The Visual Computer
https://doi.org/10.1007/s00371-021-02155-w

ORIGINAL ARTICLE

Graph-based generative representation learning of semantically and behaviorally augmented floorplans

Vahid Azizi[1] · Muhammad Usman[2] · Honglu Zhou[1] · Petros Faloutsos[2,3] · Mubbasir Kapadia[1]

Accepted: 27 April 2021

Abstract
Floorplans are commonly used to represent the layout of buildings. Research works toward computational techniques that facilitate the design process, such as automated analysis and optimization, often using simple floorplan representations that ignore the space's semantics and do not consider usage-related analytics. We present a floorplan embedding technique that uses an attributed graph to model the floorplans' geometric information, design semantics, and behavioral features as the node and edge attributes. A long short-term memory (LSTM) variational autoencoder (VAE) architecture is proposed and trained to embed attributed graphs as vectors in a continuous space. A user study is conducted to evaluate the coupling of similar floorplans retrieved from the embedding space for a given input (e.g., design layout). The qualitative, quantitative, and user study evaluations show that our embedding framework produces meaningful and accurate vector representations for floorplans. Besides, our proposed model is generative. We studied and showcased its effectiveness for generating new floorplans. We also release the dataset that we have constructed. We include the design semantic attributes and simulation-generated human behavioral features for each floorplan in the dataset for further study in the community.

Keywords Floorplan representation · Floorplan generation · LSTM Variational autoencoder · Attributed graph · Design semantic features · Human behavioral features

1 Introduction

Floorplan representations support a set of fundamental activities in the architectural design process, such as the ideation and development of new designs, their analysis and evaluation for any selected performance criteria, and the communication among the stakeholders. While computer-aided design (CAD) and building information modeling (BIM) approaches to support the creation of digital building models from which floorplans can be extracted, these methods do not support the systematic representation or comparison of floorplan features, which could be derived from geometric and semantic properties, as well as more advanced performance metrics, such as space utilization and occupant behaviors [13,31]. Image processing techniques and convolutional neural networks (CNNs) have been utilized to extract features from floorplan images [42,43]. These features are used for retrieving similar floorplans. In another branch, the floorplans are represented with graphs, and graph matching methods are utilized for comparing and retrieving similar floorplans [40,41]. Other approaches like symbol spotting methods [11,19,37] are utilized for retrieving similar floorplans. However, none of these works represent floorplans with numerical vectors. Besides, they overlooked the design semantic and human behavioral features. Design semantic features are necessary because they have high-level information for floorplans like the square footage of rooms. Humans are the primary inhabitant of these buildings, and

✉ Vahid Azizi
vahid.azizi@cs.rutgers.edu

Muhammad Usman
usman@cse.yorku.ca

Honglu Zhou
hz289@cs.rutgers.edu

Petros Faloutsos
pfal@cse.yorku.ca

Mubbasir Kapadia
mk1353@cs.rutgers.edu

[1] Computer Science Department, Rutgers University, Piscataway, NJ, USA

[2] Electrical Engineering and Computer Science Department, York University, Toronto, Canada

[3] UHN Toronto Rehabilitation Institute, Toronto, Canada

Published online: 24 May 2021

their behaviors (i.e., trajectories) are often highly correlated with environments [47]. Their interaction with the environment provides necessary information for safety or other types of design metrics like visibility and accessibility. Evacuation time, travel distances, and crowd flow (also known as egress flow) are some of the salient behavioral features which have been widely used by researchers in analyzing and optimizing architectural elements and environments using crowd simulations as they relate to human movements [6,16,17].

We propose a novel technique for floorplan representation that models floorplans with attributed graphs as an intermediate representation to address the limitations in previous works. A novel long short-term memory (LSTM) variational autoencoder (VAE) model is proposed to embed the attributed graphs in a continuous space. This method considers the design semantics and high-level structural characteristics, and crowd behavioral attributes of potential human–building interactions. This approach represents floorplans with numerical vectors in which the geometrical properties, design semantics, and human behavioral features are encoded. These vectors facilitate different applications related to floorplans, such as recommendation systems, real-time evaluation of designs, fast retrieval of similar floorplans, and any application that needs to cluster floorplans. The qualitative and quantitative results show the performance of our model for generating representative embedding vectors such that the considered features are encoded accurately. A user study is conducted to validate floorplan retrievals from embedding spaces to their similarity with the input floorplans. Floorplan generation is an active area of research in computer graphics. Recently floorplan generation methods based on machine learning have been integrated into design workflows to facilitate and enhance the design process, [9,21,52]. Although floorplan generation is not our approach's primary goal, the proposed model is generative. We can automatically generate floorplans with desired characteristics, as demonstrated by our experiments.

