Floor Plan Reconstruction from Sparse Views: Combining Graph Neural Network with Constrained Diffusion

Arnaud Gueze[1,2], Matthieu Ospici[2], Damien Rohmer[1], and Marie-Paule Cani[1]

[1]LIX, École Polytechnique, Institut Polytechnique de Paris
[2]Homiwoo
{*arnaud.gueze,damien.rohmer,marie-paule.cani*}*@polytechnique.edu*
matthieuo@homiwoo.com

Abstract

We address the challenging problem of floor plan reconstruction from sparse views and a room-connectivity graph. As a first stage, we construct a flexible graph-structure unifying the connectivity graph and the sparse observed data. Using our Graph Neural Network architecture, we can then refine the available information and predict unobserved room properties. In a second step, we introduce a Constrained Diffusion Model to reconstruct consistent floor plan matching the available information, despite of its sparsity. More precisely, we use a Cross-Attention mechanism armed with shape descriptors to guarantee that the generated floor plan reflects both the input room connectivity and the geometry observed in the sparse views.

1. Introduction

Floor plans serve as a common representation to convey accurate information about the layout of houses and apartments, facilitating effective communication between architects, designers, or parties to a real estate transaction. When considering real-case scenarios encountered in real estate applications, accurate floor plans are rarely available. Instead, apartment descriptions are often provided in the form of a small number of photos taken during visits, providing only partial information, and supplemented by textual descriptions of room types. The ability to reconstruct a plausible floor plan from such incomplete and sparse data would be of great benefit in real estate in the objective of converging toward the real plan when combined with additional data.

In recent years, there has been significant advancement in the automatic generation of floor plans. However, existing methods typically fall into two categories. The first approach involves a reconstruction method, necessitating a dense dataset containing all the required information about the layout [30]. Conversely, the second approach adopts generative models, capable of synthesizing coherent floor plans but presenting arbitrary layouts without respecting the specific constraints of the input information [29]

Our goal is more ambitious. In this work, we tackle the generation of a coherent floor plan while integrating all the existing clues, however scattered and incomplete. These clues can take the form of a partial layout of individual rooms or information about the contiguity of rooms. This case poses significant challenges as it requires effectively reconciling local constraints with global consistency. Given that only partial and local information is available, generating individual rooms separately is unfeasible. Instead, the floor plan must be synthesized as a whole while preserving these local constraints.

In our study, we examine a specific setting of the problem, where partial shape information is extracted from non-panoramic photographic views, to form the basis of our floor plan synthesis. To make this problem solvable, we also make the following assumptions: First, topological connections between rooms, represented by a complete connection graph, are assumed to be available. Furthermore, we assume that the mapping between each view and the corresponding room is known. Each room can be associated to zero, one or more partial views. Lastly, we assume that the partial layout information extracted from each view of the room is always correctly and accurately extracted, without suffering from projection or occlusion issues.

Our output is a vectorized floor plan describing the walls of the rooms as polygons, and including the connections formed by doors between these rooms. This output plan ensures that the connection aligns with the information provided in the input graph, and that the room geometries adhere to the partial constraints extracted from the available views.

To achieve these results, we introduce three technical contributions. Firstly, we propose a new, flexible Graph-based representation, specifically designed to handle sparse constraints, accommodating a variable number of views for each room. Secondly, we leverage Graph Neural Networks (GNNs) to merge the sparse geometrical representations of individual rooms, creating a shared encoding that facilitates the inference of the global geometry of the entire floor plan. Finally, we propose an extension of the HouseDiffusion [29] framework that integrates cross-attention modules, allowing geometric constraints to be taken into account in the vector floor plan synthesis process.

2. Related works

2.1. Floor plan reconstruction

Initially, floor plan reconstruction aimed at reconstructing maps from acquired images or scans, often using panoramic images [30, 32, 41] or 3D point clouds [39]. These approaches can reconstruct complex maps but require complete and dense input data covering the entire visible area to be reconstructed. Interestingly, Huang et al. [15] investigated the use of sparse and incomplete views for plan reconstruction, but the approach was limited to synthesizing a single room. In recent years, generative AI techniques have proposed floor plan generation based on very sparse constraints, incorporating a graph of topological constraints between the rooms and potentially defining the bounds of exterior walls [14, 33]. GAN-based synthesis models were also developed to generate rasterized floor plans with room connectivity graph constraints [24, 25], and were extended using diffusion models to synthesis vector outputs [29]. However, these approaches did not consider geometrical constraints specific to individual rooms. In contrast, our work extends the diffusion model proposed in HouseDiffusion [29] in injecting additional geometrical constraints associated with each room using a Cross Attention [35] method. To make our approach robust to sparse and incomplete data, we rely on the combined use of a robust prediction relying on a graph-based approach.

