SYMPOSIUM ON SIMULATION IN ARCHITECTURE + URBAN DESIGN
SimAUD 2020

# Floorplan Embedding with Latent Semantics and Human Behavior Annotations


## ABSTRACT

Floorplans provide top-view representations of buildings that highlight key relationships between spaces and building components. In the last few decades, different approaches have been proposed to compare and catalogue different floorplans for design exploration purposes. Some approaches have considered floorplans as images, while others represented them as graphs. However, both image and graph-based approaches have failed to extract and utilize essential low-level space semantics and structural features. Further, they do not encode information about space utilization determined by people movement and activities in space, which are critical to analyze a building layout. To address these issues, we use deep learning techniques to develop a floorplan embedding – a latent representations of floorplans, which encodes multiple features for clustering and comparison purposes. Specifically, we propose a novel framework that uses an attributed graph as an intermediate representation to encode space semantics, structural information and crowd behavioral features. We train Long Short-Term Memory (LSTM) autoencoders to represent these graphs as vectors in a continuous space. In addition, we contribute a floorplan dataset augmented with semantic and simulation-generated behavioral features. These representations spark new exciting opportunities for next-gen clustering and design exploration tools.



### Author Keywords

Floorplan Embedding; Attributed Graph; Design Exploration; Design Semantic Features; Human Behavioral Features; LSTM Autoencoder

### ACM Classification Keywords

G.3 Mathematics of Computing: PROBABILITY AND STATISTICS; I.6.3 SIMULATION AND MODELING : Applications; I.6.5 SIMULATION AND MODELING : Model Validation and Analysis; J.5 ARTS AND HUMANITIES: Architecture; J.6 COMPUTER-AIDED ENGINEERING: .


## 1 INTRODUCTION

Floorplans provide a well-established mean to represent buildings. As such, they afford a wide range of design activities such as ideation, analysis, evaluation and communication. Computer-Aided Design (CAD) and Building Information Modeling (BIM) approaches support the creation of digital building models, from which floorplans can be extracted. Current approaches, however, do not support the systematic comparison of floorplan features derived from geometric and semantic properties as well as more advanced performance metrics, such as space utilization and occupant behavior generated via simulation.

While research mostly developed in Computer Graphics proposed computational strategies to extract geometric floorplan features by processing them as images [20, 9] or graphs [25, 23], these approaches ignore semantic information that describes the function of each space, as well as time-based analytics of human movement and activities. Because of this, architects and planners do not have digital means to analyze, compare, and sort through a large number of floorplans that have similar of different features.

To address this issue, we propose *floorplan embedding* – "latent" representations of building layouts using an attributed graph as intermediate representation which facilitate semantically meaningful comparisons that account for not only design and structural features, but also human behavior features. Specifically, a Long Short-Term Memory (LSTM) [16] autoencoder [15] is trained to represent these graphs as vectors in a continuous space. These vector representations are then used to cluster and query floorplans with similar characteristics and attributes. Some of the advantages of presenting floorplans as vectors are: (a) compact representation, (b) efficient way to compare designs and fast retrievals, (c) scoring designs and providing feedback/recommendations, and (d) categorizing design according to our target features. In addition, since there is not any dataset augmented with mentioned features, we release a novel annotated floorplan dataset with semantics and human behavioral features generated from simulations.

The key contributions in this paper can be summarized as follows: (i) intermediate representation of floorplans as attributed graphs and augmented with crowd behavioral features (ii) novel unsupervised deep learning model to learn a meaningful vector representation of floorplans (iii) creation of a floorplan dataset augmented with semantic and crowd behavioral attributes generated from simulations.

## 2 RELATED WORK

We aim to empower floorplan representation by representing floorplans as vectors which the design semantics attributes, low-level structural characteristics and also crowd behavioral attributes are encoded. In this section, we will review some of recent works for comparing floorplans and the way they are