1.1 Contributions

Our contributions can be summarized as follow:

1. A workflow to represent floorplans as attributed graphs, augmented with design semantic and crowd behavioral features generated by running crowd simulations.
2. A novel unsupervised generative model to learn a meaningful vector representation of floorplans using LSTM variational autoencoder.
3. Generation of new floorplans using the proposed model.
4. A user study to evaluate the qualitative performance of our approach.
5. Provision of a publicly released dataset of floorplans of indoor environments, which are augmented with semantic and behavioral features. The semantic design features are extracted by our automated tool, and the human behavioral features are generated by hours of the running simulation.

2 Related work

Our work relates to research in two areas: floorplan representation and floorplan generation.

2.1 Floorplan representation

Floorplan representation aims to represent floorplans with numerical vectors whose structure and features are encoded in these vectors. To the best of our knowledge, representing floorplans by numerical vectors is not done to date. There are some prior works for retrieving similar floorplans with representing floorplans as images or graphs. They can be mainly divided into three categories: image-based, graph-based, and symbol spotting methods.

2.1.1 Image-based methods

Several approaches based on conventional image processing techniques for comparing floorplans are proposed. In these approaches, floorplans are represented as images and histogram of oriented gradients (HOG) [10], bag of features (BOF) [27], local binary pattern (LBP) [1], and run length histogram [18] have been utilized for extracting features from these images. Then, these extracted features are used for comparison and retrieving floorplans. By emerging convolutional neural networks (CNN), In [43], a deep CNN is presented for feature extraction to address the limitation of conventional image processing techniques for extracting features. These methods suits object-centric floorplans datasets in which floorplans are annotated with furniture or specific visual symbols. However, these features are not semantics and do not correctly capture the high-level design features. Moreover, human behavioral features are not considered in these methods.

2.1.2 Graph-based methods

In this category, floorplans are represented with graphs, and graph matching methods are utilized for measuring their similarity. Different strategies are used for representing floorplans as a graph. In [42] rooms are nodes, and edges capture the adjacency between the rooms. Besides, nodes are augmented with furniture types annotated in floorplans. In [41], the graphs are augmented with more attributes like room area and furniture style in three different representation layers. Since in these works, floorplans are represented with graphs,

all of them capture the floorplans structure, and some of them add attributes to nodes. However, mainly they do not include semantic and high-level features as well as human behavioral features. Moreover, all of these methods use graphs as a final representation and do not provide numerical vectors.

2.1.3 Symbol spotting methods

Symbol spotting is a special case of content-based image retrieval (CBIR) [19,37] which is used for document analysis. The system retrieves zones from the documents that are likely to contain the giving query. Queries could be a cropped or hand-sketched images. Pattern recognition techniques are used in symbol spotting methods like moment invariants such as Zernike moments in [26]. Reducing search space in symbols spotting methods is proposed based on hashing of shape descriptors of graph paths (Hamiltonian path) in [11]. SIFT/SURF [51] features being 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.

2.2 Floorplan generation

Floorplan generation aims to generate floorplan designs automatically by satisfying some constraints like room sizes and adjacency between rooms. We can divide them into two groups: procedural/optimization-based methods and recently deep learning methods.

2.2.1 Procedural/optimization-based methods

The constraints are manually defined in these methods, and optimization methods are used for constraint satisfaction to generate new floorplans. In [31], they used the Bayesian network to learn synthesizing floorplans with given high-level requirements. In [38,39], they proposed an enhanced evolutionary strategy (ES) with a stochastic hill climbing (SHC) technique for floorplan generation.

2.2.2 Deep learning methods

In [52], a deep network was proposed for converting a given floorplan layout as input to a floorplan with predicting rooms and walls location. In [9], they proposed a method comprising three deep network models to generate floorplans. In the first step, the model generates the layout. In the second step, the room locations are generated, and finally, furniture locations will be generated. In this model, users are in the loop, and they can modify the input for the next steps. In [21], they proposed a framework based on deep generative network. Users specify some properties like room count, and their model converts a layout graph, along with a building boundary, into a floorplan. A graph-constrained generative adversarial network is proposed in [34]. They took an architectural constraint as a graph (i.e., the number and types) and produced a set of axis-aligned bounding boxes of rooms.