2.2. Graph neural networks

Graph Neural Networks (GNN) have recently gained popularity for learning and inferring information structured as graph with arbitrary topology. GNNs first featured the message passing framework (MPNN) [11, 18, 36] using neighborhood feature aggregation to better represent a node's features. Although MPNNs perform well in local data inference, they exhibit limitations with depth and long range interactions, such as over-squashing [34] and over-smoothing [26]. Graph Transformers [23, 38] have been introduced to overcome these limitations. These models enable nodes to access information from all other nodes in the graph through Global Self Attention. Using GraphTransformers requires enriching the node feature vectors with positional and structural information about the graph, commonly known as Positional Encodings and Structural Encodings [28]. Positional Encodings are typically derived from the Graph Laplacian [6, 19], while Structural Encodings are often based on a Random Walk Matrix [7]. Combining MPNN and Graph Transformer offers an efficient compromise, leveraging their strengths at both local and global scales. For instance, GraphTrans [37] uses an MPNN encoder to enrich the node features before applying a Graph Transformer on top. The so-called GPS (General, Powerful, Scalable Graph Transformer) framework [28], associates the two combining their resulting representations of nodes at every layer. In this work, we rely on the recent GPS++ [22] introduced in the context of molecular property prediction that we adapt to the case of room connectivity. To the best of our knowledge, a GNN has not been used to leverage geometrical information in combination with a generative model for floor plan synthesis.

3. Methodology

3.1. Overview

Our method relies on the main steps illustrated in Fig. 1. We consider as inputs a graph G indicating the connectivity between the rooms i, and a set of associated partial views represented as partial polygons. As explained in Sec. 3.2, these two inputs are combined into an extended graph G' incorporating additional nodes representing the geometrical features of the partial views using Zernike moments. We then employ a GNN following a GPS++ architecture to predict the room features as well as the number of corners c_i of each of them. This prediction of the number of corners is efficiently accomplished through the use of an ordinal regression technique, as outlined in Sec. 3.3. The final step uses a diffusion model for synthesizing the floor plan. This diffusion model inspired from HouseDiffusion [29], and takes as input the connectivity graph, the room features learned by the GNN, as well as the complete number of predicted corners c_i. As described in Sec. 3.4, we extend this diffusion model to account for the geometry of the partial shapes, constraining the alignment of the visible corners between neighboring rooms. This extension is obtained by encoding the polygons as a chain code sequence, and adding a cross-attention module using this information in the decoder layer of the diffusion architecture.

3.2. Graph construction

Let $G = (V, E)$, called room connectivity graph, be an undirected graph with V being a set of N nodes, where N

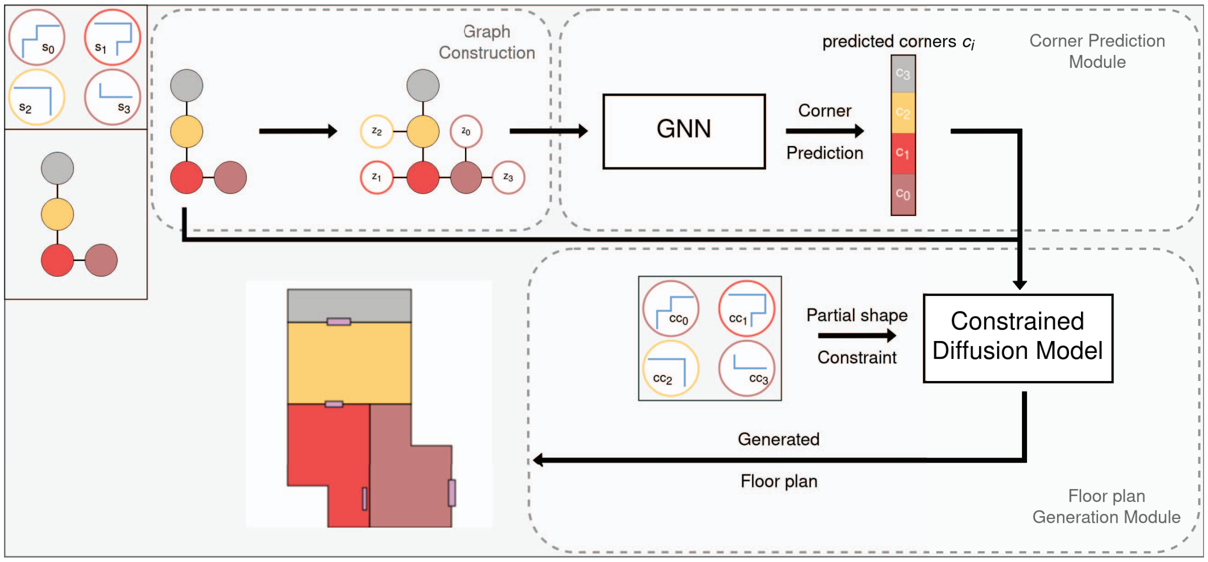


Figure 1: We consider for input the connectivity graph and a set of partial shapes associated to different rooms. The graph construction method attaches the Zernike Moments $z_i = Zernike(s_i)$ to the corresponding room, for every partial shape s_i. We then computes the number of corners per room c_j thanks to our Graph Encoder. These information are then utilized as well as the Chain Code $cc_i = ChainCode(s_i)$ to generate the corresponding floor plan. Each cc_i is used to condition the Constrained Diffusion Model that outputs the floor plan

corresponds to the number of rooms. The edge $e_i \in E$ represents whether two room nodes are accessible from one another through a door. We call $X = \{x_0, x_1, ..., x_N\}$ the graph features vector, where $x_i \in \mathbb{R}^d$ corresponds to the per-room feature element representation. In our approach, the feature elements, such as room types, can be arbitrary data. Notably, these features x_i, are not directly provided as input but are learned and inferred by the subsequent GNN using the input graph connectivity and partial geometry. To initiate the process, the values of x_i are initialized with random noise at the outset of our pipeline.