SimAUD 2020 May 25-27 Vienna, Austria

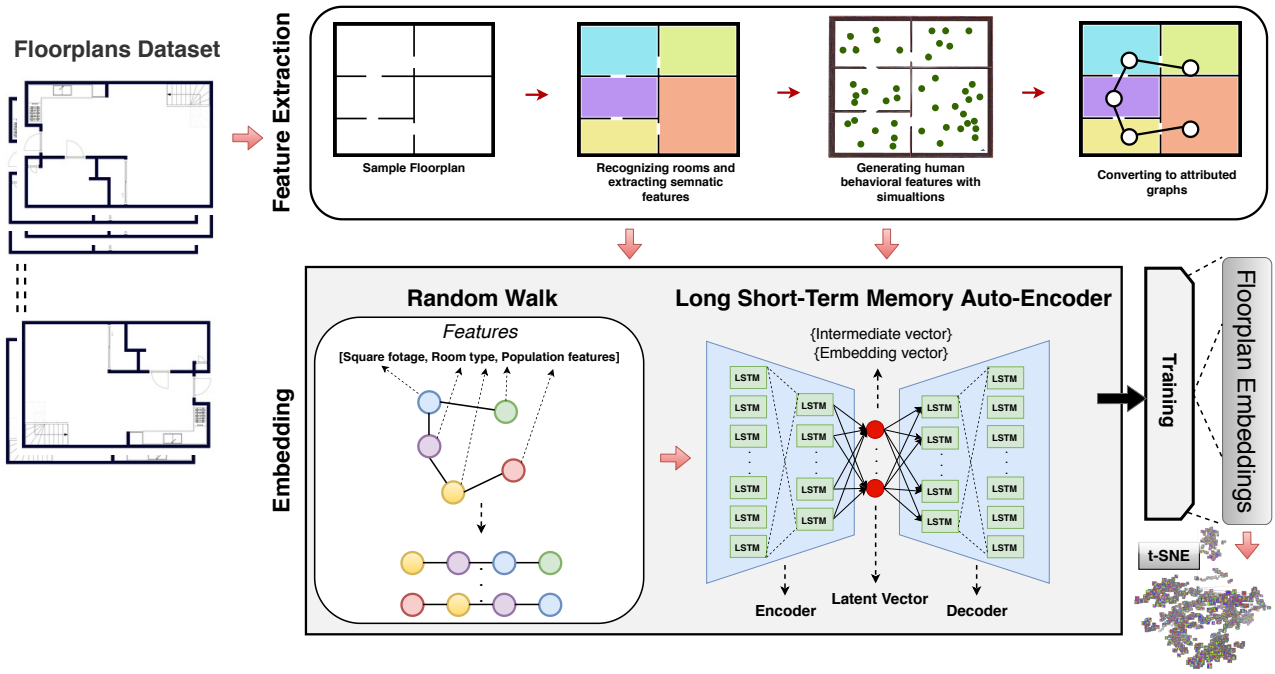



**Figure 1**: An overview of the proposed framework. In pre-processing, rooms and their design semantic properties are identified, then a 3D model is generated for crowd simulator to generate dynamic behavioral features, and at the end, the floorplan is converted to an attributed graph. In the embedding part, first a random walk is performed to convert attributed graphs to a set of sequences. Next, a LSTM autoencoder is proposed for training and floorplan predictions. The autoencoder has a 2 layer LSTM: first has 84 and second one has 64 LSTM units. The encoder has same architect in reverse.

represented. The prior works can be mainly divided into three categories: image-based, graph-based and symbol-spotting methods.

Several approaches to compare floorplans have been proposed based on conventional image processing techniques [6, 13]. In these approaches features like Histogram of Oriented Gradients (HOG) [7], Bag of Features (BOF) [18], Local Binary Pattern (LBP) [2] and Run-Length Histogram [8] have been used for extracting features from floorplan images. The extracted features are used for comparing floorplans. In [26], a deep Convolutional Neural Network (CNN) is proposed to present floorplans as vectors to address the limitation of conventional image processing techniques for extracting meaningful features. These methods are based on object-centric databases in which floorplans are annotated with furniture or specific visual symbols, and only the visible features from images are encoded. These features, however, are not semantics and also do not correctly capture the design structures.

Graph matching is another track for comparing floorplans. In these methods floorplans are represented as graph and graph matching methods are applied for scoring similarity. Different strategy is proposed for presenting floorplans as graph. In [25], floorplans are modeled as graphs to capture adjacency between the rooms and augmented with arrangement of annotated furniture. In [23], the floorplans are converted to graphs by considering the connectivity, room type and accessibility in graph construction. In [1], a framework for automatic analysis of floorplans is proposed in which floorplans are represented by an attributed graph where attributes for each room are extracted using SURF technique. In [24], three different representation layers are merged to form a graph representation. These representations are area, furniture style, and an adjacency matrix. Since in all of these methods floorplans are represented with graph, the buildings structure like connection of rooms are captured. Some of them add some features to nodes which are mostly extracted features by image processing techniques. In addition all of these methods use graph representation as final representation and do not generate vector representation. Then graph matching methods are used for finding similarity and other type of operations are applied directly over graphs. Since it might be the case that there is no exact matching, subgraph matching [1, 31] is proposed. It means if two graphs are not isomorphic but their parts are, they can be detected with subgraph matching.

Symbol spotting is another mechanism which has been used for floorplans comparison and retrieval. It is a special case of Content-based Image Retrieval (CBIR) [14, 22] which is used for technical document analysis. In symbol spotting, a query is submitted and the system retrieves zones from the document which are likely containing the query. This query could be a cropped image or hand-sketched. Pattern recognition techniques are utilized for this aim, as example moment invariants like Zernike moments in [17]. Reducing search space in spotting symbols is proposed based on hash-

ing of shape descriptors of graph paths(Hamiltonian paths) in [10]. SIFT/SURF [30, 3] features that provide an efficient and scale-invariant are commonly used for spotting symbols in graphical documents. Symbol spotting methods are applied to small datasets which do not have complex images, and they are only applicable for retrieval purpose.

## 3 METHODOLOGY

The proposed framework is illustrated in Fig. 1. It consists of two components. The first component is for preprocessing floorplans and convert them into attributed graphs. The result of this step is a novel floorplan dataset augmented with design semantic and crowd behavioral features. The second component is the embedding for creating latent vectors of these graphs. Further details on each component are presented in the following subsections.