2.3 Comparison to prior work

Previous works do not represent floorplans with numerical vectors. They usually overlooked the design semantic and human behavioral features. This paper presents floorplans with numerical vectors encoded with design semantic, human behavioral features, and room directions. Besides, we also utilize generative models for addressing floorplan generation at the same time. The proposed method is an extension of a recent approach [2] with the following extensions:

- A novel LSTM variational autoencoder with two branches for representing attributed graphs with features on both nodes and edges with numerical vectors. Besides, we also include rooms' directions as edge attributes to maintain layout symmetry.
- We study and showcase the generative power of our model for generating new floorplans.
- We conducted a user study to evaluate the qualitative performance of our embedding approach.

3 The proposed framework

Figure 1 illustrates the proposed framework, comprising of two components. The first component models floorplans with attributed graphs that nodes and edges of these graphs are augmented with design semantic and human behavioral features. The second component embeds attributed graphs in a continuous space. The details are provided in the following sections.

3.1 Dataset

We used the HouseExpo dataset [28] that includes 35, 126 2D floorplan layouts in JavaScript Object Notation (JSON) format. There are 25 room types in this dataset where some of them share similar semantic labels (e.g., toilet and bathroom or terrace and balcony). We reduced the types to 10. This reduction is made by removing less common types based on reported statistical metrics in the dataset (e.g., freight elevator) and considering a unique type for similar proposed components. The final room types are Bedroom, Bathroom, Office, Garage, Dining Room, Living Room, Kitchen, Hall, Hallway and Unknown. Unknown type is considered for room segments with a noisy label that we cannot assign a unique label. Additionally, we remove from the set the floor-

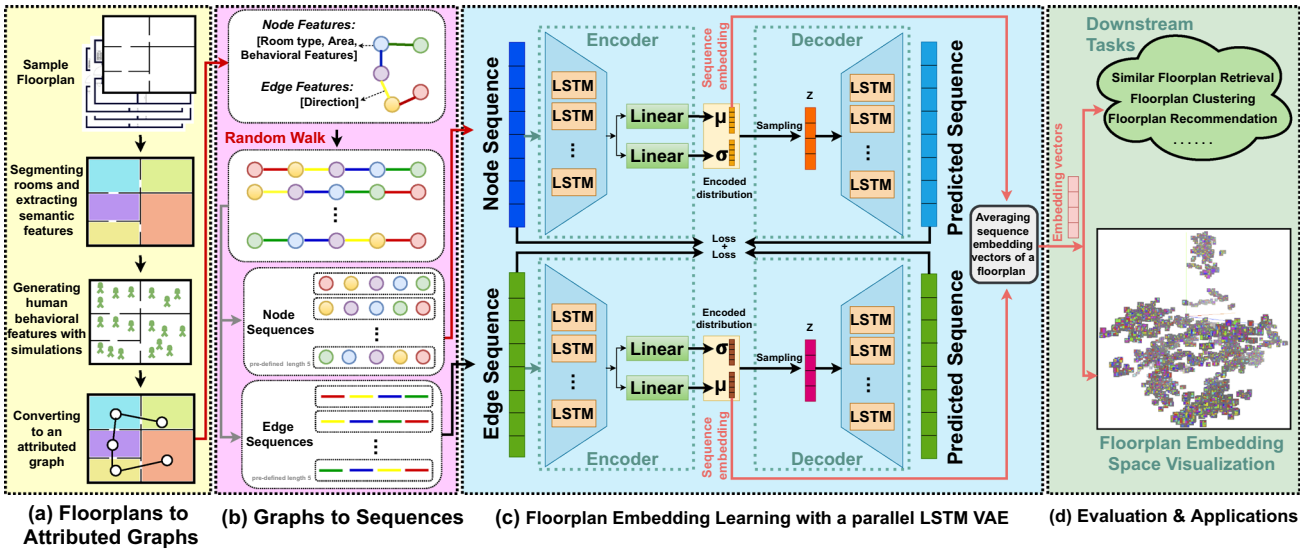

Fig. 1 An overview of the proposed approach. **a** Floorplans are firstly converted to *attributed graphs* as immediate representation, with attributes residing on both nodes and edges; **b** *random walk* is applied to the attributed graphs to generate a set of node sequences and edge sequences; **c** embedding vectors are learned with a novel parallel LSTM VAE model; **d** the learned embedding vectors can be used for different downstream tasks

plans with inaccurate or missing labels. At the end of this process, we obtain 8729 floorplan layouts. This preprocessing is done to make the dataset reliable for training. Corrupted data will decrease the training accuracy and makes the outcomes undesirable.