For each image representing a partial view of a room v_i, we consider the polygonal geometry of the walls from a top-down view, extracted by the approach from Zhang et al. [40]. These top-down views $s_{j\in\mathcal{I}(v_i)} \in \mathcal{S}$ will be referred to as *partial shapes* throughout the paper with $\mathcal{S}$ the set of partial shape and $\mathcal{I}(v_i)$ the set of views of the room v_i. Note that each room can be associated to 0, one or more partial shapes.

Since the input images lack calibration relative to a global reference frame, the coordinates of the partial shapes are expressed in an arbitrary frame. To mitigate the impact of this reference frame, we adopt a method to encode the polygon's geometry to various levels of details using descriptors invariant to rotation, translation, and scale. For this purpose, we propose the use of Zernike Moments [16] $z_j = Zernike(s_j) \in \mathbb{R}^d$, which corresponds to the projection of the filled top-view polygon's image over the first Zernike polynomials defining an orthogonal basis on a disk.

Given these partial shapes values, we construct the extended Graph G' as a superset of G with additional geometric information

$$\begin{aligned} G' &= (V', E') \\ V' &= V \cup \mathcal{S} \\ E' &= E \cup \{(v_i, s_j), s_j \in \mathcal{I}(v_i), v_i \in V\} \end{aligned} \tag{1}$$

G' connects, for each room, each partial shape $s_{j\in\mathcal{I}(v_i)}$ to the correct room node v_i. Each of these partial shape nodes have as node features their respective Zernike Moments.

Adding such additional nodes enables us to decouple room nodes from the information provided about them so that predictions can be made whether data is present or not. This fits the sparse nature of our input data as well as contextualizing information. A floor plan is a compact structure, often viewed as a jigsaw puzzle [12], so enabling the structure to communicate between every room shape is key to predict a coherent floor plan. Thus, a node with no information attached to it, can still get an accurate representation based on the representations of its neighbors. While on the other hand, a node with a lot of observed information will benefit from a more precise representation and provide shape information to its neighborhood.

3.3. GNN-based Corner Prediction

The second step of our pipeline corresponds to the use of a trained GNN on the extended graph G' to fill the generic feature vectors X and predict the number of corners of each room $c_i \in \mathbb{N}$ with the help of the information of the Zernike Moments. To this end, we use the architecture proposed in GPS++ [22] and slightly adapt the MPNN module. We replace the *sum* aggregation function, by the concatenation of both *sum* and $\max$ aggregations for the node and graph feature aggregations:

$$Agg_{j \in V'}(x_j) = [\sum_{j \in V'} x_j \mid \max_{j \in V'}(x_j)] \tag{2}$$

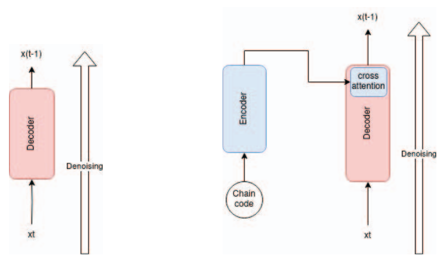
We find that such aggregation scheme empirically improves performance. We further used a Biased Attention Module with the Shortest-Path-Distance bias [38], and relied on the standard positional and structure node features with, respectively, Laplacian Eigenvectors Positional Encoding, and Random-Walk Structural Encodings, as well as node degree information.

Our Graph Encoder computes representations for each node in the graph. Mainly it will give us some geometry-aware embeddings for the different nodes in the graph. Also, our method is suited for our sparse setting as the number of predictions in the output depends only on the connectivity graph G and not the observed shapes $s_{j \in \mathcal{I}(v_i)}$. We can then use these room embeddings to predict the number of corners c_i.

Although the number of corners could be predicted as any generic feature element, our study revealed that employing an Ordinal Regression technique leads to more robust prediction. We first consider a so called *Manhattan assumption* in our dataset, where only 90° angles exist between walls, ensuring that all rooms have an even number of corners. Given this information, we directly predict the number of corners in the set of discrete even numbers $\{2n \geq 4, n \in \mathbb{N}\}$. To this end, we employ the ordinal regression technique proposed by Shi et al. [31] that enables making predictions in an orderly manner.

Since our predicted classes follow an ascending order – For instance, the class of a 6 corners room is necessarily after the class of the 4 corners room, and before the one with 8 corners –, the ordinal regression model can take advantage of dependencies between the classes to provide more accurate predictions compared to a generic classification where the classes are treated independently.

The method is also more robust than standard regression approaches as it predicts discrete values from the specific distributions of valid corner values, allowing stable results even for challenging rooms with complex geometries and a high number of corners.



(a) Housediffusion architecture (b) Our proposed architecture

Figure 2: In both architectures, the diffusion denoising (from x_t to x_{t-1}) is performed by the decoder, but unlike HouseDiffusion [29], we add a cross-attention module and an encoder to take into account constraints on partial shapes, which are encoded as chaincodes.