### 3.1 Dataset

HouseExpo dataset [19] is used in this work. It includes about 35000 2D floorplan layouts that are presented in JavaScript Object Notation (JSON) format. They are mostly floorplan segments belong to big buildings. A sparse labeling for layout components is also given for each floorplan. Figure 2 shows an image of a raw floorplan compiled as image. There are about 25 different components types in the whole dataset. Some of them, however, have similar semantics (e.g. toilet and bathroom, terrace and balcony, etc.). We reduce the components types to 11, considering only single type-name for rooms labeled for similar purposes and removing minor components like freight elevator.

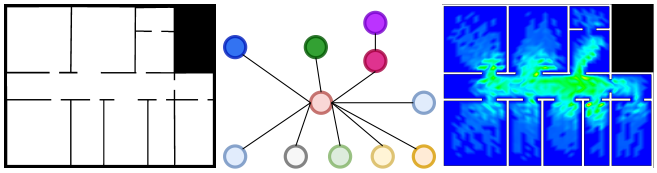



**Figure 2**: A sample raw floorplan from the dataset. This image is the result of compiling JSON file as image. The black pixels are wall (or outdoors) and white pixels are building components (or indoors). These floorplans are converted to graphs and augmented with design semantic and bahvioral features in our preprocessing step.

### 3.2 Floorplans to Attributed Graphs

In order to convert floorplans into attributed graphs, first we recognize all potential room segments by a series of image processing operations. Then, we assign labels based on provided annotations. Some of the floorplans are not annotated perfectly (e.g. the given coordinates for the labels are not match with room coordinates). In such cases, we calculate overlaps of each room with given label coordinates, and then assign labels to rooms which have the maximum overlap.

The rooms are the nodes in the graph and edges are the potential connections between them. There will be an edge between two rooms if there is an immediate door between them. Edges are formed by image processing techniques. To detect doors, we create a combined image of two rooms and apply blob detection on it. If the number of blobs found is one, it means there is a door between them, otherwise, there is no connection between the two rooms. Up till point, graphs are representing the structure of floorplans but they are not attributed. Next, we assign to nodes (rooms) their respective semantic features like square footage and room type, which are calculated using image processing techniques. To generate crowd related features, the 2D floorplans are converted to 3D models loadable in a crowd simulator, SteerSuits [27]. The simulator automatically populates virtual agents in each room with the target to exit the floorplan. It then calculates features like maximum, minimum and average evacuation times and traveled distances, as well as overall exit flow for all the agents in room. More details on these crowd aware features is presented in the following subsection. This part is done in an end to end procedure and does not need human interactions.

As mentioned, as a part of this work we are generating a novel floorplan dataset augmented with semantic and crowd behavioral features which is the result of this section. In this dataset, for each sample we have a JSON file augmented with semantic and crowd behavioral features.

### 3.3 Features

This section presents the procedure for generating a floorplan dataset augmented with semantic and crowd behavioral features. At this point, we have floorplans which are presented as attributed graphs. The formation of a graph captures the structure of a floorplan, and attributes of nodes present its features. The available set of features can be seen in Table 1. These are divided into two groups: design semantic and crowd behavioral features which are generated with simulations. Design semantic features include room types and square footage which are generated by image processing techniques. Since room types are categorical features, they are presented as one-hot vector with 11 dimension and footage square is represented with an one dimensional scalar. The total dimension of semantic features is 12. All crowd behavioral features are represented with a one dimensional scalar value. In total we have 9 crowd features which lead to have a 9 dimensional vector for crowd features. Sum of all feature dimensions is 21.

However, our approach is not bound to use only the selected features, and more can be adapted into the framework depending on the application. The values of scalar features have different ranges. In order to keep all of them within a same range for robust training, we normalize them between [0, 1] except for features which are presented with one-hot vector. All the outliers are removed before normalization.

### 3.4 Floorplan Embedding

This section talks about the conversion of attributed graphs into continues latent vectors which encode both structure and semantics of the floorplans, as well as their crowd behavioral features. We use graph embedding approach to transform graph nodes, edges, and their features into a vector space (lower dimensional) while preventing any information loss. Graphs, however, are tricky to deal with because they can vary in terms of their scale, specificity, and subject [5, 11].

| Feature Class | Features types | Dimension |
|---|---|---|
| **Design semantic** | Square footage | 11 |
| | Room type | 1 |
| **Behavioral** | Not completed agents | 1 |
| | Max evacuation time | 1 |
| | Min evacuation time | 1 |
| | Exit flow rate | 1 |
| | Completed agents | 1 |
| | Max traveled distance | 1 |
| | Ave evacuation time | 1 |
| | Avg traveled distance | 1 |
| | Min traveled distance | 1 |

**Table 1**: First column shows the two feature classes, second column shows features available in each class and third columns is their dimension. Dimension for semantics features is 12 and for behavioral is 9, in total the features dimension is 21, all scaled between [0 1].

There exist some approaches to perform graph embedding [12, 29, 28, 21]. However, they are only suitable for unattributed graphs and mostly capture just the graph structure. But in our case, the graphs are attributed. So in order to account for these attributes, we proposed an (Long Short Term Memory) LSTM autoencoder. Auto-encoders [15] are trained to learn the full properties of the data and reconstruct their inputs. They generally have two parts: an encoder that maps the input to an intermediate representation and a decoder that reconstructs the inputs from intermediate representation. The intermediate representations are latent vectors. LSTM is a recurrent neural network (RNN) for capturing long-distance dependencies in sequential data and also supports varying data lengths.