3.2 Floorplans to attributed graphs

After pruning the dataset, we model each floorplan with an attributed graph. The rooms compose nodes, and the edges are their connectivity if there is an immediate door between the room pairs. For this conversion, we compiled HouseExpo samples as images. Then we utilized a series of image processing techniques for room segmentation and finding their connectivity. Graph structure resembles the structure of floorplans like the number of rooms and their connectivity. Considering floorplan structure is necessary but not enough. To have a better and more meaningful representation, we need to integrate high-level design semantic features. Moreover, humans inhabit these buildings, and their interaction with the environment provides valuable implicit information. Integration of how they interact with environments is necessary for safety or other types of design metrics like visibility and accessibility. Therefore, we augmented the graphs with both high-level design semantic features and human behavioral features (Table 1).

The semantic design features include room type, square footage, and the connection direction. The room types represented with a 10-dimensional one-hot vector where $roomType_i = 1$ if the type is ith type and other entities are zero. The square footage is represented with a scalar value and direction of connection with a four-dimensional one-hot vector. We considered four main directions: North, East, South, and West. Thus, $direction_i = 1$ if direction belongs to ith direction and other entities are zero. The room types are provided as a label in the dataset, and image processing techniques extract both square footage and direction of the connection. We are not given the cardinal directions. Therefore, we considered the top left corner of floorplan images as the origin. Hence, the $+y$ axis points to the north, and other directions are considered relatively. For calculating the direction, we need a reference point (i.e., room). The direction between rooms is the direction from the node (i.e., room) with the highest degree (room with more connections) to the node with a low degree. Usually, the node (i.e., room) with the highest degree is the main room in the floorplans like the living room or hallway. In other words, the node with the highest degree is the reference for setting the directions. Please note in the graph that the edges are bidirectional if there is a door between two rooms, and connection direction is the assigned feature. The room type and square footage are specific to each room, and we consider them as node features. However, the connection direction is a shared property between room pairs; we add it to the edge features (edge between room pairs).

The human behavioral features are generated by running simulations (Fig. 2). They include metrics regarding evacuation time, traveled distance, flow rate, and the number of successful/unsuccessful agents to exit from the corresponding building (Table 1). To generate these behavioral features,

Table 1 Features on nodes and edges

	Feature classes	Feature types	Dimension
Node features	Design semantic	Square footage room types	1
			10
	Behavioral	Not completed agents	1
		Max evacuation time	1
		Min evacuation time	1
		Exit flow rate	1
		Completed agents	1
		Max traveled distance	1
		Avg evacuation time	1
		Avg traveled distance	1
		Min traveled distance	1
Edge features	Design semantic	Direction	4

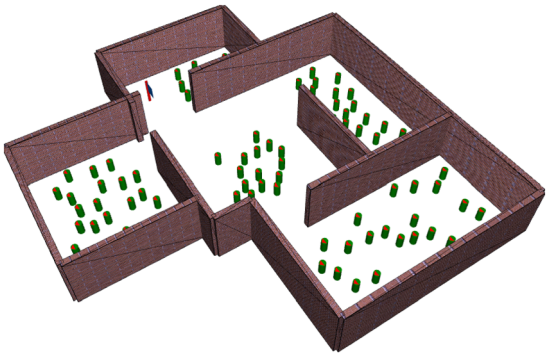

Fig. 2 Crowd simulations are used to compute behavioral features for the floorplans. Layout walls are shown in brown, crowds are shown in green, and the blue flag shows the building exit point (for evacuation)

we converted 2D floorplans to 3D models loadable in a crowd simulator, SteerSuite [46]. The simulator automatically populates virtual agents in each room with the target to exit the floorplan. In one of the training models (e.g., Model 3) presented in Sect. 4.2, we incorporated these behavioral features (retrieved from simulations–dynamic) along with design semantic and edge features (retrieved from the geometric/topological characteristics of the environment–static) to evaluate how effectively they impact the embedding space. For example, we can expect to retrieve neighbors (floorplans) from Model 3, which do not look symmetrical in terms of holding their overall shape but are behaviorally similar and would yield similar values for behavioral features, and also if users are able to correctly perceive the ordering of the retrieved neighbors which are retrieved using Model 3 than other embedding models that are trained using design semantic alone or design semantic plus edge features.