3.4. Constrained Diffusion Floor plan generation

3.4.1 Diffusion Model architecture

The last step corresponds to the use of a diffusion model that synthesizes the rooms as a vectorized floor plan represented by $P = \{P_1, P_2, ..., P_N\}$ which is a set of polygon for each room or door to be generated by the diffusion process embedded in a global reference frame. Our method relies on an extension of the existing HouseDiffusion model [29]. The original model can be seen as a decoder structure, and makes use of a Transformer architecture to apply the diffusion process to the different corners' coordinates, while suitable properties such as orthogonality, and parallelism, are obtained using a mixture of continuous and discrete denoising [2]. However, this existing approach cannot take into account clues about the expected geometry of the rooms, and is thus generating arbitrary room shapes.

To constrain the diffusion with additional information about the room geometry, we propose to extend the approach into an encoder/decoder architecture as illustrated in Fig. 2. While the decoder follows the same structure as the original HouseDiffusion, we add an encoder taking as input a descriptor of the geometry extracted from the partial views, and plug the encoded geometric features to an additional cross attention module added into the decoder. This final cross-attention is used to inject the partial shape information into the denoising process. Throughout the process, the geometrical features are used as soft constraints, allowing robust handling of unseen rooms where no geometrical clue is provided, as well as preserving compact and coherent assembly even when the network needs to fill-in missing information.

The details of the modified architecture are as follows:

The encoder has four multi-heads attention layers. It consists of a standard self attention layer with residual connections followed by a normalization and a feed-forward layers. The decoder uses the three multi-head attention modules (four heads) described in the original HouseDiffusion framework [28]. The modifications we have made to this architecture are as follows:

The decoder uses the three multi-head attention modules (four heads) described in the original HouseDiffusion framework [29]. The additions we have made are as follows: the outputs of the three attention modules are summed, followed a residual connection and a normalization. Next, the output is fed into the cross-attention module, followed by a final residual connection and normalization plus a linear layer. The key and value of this cross-attention layer is then the encoder output.

The constrained local alignment between the room shape and the visible corners is ensured via the use of a cross-attention mask [35] associated to the cross-attention module. This block diagonal mask allows to focus the attention of the geometrical representation relying on chain code encoding, as detailed in the next part, with the corresponding polygon corner.

3.4.2 Partial shape encoding

To encode the partial shape of each room for the cross-attention module, we use the Chain Code encoding [9] that explicitly encodes the corners of the polygon as a sequence compatible with a transformer. We further use the *first-difference transform* over the chain code, ensuring that the encoding is rotation-invariant.

We considered the chain code cc_i of the room i as a sequence of vectors cc_{ij} described with eight dimensions (7 one-hot encoding, and one length), as follows:

$$cc_{ij} = [\underbrace{\text{nature}}_{\text{3 dims}}, \underbrace{\text{direction}}_{\text{4 dims}}, \underbrace{L}_{\in[-1,1]}] \quad (3)$$

Nature is a 3-dimensions one-hot encoding the nature of the element. Following [5, 35], we employ a specific encoding to delimit the beginning and end of a chain code: $[1, 0, 0]$ indicates the start of the chain code, $[0, 0, 1]$ its end, and $[0, 1, 0]$ represents an element of the chain code. *Direction* is a 4-dimensions one-hot encoding describing the four possible local change of direction relative to the current one. L is the length of the polygon edge, normalized between $[-1, 1]$. The normalization is performed per house. For each room i, we represent the entire chain code cc_i as a matrix of size $l_i \times 8$ where l_i is the number of vectors used to represent the polygon associated with the partial view. The global chain code representing the entire house is obtained in concatenating all the matrices associated to cc_i.

Finally, the chain codes are processed using a linear layer to align their dimension with that of the transformer (512). The final size of the matrix of each chain code is therefore $l_i \times 512$. As the chain codes are ordered, we add a standard positional sine and cosine encoding [21] to the chain codes.

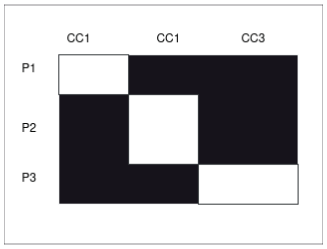


Figure 3: Example of a cross-attention mask for three rooms. Each white areas indicates where attention is focused. With this mask, a chain code CC_i focuses only on P_i. The length of the chain codes and the number of corners in each room may not be equal, so the cross-attention mask is not a square matrix.

These chain codes are finally used as input to the cross-attention module to constrain the polygon corners P_i generated in the diffusion process by the chain code cc_i. The cross attention is guided by a cross-attention mask to ensure that cc_i constrains only P_i. As illustrated in Fig. 3, this mask can be represented as a block diagonal matrix of size $N \times M$, with N the number of corners in the entire scene, and M the size of all the concatenated chains code.

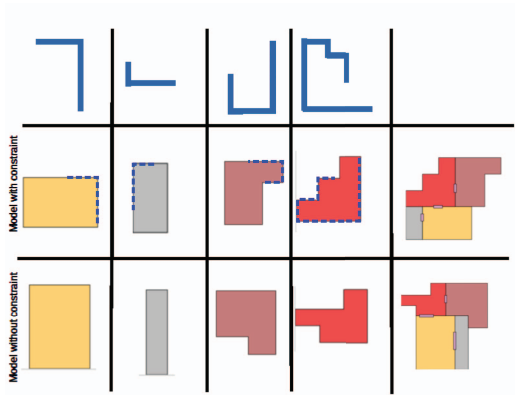


Figure 4: Effect of constraints on the diffusion process. The first line contains the partial shape constraints (rotation and scaling are arbitrary). The second and third line represent the generated rooms with and without cross-attention respectively. The last column represents the final floorplan generated by the diffusion model. The blue dotted lines are manually added to illustrate the possible matching corners.