We propose a LSTM autoencoder to learn embedding space in a way that keeps floorplans of similar structure, design semantic and crowd behavioral features, close to each another in the embedding space. Our proposed model is illustrated in Fig. 1. We learn the function for mapping graph $G$ to vector $R$ in a d-dimensional space. In a floorplan (graph) $G = (V, E)$, $V$ denotes its vertex set (rooms) and $E \subseteq VxV$ denotes its edge set (potential doors between rooms). We have unlabeled graphs in our dataset. Each node has a constant dimensional feature-vector $F_V$.

Adjacency matrix is one of the ways to present graphs as input to algorithms. But since the graphs are varying in number of nodes and edges, presenting them as adjacency matrix opens new challenges. This is because the dimension of the adjacent matrix is different for different graphs. To address this, we convert graphs to multiple varying length sequences. This conversion is done by using Random Walk. In a random walk we start from a given source node and the next node will be selected randomly with probability $1/D(N)$, where $D(N)$ is the degree of node $N$. Each graph is converted to a set of sequences and these sequences are feed to the model for training. Then the average latent vectors of corresponding sequences to a graph are used as a representation vector for the graphs (Equation 1).

$$\Phi(G) = \frac{1}{Nseq} \sum_{n=1}^{Nseq} RS_n \tag{1}$$

The $Nseq$ is the number of sequences and $RS$ is corresponding latent vector to each sequence.

### 3.5 Training

Some of malformed graphs are removed from the dataset. We use about 17000 floorplans' graphs which are converted to a set of sequences. We run random walks on the graphs to generate corresponding sequences. These sequences are feed into our model for training, both as input and output since the method is unsupervised. The loss function in our model is Mean Squared Error (MSE) (Equation 2).

$$loss(s) = \frac{1}{|s|} \sum_{i=1}^{|s|} \bar{Y} - Y \tag{2}$$

Which $s$ is the given sequence and $|s|$ is the number of nodes in the sequence. $\bar{Y}$ is the reconstructed vector for a node and $Y$ is the true vector for that node. Loss function calculates the difference between reconstructed vector(output of decoder) and input sequences which each node has feature vectors $F_v$. In other words the encoder is trying to reconstruct the node features in the sequence.

## 4 CASE STUDY AND APPLICATIONS

This section validates and showcases the potential of our embedding methodology with the help of 3 different use cases. For following studies, we trained three models with the architecture described. The difference between these models is the considered features. First model is trained only with design semantic features, second only with behavioral features and the third with all of the features.

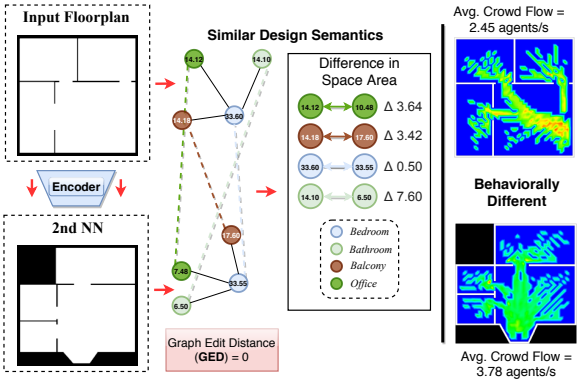
### 4.1 Pairwise Similarity between Floorplans

In this use case we demonstrate a pairwise comparison between input floorplan and its second nearest neighbour from the embedding space. Underline graphs of both floorplans are compared using a popular graph similarity distance metric, Graph Edit Distance (GED) [4]. The edit distance between $G1$ and $G2$, GED(G1, G2), is the count of edit operations that transform $G1$ into $G2$, where the edit operations on a graph $G$ can be an insertion or deletion of an edge or node. All edit operations have the same constant cost which is 1. If two graphs are identical, their GED is 0.

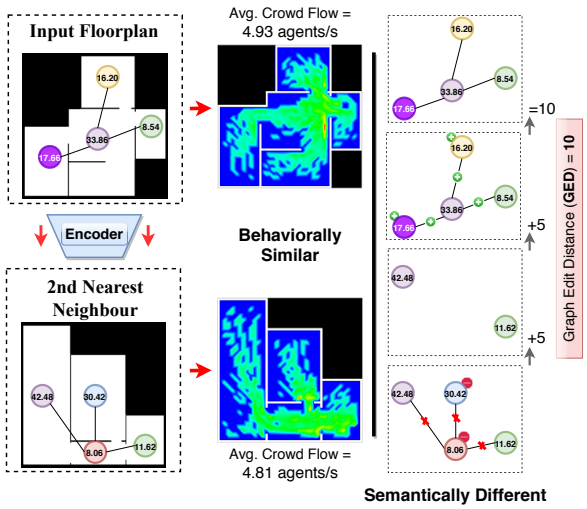
First, we retrieved a semantically similar design (second nearest neighbour) from the embedding space for the given input floorplan such that they have similar structural attributes but significantly different behavioral attributes, Figure 3a. The GDP value is reported as 0, showcasing that the graphs are similar in their design semantics. Average exit flow values and color-coded density heatmaps are also shown for both floorplans. Input floorplan yielded comparatively lower exit flow than its nearest neighbour found from the embedding.

