-side with the original floor plan.

A random sample of 50 floor plans were evaluated by the authors, following the above procedure. 68.0% of the graph results correctly reflected the original floor

plans. It is worth mention that 93.8% of the incorrect cases were caused by data labelling mistake from CubiCasa5K, which proved the correctness and robustness of the proposed algorithms.

3.3.3. Improvements

The original algorithm relied heavily on the correct overlapping of polygon boundaries and the low-tolerance matching of their vertices. As discussed in Section 1.3, many architectural plans are not drawn so precisely, including many of the CubiCasa5K dataset, resulting in incorrect graph topographies. In our improved room relation calculation algorithm, geometric operations (i.e. dilation and intersection calculation) are performed to decide if two polygons are adjacent. Instead of simply deciding if the boundaries of the two polygons share more than a single point, two thresholds are set for the minimum intersection between two rooms and between a room and a door. We demonstrate the necessity of such operations by comparing the graph results generated by two earlier versions of our algorithm.

No Threshold between Two Rooms. When room polygons are not accurately drawn, meaning that they don't share boundaries when they should (see Figure 7), adjacent rooms would not be detected by the algorithm.

No Threshold between Door and Room. When a door polygon fails to touch its intended adjacent room or accidentally touches a room other than its intended neighbor, the algorithm is not able to decide the correct door-connect relation. (See Figure 7).

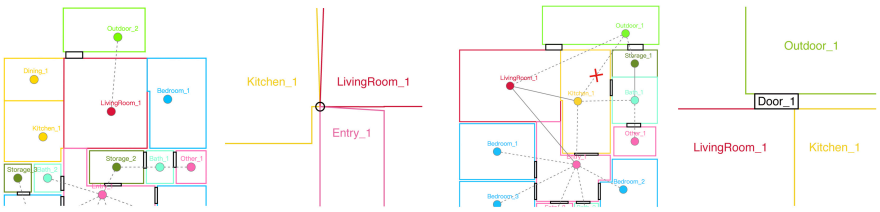

Figure 7. Visualized Graph and Diagram: Plan 353 (left), Plan 283 (right).

Other limitations of CubiCasa5K dataset, such as incorrectly labelled rooms or elements with invalid polygons, cannot be handled at this time by our proposed algorithm. A complete list of floor plans with invalid features is available in the repository's documentation. We leave the handling of those edge conditions to future work.

4. Contribution

CubiGraph5K, the matching graph dataset of CubiCasa5K, is created by our proposed algorithms. It is published along with our implementation of the algorithms and additional graph query functions library at https://github.com/luyueheng/CubiGraph5K".

5. Discussion and Conclusion

In this paper, we proposed a novel algorithm to systematically parse structured floor plans, generate graph representations and serialize the results in portable format. Our approach was tested on CubiCasa5K, generating the corresponding CubiGraph5K and providing accessibility to future researchers in the realm of architecture design computation.

There are mainly two limitations of the proposed workflow. First, it highly relies on the consistency of the vectorized floor plans thus it was only tested on CubiCasa5K with properly labelled SVG files. Second, our definition of room relations didn't take cross-level connections into account thus it only works for single-level floor plans. Despite the limitations, our approach has the potentials to be integrated in graph-based floor plan generation network, i.e. Graph Convolution Network. By defining more targeting room relations, the generated graph could depict more complex floor plan layouts. Especially, similar workflows could be applied on structured three-dimensional BIM datasets.

We hope the proposed approach and dataset contributed with this research will be beneficial for further studies dealing with topological representations of architectural drawings.

References

Baglivo, J.A. and Graver, J.E.: 1983, *Incidence and Symmetry in Design and Architecture*, Cambridge University Press Cambridge.

Emmons, P.: 2017, Embodying Networks: Bubble Diagrams and the Image of Modern Organicism, *The Journal of Architecture*, **22**(5), 854–874.

Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O. and Dahl, G.E.: 2017, Neural Message Passing for Quantum Chemistry, *Proceedings of the 34th International Conference on Machine Learning*, 1263–1272.

Hu, R., Huang, Z., Tang, Y., van Kaick, O., Zhang, H. and Huang, H.: 2020, Graph2Plan: Learning Floorplan Generation from Layout Graphs, *ACM Transactions on Graphics*, **39**(4).

Isola, P., Zhu, J.Y., Zhou, T. and Efros, A.A.: 2017, Image-to-Image Translation with Conditional Adversarial Networks, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1125-1134.

Johnson, J., Gupta, A. and Fei-Fei, L.: 2018, Image Generation from Scene Graphs, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1219-1228.

Kalervo, A., Ylioinas, J., Häikiö, M., Karhu, A. and Kannala, J.: 2019, CubiCasa5k: A Dataset and an Improved Multi-task Model for Floorplan Image Analysis, *Scandinavian Conference on Image Analysis*, 28-40.

Levin, P.H.: 1964, *Use of Graphs to Decide the Optimum Layout of Buildings*, Building Research Station, Garston.

Nauata, N., Chang, K.H., Cheng, C.Y., Mori, G. and Furukawa, Y.: 2020, House-GAN: Relational Generative Adversarial Networks for Graph-constrained House Layout Generation, *Computer Vision – ECCV 2020 Lecture Notes in Computer Science*, 162–177.

Roth, J. and Hashimshony, R.: 1988, Algorithms in Graph Theory and Their Use for Solving Problems in architectural design, *Computer-Aided Design*, **20**(7), 373-381.

Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M. and Monfardini, G.: 2008, The Graph Neural Network Model, *IEEE Transactions on Neural Networks*, **20**(1), 61-80.