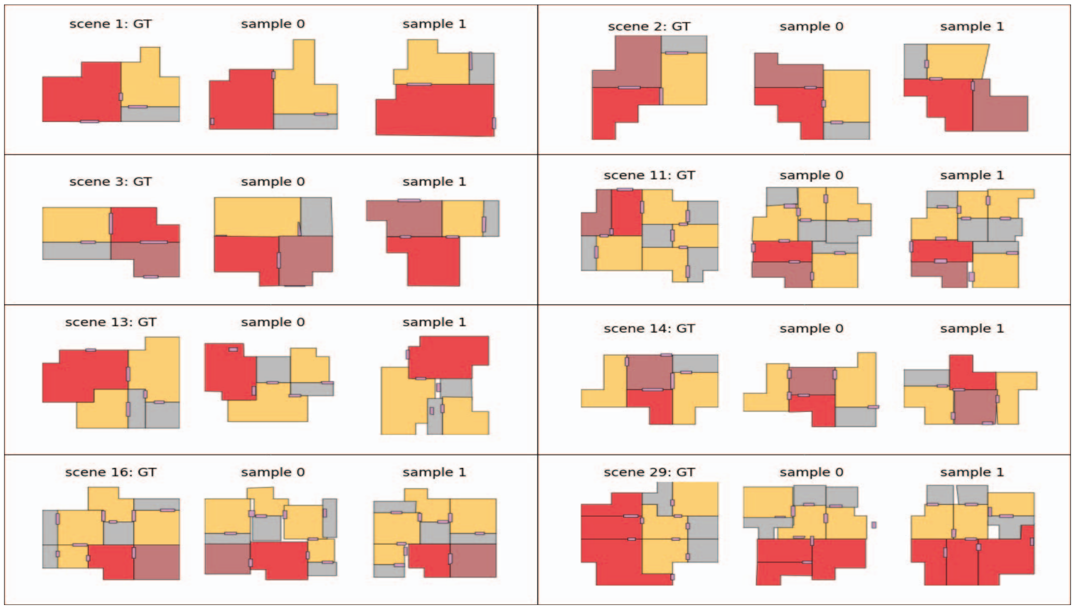


Figure 5: A few generation examples using our approach. For each scene the first floor plan is the Ground Truth and then two generations outputs of the Constrained Diffusion Model

4. Results

4.1. Architecture setup

4.1.1 Dataset

For our experimental setup, all experiments are done on the ProcTHOR-10K dataset [4]. This dataset contains a set of ten thousand house scenes generated procedurally. We use a sub-sample of this dataset, removing scenes that do not contain a Living room, a Kitchen and a Bedroom, leading to 5149 valid scenes. The scenes in this dataset are generated procedurally and contain some invisible walls. We merge the polygons with an invisible wall as a common side. These merged polygons are the room layouts we are trying to reconstruct. We rely on procthor10k's door annotations to extract the connectivity graphs from the scenes based on which rooms are connected by a door.

4.1.2 Initialization

To simulate the partial shapes extracted by a vision algorithm, we generate random contiguous splits of the groundtruth layout polygon for each room. These partial shapes are then used to compute their respective shape descriptors z_i and cc_i, set as features of the additional leaf nodes and generations constraint, respectively. While the room-node x_i are initialized with random noise. For the corner prediction task, the number of corners of each room is labeled with an integer value. These values are mapped in an orderly manner so that for $c_i > c_j$ we have $l_i > l_j$ with c the number of corner in the room, and l the label associated.

4.1.3 Models training

In our experiment, we use PyTorch Geometric [8] to implement our graph encoder and use PyTorch for the diffusion model and train with the Adam [17] optimizer.
The graph encoder is trained for 200 epochs with 10 epochs of learning rate warmup, followed by a linear decay. Following our graph encoder, every room node embeddings are extracted and are the input of a CORN [31] loss to perform Ordinal Regression.
The diffusion model is trained during 200000 steps with a batch size of 192 on a P100 NVIDIA GPU. The training takes approximately 1.5 days to complete. The initial learning rate is $1e-3$ which is divided by 10 after 100000 steps. The number of diffusion steps is 1000.

4.2. Floor plan reconstruction results

We illustrate the results obtained by our approach, with a scenario using one view per room, *i.e.* with exactly one partial shape assigned to each room. Compared to previous baseline methods such as HouseDiffusion [29]. Figure 4 shows the geometrical constraint (first line), the synthesized room and scene using our approach (second line), and a baseline diffusion model without the cross-attention module associated to the chain code (third line). We can see that our approach integrating the constraints correctly synthesize rooms featuring the expected geometry, while the more naive diffusion approach synthesizes arbitrary geometry that only preserves the correct number of corners. Figure 5 shows a wide variety of floor plans generated by our approach and illustrates qualitatively its capability to adhere to the geometrical constraints given by the partial shapes while providing a coherent map at the global scale. We may note that while the method is able to generate a very compact structure when few rooms are involved, some gaps appear between walls when dealing with maps associated with a large number of rooms. This characterizes the limit of the Diffusion Model to handle complex compact floor plans. However even in these cases we can see that the geometrical constraints are still respected (*e.g.* Scene 29).