In next scenario, we retrieved a behaviorally similar floorplan (second nearest neighbour) from the embedding space for the given floorplan such that they have similar behavioral attributes but significantly different design semantics. A GDP value of 10 is reported from the graphs comparison, showcasing the two graphs are design semantically different. However, their corresponding floorplans yielded similar exit

flow values. A GDP graph transformation as well as density heatmaps are shown, Figure 3b.




(a) Two floorplans which are structurally similar (Graph Edit Distance (GED) = 0) but have different crowd behavioral attributes.




(b) Two floorplans which are behaviorally similar (having similar average exit flows) but design semantically different (Graph Edit Distance (GED) = 10).

**Figure 3**: Pairwise similarity comparison between input floorplan and queried nearest neighbour from the embedding space.

### 4.2 Behavioral and Geometrically-powered Floorplans Retrieval

In this use case we demonstrate the potential of using our embedding methodology to retrieve geometrically-powered (includes static features), behaviorally-powered (includes crowd-based features) and combined, similar and related floorplans.

Figure 5a showcases an example for geometrically-powered floorplans retrieval. This embedding setting contains a 12 dimensional design semantic vector, 11 for room types and 1 for room dimensions. Given an input floorplan, a set of 5 nearest neighbors from the embedding space are retrieved. For an embedding space to be meaningful and valid, a queried graph should have itself as the first nearest neighbor during a retrieval, and therefore, in the figure, the first neighbor is same as the queried floorplan itself. For first query, the second and third nearest neighbours have same structure, the hallway in the middle and two bedrooms, one kitchen and one bathroom around it. Though, they have different design semantic features as annotated in the image. Fourth neighbour has close structure and only with missing the kitchen node. The last neighbour almost has same structure but the middle node is different.

Figure 5b showcases an example for behaviorally-powered floorplans retrieval. In this example the query graph is the same as last one and we retrieve the top 5 nearest neighbours. The model trained by behavioral features is used. All the neighbors have one node in the middle and four in the surroundings with same node degree. The second row shows simulation heatmaps for behavioral features. Since semantic features are not considered in this use case, two of the neighbours have totally different room types and more-less similar square footage. Also, the second nearest neighbour is exactly same as second nearest neighbour in case 1. It shows even without design semantic features, the structure and behavioral features can capture floorplans similarity.

Figure 5c showcases an example of floorplans retreival from a collectively trained model with both semantic and behavioral features. The features dimension for each node is 21. The top 5 nearest neighbours for the same input floorplan are retrieved from the embedding. Since in this case all of the features are considered, the neighbours must follow similar structure, similar design semantic and also similar behavioral pattern. Regarding structure, they almost have same structure, one node is missing in fourth and fifth neighbours and one node is different in fourth neighbour. But node degrees and structure of the graphs are almost same. The heat-maps visualizing the behavioral features are shown in second row and as it is clear their similarity decreases for later neighbours. Semantic features as annotated in the images are more-less similar. There are some noticeable points in this study. First the floorplan in second rank has the second rank in two later cases. This shows the validity of embedding space which the similarity and dissimilarity between floorplans are captured perfectly in vectors in embedding space. The third neighbours is also seen in last two cases. Semantically it was the third neighbour but behaviorally it was in fourth rank, the accumulation of both class features push it in third rank. This behavior is another justification for embedding space validity and benefit of considering behavioral features. Behavioral features weakly represent the semantic attributes and integrating them helps to fin the best fit nearest neighbours. Third point is the presence of a new floorplan in fourth rank which is not seen in both cases but the accumulation of features classes as discussed later lead it to get fourth rank. Fifth neighbour was the fourth neighbour in first study and was not seen in second study. It means this floorplan has similar structure design semantic but behaviorally different pattern which their accumulation keeps it in fifth rank.

In order to have an evaluation over embedding space, we find the rank of each floorplan by itself between top 5 nearest neighbours. A meaningful embedding space should lead to a case that each graph has its own as first nearest neighbor. We repeat this process for all three models, model trained with only design semantic features, model trained with only behavioral features and model trained with all features. We got 100% for three models, table 2. As another metric, we generate one proxy graph for each floorplan. Proxy graphs are the result of running random walk one more time that makes different sequence for each floorplan. These proxy graphs are not involved in training. We feed them to the trained model and add their representative vectors to embedding space. Then we find the top 10 nearest neighbors for each floorplan. The reason for top 10 nearest neighbours is to make sure we see them between neighbours. Having these proxy graphs in higher ranks shows the validity of embedding space. The result is provided in table 2 for all three models. The reason that all of the proxy graphs are not seen in second rank is because our graphs are unidirectional and we allow having loop in random walk. The randomness in random walk could lead to sequences that is back and fourth between only two nodes or a repeated subsequence because of loop. In both cases the generated sequences are not presenting the graph properly. This situation also has dependency to node degrees in the graphs since next node in random walk is selected based on weighted probability of nodes degree. This can be addressed by running random walk more than one time for each node or avoiding loops, both in training and test time.