4.3. Quantitative Analysis

4.3.1 Corner Prediction

We first show the advantage of using our enhanced graph G' in combination with our GNN to predict the number of corners. To this end, we compared our method to an alternative case where only one single view is seen for each room. In this case, we generate a second simplified graph solely based on G, attaching the single shape description to the node feature $x_i = z_i$. Furthermore, we investigated the performance obtained by using a GNN compared to an MLP, as well as the effectiveness of the Ordinal Regression approach. The results obtained in the various cases are depicted in Table 1, where we display the Mean Absolute Error (MAE) per class in the test set, with the number of corners used as the unit of measurement.
The top part (first three lines) corresponds to using the simplified graph with one partial view per room. The first two lines demonstrate the improvement provided by the Ordinal Regression technique [31] compared to a more trivial regression on the number of corners. The first line further highlights that our GNN architecture improves predictions, particularly for more complex rooms with a higher number of corners, as the GNN can utilize the global context of other rooms. The bottom part of the table shows the results obtained using our enhanced GNN capable of handling a variable number of partial views per room. We present results for scenarios with 50%, 70%, 1, and 2 partial view per room, where 50% indicates one partial view every two rooms, and 2 indicates two partial views for each room. Our enhanced graph approach always achieved superior accuracy compared to the simplified ones using MLP or GNN in similar conditions, i.e. one partial view per room (Enhanced GNN-1). Interestingly, it even obtained comparable, or sometimes better accuracy, when using only 70% of the views (Enhanced GNN-70%), indicating that the network effectively leveraged the model's sparsity through the additional node feature. Additionally, the Enhanced GNN-1 and Enhanced GNN-2 show that the results consistently improve with the addition of more views per room. This finding confirms that our method is robust and capable of effectively handling varying situations.

4.3.2 Floor plan reconstruction

We assess the accuracy of the floor plan reconstruction over our test dataset using two metrics. The first metric compares the reconstructed floor plan to the ground truth. For a given house, we compare each polygon of the generated rooms with the polygons of the corresponding ground truth rooms using Hu invariants [13]. These provide an error measurement between the polygons, taking into account scale and rotation invariance. Given that we consider only partial views, our reconstruction cannot be identical to the ground truth but provides, nevertheless, a reliable indicator of the shape reconstruction quality. The second metric employed is the modified Graph Edit Distance [1, 29] (mGED), measuring the similarity between the input connection graph and the estimated graph generated from the reconstructed floor plan. This metric thus measures the extent to which the connectivity graph is respected in the generated floor plans.

Results are shown in Table 2 and compared the results obtained with our approach using shape constraints (SC) to the one obtained without considering such constraints (NSC). We further assessed the influence of the corner number estimation accuracy toward this reconstruction using, respectively, the correct ground truth corner number (GT), the mean or median corner number associated with each type of room based on the ground truth, or using our dedicated GNN-based estimator. One may note that the results using our shape constraint (SC) yields consistently better results that without (NSC) for both Hu invariants and mGED metrics, highlighting the effectiveness of our chain-code based cross-attention module. While the scores obtained using the ground truth prediction of the number of corners are the lowest, we note that our results achieved using GNN-based predictions are remarkably close to this ideal scenario, and significantly outperform the naive use of mean or median predictions.

5. Conclusion and future works

We have proposed an approach for synthesizing floor plans consistent with partial and sparse constraints on the rooms geometry, as well as with hypotheses on rooms connectivity. Our approach is built on a graph-based framework, allowing to seamlessly handle the geometrical information provided by partial views. The generation method combines a Graph Neural Network to predict the number of corners with a diffusion model for floor map synthesis, which takes into account geometric information through a cross-attention module. The general pipeline is robust to an arbitrary number of input views per room. This is partic-

Configuration	MAE per class (↓) $\pm SE$				
	4 corners	6 corners	8 corners	10 corners	12 corners
MLP-Regression	0.16 ± 0.0	0.9 ± 0.0	2 ± 0.0	3.9 ± 0.01	5.67 ± 0.0
MLP-Ordinal [31]	0.07 ± 0.0	0.59 ± 0.0	1.55 ± 0.0	3.28 ± 0.03	5.33 ± 0.0
GNN-Ordinal	0.19 ± 0.0	0.67 ± 0.0	1.45 ± 0.01	2.27 ± 0.04	4.67 ± 0.24
Enhanced GNN-50%	0.47 ± 0.0	0.64 ± 0.0	2.52 ± 0.01	4.1 ± 0.08	5.9 ± 0.28
Enhanced GNN-70%	0.28 ± 0.0	0.42 ± 0.0	1.41 ± 0.01	2.1 ± 0.11	3.1 ± 0.19
Enhanced GNN-1	0.0 ± 0.0	0.03 ± 0.0	0.42 ± 0.01	1.44 ± 0.07	3.6 ± 0.31
Enhanced GNN-2	**0.0 ± 0.0**	**0.02 ± 0.0**	**0.28 ± 0.02**	**0.9 ± 0.04**	**3.2 ± 0.28**

Table 1: Results for the Corner Prediction task: The MAE is in terms of number of corners ± Standard Error. The MLPs take in a shape descriptor and output a number of corner. GNN-Ordinal doesn't make use of our graph construction technique and has descriptors as node features. For the results utilizing our graph construction, Enhanced GNN-50% means that only 50% of rooms have geometrical information. Where Enhanced GNN-2 has 2 shape descriptor per room. Predictions are always made for the entire set of rooms, whether or not it has a shape descriptor

Configuration	Hu invariants [13] (↓)	mGED [1] (↓)
NSC/GT	*0.32 ± 0.01*	*1.31 ± 0.12*
NSC/mean	0.40 ± 0.04	1.4 ± 0.11
NSC/median	0.30 ± 0.02	1.42 ± 0.18
NSC/GNN	0.31 ± 0.02	1.33 ± 0.12
SC/GT	*0.11 ± 0.01*	*0.91 ± 0.11*
SC/mean	0.27 ± 0.05	1.5 ± 0.13
SC/median	0.28 ± 0.01	1.7 ± 0.23
SC/GNN	**0.143** ± 0.01	**1.13** ± 0.13

Table 2: Quality of the reconstruction given the Hu invariants and the mGED. NSC means No Shape Constraint (the diffusion process is therefore not constrained by the partial shape). Mean / Median, GNN and GT (Ground Truth) is the method for estimating the number of corners per room. $\pm SE$

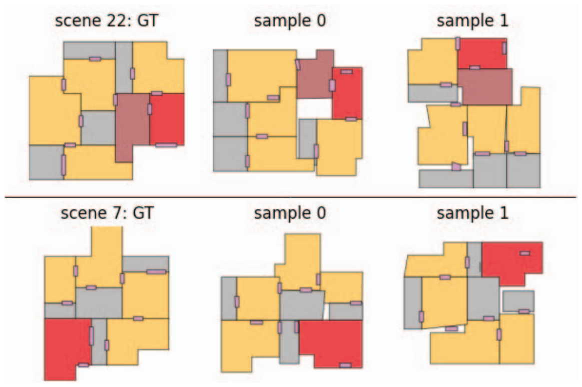


Figure 6: Two examples of failure cases. Generally the network has trouble generating very compact floorplans that contain a lot of rooms. And a yellow room has the wrong number of corners (2 more than it should) predicted leading to wrong or incoherent shapes

ularly useful in real-case applications, where more images can be added in a subset of the rooms to improve the reconstruction of the global map, without requiring a dense and complete description of all the rooms, such as through panoramic acquisition.

More generally, this work has shed light on the methodology needed to solve the problem of floor plans from sparse views. The use of graphs enriched by the different observations seems to be a major advantage in dealing with the sparse character of the observations. However some improvements could be made on that aspect. First, other shape descriptors could be studied and may further improve the results. Using chain code encoding instead of, or in addition to, the Zernike moments may be beneficial fot the GNN. Second, our model shows some failure cases that can be seen by the gaps between the walls shown in Figure 5, as well as in Figure 6 in predicting the wrong number of corners and thus generating a wrong shape. Improving the encoder may lead to improved results in allowing it to handle additional properties that are inherently part of the architecture such as in Equivariant Networks [3, 10, 20]. Furthermore we can also note that the Graph Construction could be extended to address the multi-modal aspect of the considered real data (*e.g.* rental listings), as pre-processed textual and visual data could co-exist in this framework (*e.g.* using CLIP embeddings [27]). For the generation part, the choice of shape descriptor remains important to study. A general idea for the future would be to use the graph encoder as a shape aggregator, so that every room embedding contains the combined geometrical information. These room embeddings would then be used as the main constraint for the generation, enabling us to jointly optimize the two networks.

References

[1] Zeina Abu-Aisheh, Romain Raveaux, Jean-Yves Ramel, and Patrick Martineau. An exact graph edit distance algorithm for solving pattern recognition problems. *ICPRAM - Conference on Pattern Recognition Applications and Methods, Proceedings*, 1, 01 2015. 7, 8

[2] Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. *arXiv:2208.04202*, 2023. 4

[3] Pim de Haan, Maurice Weiler, Taco Cohen, and Max Welling. Gauge equivariant mesh cnns: Anisotropic convolutions on geometric graphs. *arXiv:2003.05425*, 2021. 8

[4] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Jordi Salvador, Kiana Ehsani, Winson Han, Eric Kolve, Ali Farhadi, Aniruddha Kembhavi, and Roozbeh Mottaghi. ProcTHOR: Large-Scale Embodied AI Using Procedural Generation. In *NeurIPS*, 2022. 6

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. 5

[6] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*, 2021. 2

[7] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. *ICLR*, 2022. 2

[8] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. 6

[9] Herbert Freeman. Computer processing of line-drawing images. *ACM Comput. Surv.*, 6(1):57–97, mar 1974. 5

[10] Fabian B. Fuchs, Daniel E. Worrall, Volker Fischer, and Max Welling. SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks. *NeurIPS*, 2020. 8