### 4.3 Generation of a Composite Floorplan

Embedding spaces have commonly been used to make predictions and retrieving similar objects given some input criterion or an object. In this use case we demonstrate that our embedding methodology can also be used to generate a compound floorplan with collective features given multiple floorplans with a set of completely or partially different features. Figure 4 showcases one such example. Two floorplans with 2 and 3 different room types respectively, are given as input to our autoencoder, and the first nearest neighbour is a composite floorplan consist of 5 different room types. It not just contains all the features from input floorplans but also maintains a significant proportion of their geometric symmetries.

| ***Ranks/Models*** | **Model1** | model2 | **Model3** |
|---|---|---|---|
| First rank(query graph itself) | 100% | 100% | 100% |
| Second rank(proxy graph) | 86% | 83% | 85% |
| Third rank(proxy graph) | 13% | 13% | 14% |
| Fourth rank(proxy graph) | 1% | 3% | 1% |
| Fifth rank(proxy graph) | 0 | 2% | 0 |

**Table 2**: This table shows the percentage of the floorplans that have them-self as first nearest neighbours and the rank percentage of their proxy graphs.

## 5 CONCLUSION AND DISCUSSION

In this paper we represented floorplans as a vector in a continuous space. Specifically, the framework consists of two components. The first component is for representing floorplans as intermediate attributed graphs and the second one is the embedding for transforming attributed graphs to a vector. For converting floorplans to the attributed graphs we designed an automated tool based on image processing techniques that gets the image or JSON file of a floorplan as input and convert it to an attributed graph augmented with design semantic and crowd behavioral features generated by simulation. For embedding, we propose an LSTM Autoencoder that converts these graphs to vectors. The floorplans with similar structure, design semantic and crowd behavioral features have close vector in embedded space, showing promising results in different applications related to architecture design and analysis.

We posit that this contribution paves the way for novel, ground-breaking developments for human-in-the-loop or automated floorplan clustering, exploration, comparison and generation. By encoding latent features in floorplan embeddings, designers can store multi-dimensional information of a building design and performance to quickly identify floorplan alterations that share similar or different features. While in this paper we encode features derived from dynamic crowd simulations of building occupancy, the proposed approach can virtually scale to encode any kind of static or dynamic performance metric.

## REFERENCES


1. Ahmed, S., Weber, M., Liwicki, M., Langenhan, C., Dengel, A., and Petzold, F. Automatic analysis and sketch-based retrieval of architectural floor plans. *Pattern Recognition Letters 35* (2014), 91–100.
2. Ahonen, T., Hadid, A., and Pietikainen, M. Face description with local binary patterns: Application to face recognition. *Transactions on Pattern Analysis & Machine Intelligence*, 12 (2006), 2037–2041.
3. Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. Speeded-up robust features (surf). *Computer vision and image understanding 110*, 3 (2008), 346–359.
4. Bunke, H. What is the distance between graphs. *Bulletin of the EATCS 20* (1983), 35–39.
5. Cai, H., Zheng, V. W., and Chang, K. C.-C. A comprehensive survey of graph embedding: Problems, techniques and applications, 2017.
6. Chechik, G., Shalit, U., Sharma, V., and Bengio, S. An online algorithm for large scale image similarity learning. In *Advances in Neural Information Processing Systems* (2009), 306–314.
7. Dalal, N., and Triggs, B. Histograms of oriented gradients for human detection (2005).
8. de las Heras, L.-P., Fernández, D., Fornés, A., Valveny, E., Sánchez, G., and Lladós, J. Runlength histogram image signature for perceptual retrieval of architectural floor plans. In *Workshop on Graphics Recognition*, Springer (2013), 135–146.
9. Dosch, P., and Masini, G. Reconstruction of the 3d structure of a building from the 2d drawings of its floors. In *Document Analysis and Recognition*, IEEE (1999), 487–490.

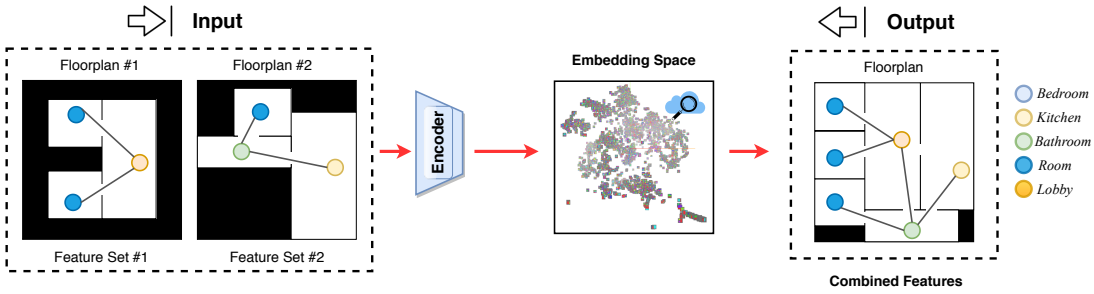
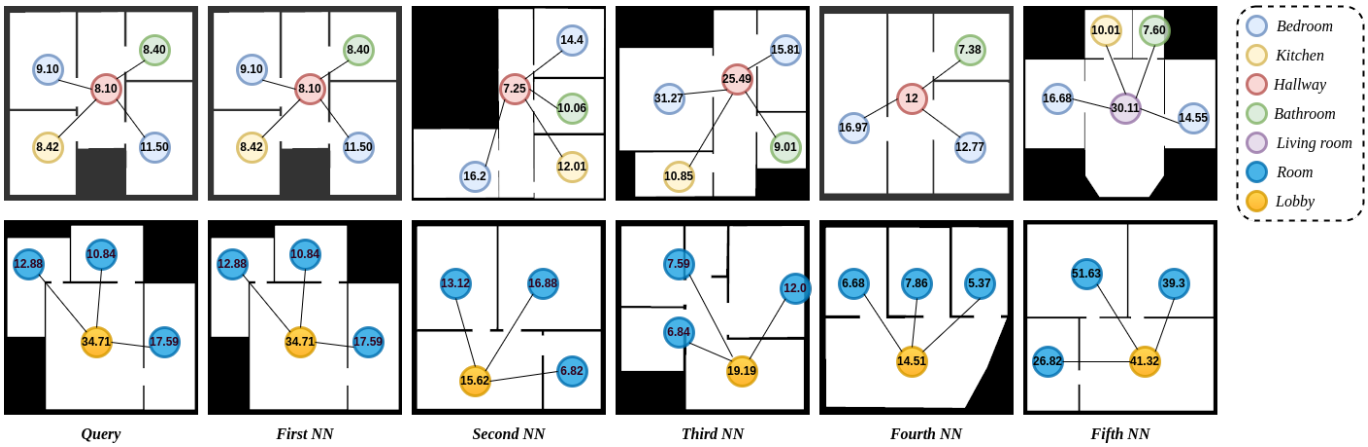
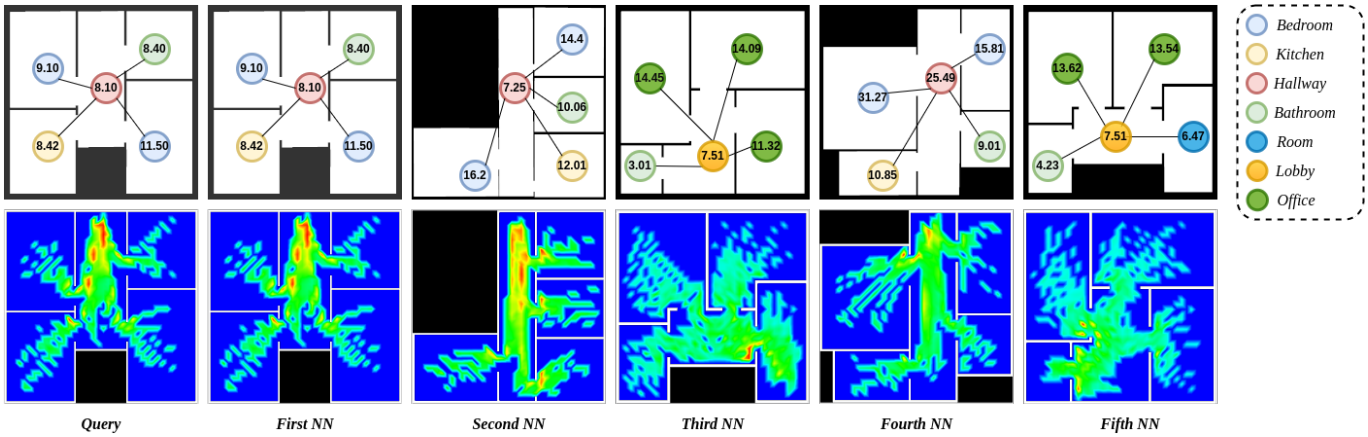



**Figure 4**: Generation of a single floorplan with combined features from an embedding space given two floorplans with different set of features as input. Nodes of underline graph for each floorplan are color-coded based on room types.