[11] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *NeurIPS*, 2018. 2

[12] Sepidehsadat Hosseini, Mohammad Amin Shabani, Saghar Irandoust, and Yasutaka Furukawa. JigsawPlan: Room Layout Jigsaw Puzzle Extreme Structure from Motion using Diffusion Models. *arXiv:2211.13785*, Nov. 2022. arXiv:2211.13785 [cs]. 3

[13] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 8(2):179–187, 1962. 7, 8

[14] Ruizhen Hu, Zeyu Huang, Yuhan Tang, Oliver Van Kaick, Hao Zhang, and Hui Huang. Graph2plan. *ACM Transactions on Graphics*, 39(4), aug 2020. 2

[15] Jiahui Huang, Zheng-Fei Kuang, Fang-Lue Zhang, and Tai-Jiang Mu. Wallnet: Reconstructing general room layouts from rgb images. *Graphical Models*, 111:101076, 2020. 2

[16] A. Khotanzad and Y.H. Hong. Invariant image recognition by zernike moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(5):489–497, 1990. 3

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 6

[18] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ICLR*, 2017. 2

[19] Devin Kreuzer, Dominique Beaini, William L. Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *NeurIPS*, 2021. 2

[20] Yi-Lun Liao and Tess Smidt. Equiformer: Equivariant graph attention transformer for 3d atomistic graphs. *NeurIPS*, 2023. 8

[21] Shengjie Luo, Tianlang Chen, Yixian Xu, Shuxin Zheng, Tie-Yan Liu, Liwei Wang, and Di He. One Transformer Can Understand Both 2D & 3D Molecular Data. *arXiv:2210.01765*, 2022. 5

[22] Dominic Masters, Josef Dean, Kerstin Klaser, Zhiyi Li, Sam Maddrell-Mander, Adam Sanders, Hatem Helal, Deniz Beker, Andrew Fitzgibbon, Shenyang Huang, Ladislav Rampášek, and Dominique Beaini. GPS++: Reviving the Art of Message Passing for Molecular Property Prediction. *arXiv:2302.02947*, 2023. 2, 4

[23] Luis Müller, Mikhail Galkin, Christopher Morris, and Ladislav Rampášek. Attending to graph transformers. *arXiv:2302.04181*, 2023. 2

[24] Nelson Nauata, Kai-Hung Chang, Chin-Yi Cheng, Greg Mori, and Yasutaka Furukawa. House-GAN: Relational generative adversarial networks for graph-constrained house layout generation. In *ECCV*, pages 162–177. Springer, 2020. 2

[25] Nelson Nauata, Sepidehsadat Hosseini, Kai-Hung Chang, Hang Chu, Chin-Yi Cheng, and Yasutaka Furukawa. House-GAN++: Generative Adversarial Layout Refinement Networks. *CVPR*, 2021. 2

[26] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *ICLR*, 2021. 2

[27] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *arXiv:2103.00020*, 2021. 8

[28] Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *NeurIPS*, 2023. 2

[29] Mohammad Amin Shabani, Sepidehsadat Hosseini, and Yasutaka Furukawa. HouseDiffusion: Vector Floorplan Generation via a Diffusion Model with Discrete and Continuous Denoising. *CVPR*, 2022. 1, 2, 4, 5, 6, 7

[30] Mohammad Amin Shabani, Weilian Song, Makoto Odamaki, Hirochika Fujiki, and Yasutaka Furukawa. Extreme structure from motion for indoor panoramas without visual overlaps. In *ICCV*, October 2021. 1, 2

[31] Xintong Shi, Wenzhi Cao, and Sebastian Raschka. Deep neural networks for rank-consistent ordinal regression based on conditional probabilities. *Pattern Analysis and Applications*, jun 2023. 4, 6, 7, 8

[32] Cheng Sun, Chi-Wei Hsiao, Min Sun, and Hwann-Tzong Chen. Horizonnet: Learning room layout with 1d representation and pano stretch data augmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 1047–1056, 2019. 2

[33] Jiahui Sun, Wenming Wu, Ligang Liu, Wenjie Min, Gaofeng Zhang, and Liping Zheng. Wallplan: Synthesizing floorplans by learning to generate wall graphs. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 41(4), jul 2022. 2

[34] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *ICLR*, 2022. 2

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017. 2, 5

[36] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *ICLR*, 2018. 2

[37] Zhanghao Wu, Paras Jain, Matthew A. Wright, Azalia Mirhoseini, Joseph E. Gonzalez, and Ion Stoica. Representing long-range context for graph neural networks with global attention. *NeurIPS*, 2022. 2

[38] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation? *NeurIPS*, 2021. 2, 4

[39] Yuanwen Yue, Theodora Kontogianni, Konrad Schindler, and Francis Engelmann. Connecting the Dots: Floorplan Reconstruction Using Two-Level Queries. *CVPR*, 2023. 2

[40] Weidong Zhang, Wei Zhang, and Yinda Zhang. Geolayout: Geometry driven room layout estimation based on depth maps of planes. *ECCV*, 2020. 3

[41] Chuhang Zou, Jheng-Wei Su, Chi-Han Peng, Alex Colburn, Qi Shan, Peter Wonka, Hung-Kuo Chu, and Derek Hoiem. Manhattan room layout reconstruction from a single 360 image: A comparative study of state-of-the-art methods. *International Journal of Computer Vision*, 129(5):1410–1431, 2021. 2