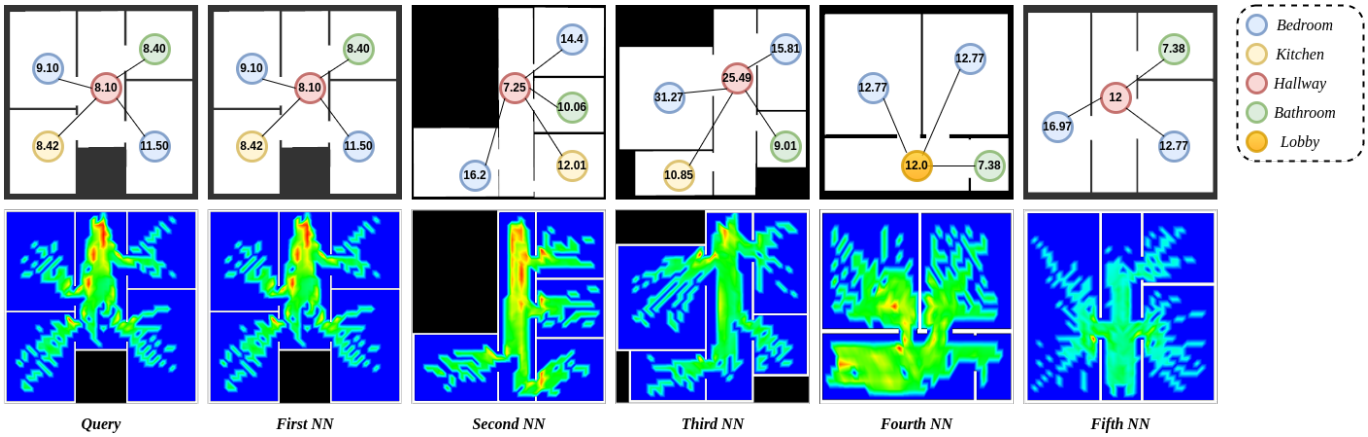

10. Dutta, A., Llados, J., and Pal, U. Symbol spotting in line drawings through graph paths hashing. In *Document Analysis and Recognition*, IEEE (2011), 982–986.
11. Goyal, P., and Ferrara, E. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems 151* (Jul 2018), 78–94.
12. Grover, A., and Leskovec, J. node2vec: Scalable feature learning for networks. In *Knowledge discovery and data mining*, ACM (2016), 855–864.
13. Gupta, N., Das, S., and Chakraborti, S. Extracting information from a query image, for content based image retrieval. In *Advances in Pattern Recognition*, IEEE (2015), 1–6.
14. Heylighen, A., and Neuckermans, H. A case base of case-based design tools for architecture. *Computer-Aided Design 33*, 14 (2001), 1111–1122.
15. Hinton, G. E., and Zemel, R. S. Autoencoders, minimum description length and helmholtz free energy. In *Advances in neural information processing systems* (1994), 3–10.
16. Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.
17. Lambert, G., and Gao, H. Line moments and invariants for real time processing of vectorized contour data. In *International Conference on Image Analysis and Processing*, Springer (1995), 347–352.
18. Lazebnik, S., Schmid, C., and Ponce, J. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition*, vol. 2, IEEE (2006), 2169–2178.
19. Li, T., Ho, D., Li, C., Zhu, D., Wang, C., and Meng, M. Q. H. Houseexpo: A large-scale 2d indoor layout dataset for learning-based algorithms on mobile robots, 2019.
20. Macé, S., Locteau, H., Valveny, E., and Tabbone, S. A system to detect rooms in architectural floor plan images. In *Workshop on Document Analysis Systems*, ACM (2010), 167–174.
21. Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., and Jaiswal, S. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005* (2017).
22. Richter, K., Heylighen, A., and Donath, D. Looking back to the future. an updated case base of case-based design tools for architecture (01 2007).
23. Sabri, Q. U., Bayer, J., Ayzenshtadt, V., Bukhari, S. S., Althoff, K.-D., and Dengel, A. Semantic pattern-based retrieval of architectural floor plans with case-based and graph-based searching techniques and their evaluation and visualization. In *ICPRAM* (2017), 50–60.
24. Sharma, D., and Chattopadhyay, C. High-level feature aggregation for fine-grained architectural floor plan retrieval. *IET Computer Vision 12*, 5 (2018), 702–709.
25. Sharma, D., Chattopadhyay, C., and Harit, G. A unified framework for semantic matching of architectural floorplans. In *Pattern Recognition*, IEEE (2016), 2422–2427.
26. Sharma, D., Gupta, N., Chattopadhyay, C., and Mehta, S. Daniel: A deep architecture for automatic analysis and retrieval of building floor plans. In *Document Analysis and Recognition*, vol. 1, IEEE (2017), 420–425.
27. Singh, S., Kapadia, M., Faloutsos, P., and Reinman, G. An open framework for developing, evaluating, and sharing steering algorithms. In *Motion in Games*, Springer-Verlag (2009), 158–169.
28. Taheri, A., Gimpel, K., and Berger-Wolf, T. Learning graph representations with recurrent neural network autoencoders. *KDD Deep Learning Day* (2018).
29. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. Line: Large-scale information network embedding. In *World wide web*, International World Wide Web Conferences Steering Committee (2015), 1067–1077.
30. Weber, M., Liwicki, M., and Dengel, A. A. scatch-a sketch-based retrieval for architectural floor plans. In *Frontiers in Handwriting Recognition*, IEEE (2010), 289–294.
31. Wessel, R., Blümel, I., and Klein, R. The room connectivity graph: Shape retrieval in the architectural domain.




(a) Floorplans retrieval from the embedding w.r.t. design semantic features alone. Top 5 nearest neighbours of two queried floorplans are shown.




(b) Floorplans retrieval from the embedding w.r.t. behavioral features alone. Top 5 nearest neighbours of for queried floorplan is shown. The ranking of the neighbours is computed based on differences in their behavioral attributes. Time-based behavioral dynamics for human-building interactions are also shown as color-coded heatmaps, where red areas highlight over crowded regions in space.




(c) Floorplans retrieval from the embedding w.r.t. combined design semantics and behavioral features. Top 5 nearest neighbours for the queried floorplan is shown. Time-based behavioral dynamics for human-building interactions are also shown as color-coded heatmaps, where red areas highlight over crowded regions in space.

**Figure 5**: Retrieval of 5 nearest floorplans for the given design layout from embedding space for design semantic only, behavioral only and combined set of features. We used same input design layout in (a), (b) and (c) to demonstrate how the embedding retrievals vary among different categories of features space.