Note that in our simulations, the only obstacles the agents interact with are the walls of the environment (static obstacle) or the other agents (dynamic obstacle). However, our simulation setup does not restrict us from including different kinds of obstacles in the environment (e.g., pillars or other physical objects). The past research has shown that the placement of pillars or other obstacles at proper locations can often facilitate movements (e.g., crowd flow) during the evacuation of the environment [6,16,17]. Figure 3 shows a snapshot of the simulation in the presence of 4 obstacles in the environment. All human behavioral features are presented with one scalar value with a total dimension of 9. These features are generated for each room; hence we added them to the nodes feature vector.

At the end of this step, each floorplan is represented with an attributed graph $G = (V, E)$ in which V denotes its vertex set (room segments) and $E \subseteq V \times V$ denotes its edges (connectivity between room pairs). Each node v has a 20-dimensional feature vector F_v and each edge e has a four-dimensional feature vector F_e.

3.3 Floorplan embedding

We model the floorplans with attributed graphs with features on both nodes and edges as described in Sect. 3.2. These graphs represent floorplan geometry, their design semantics, and behavioral features. They can be used directly for floorplan representation. However, graph analysis is expensive in terms of computation and space cost. This challenge is addressed by proposing efficient graph analysis methods like [14,25,29] but are not efficient enough. Besides, These methods do not represent graphs with a compact numerical vector. Another solution for addressing the complexity of graph analysis is graph embedding. Graph embedding maps the graphs to a low-dimensional space in which their properties and information are maximally preserved. In this low-dimensional space, the graphs with similar properties are close. We have different graphs like the heterogeneous graph, homogeneous graph, attributed graph. It means the input for graph embedding methods are varying, and a single method cannot handle all types. Graph embedding can mainly be divided into node embedding, edge embedding, hybrid embedding, and whole-graph embedding [8].

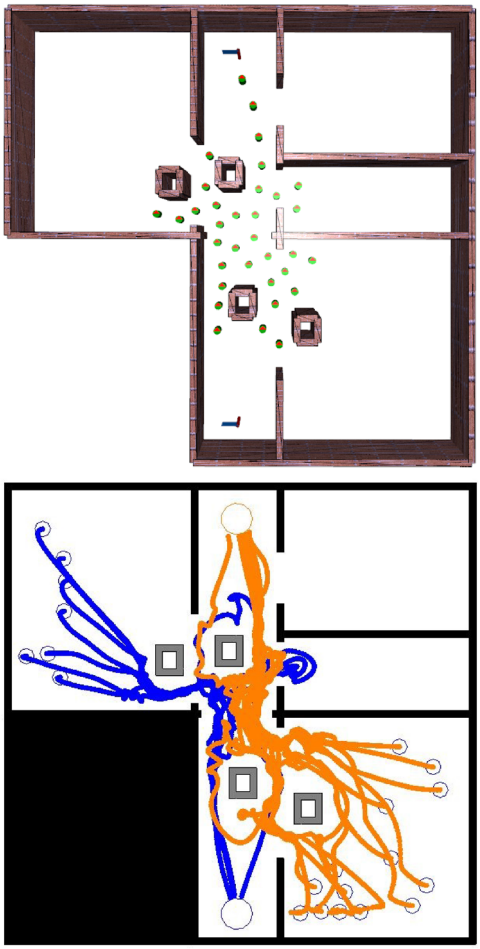

Fig. 3 A snapshot of the simulation in the presence of the obstacles (e.g., pillars/hurdles). Top: the crowd simulation in 3D. Bottom: Crowd trajectories overlayed on top of the environment layout

In this paper, we are dealing with whole-attributed-graph embedding. We want to represent each attributed graph with a vector. These vectors encode graph (floorplan) structure as well as their design semantics and behavioral features. Besides, these graphs are unlabeled, i.e., we do not have a label for each graph to perform supervised classification or regression. Moreover, the graphs vary (in terms of the number of nodes), and the number of nodes is relatively small. We can use other types of embedding like node embedding and then use the node embedding vectors' average as the whole-graph representation. However, with this strategy, the whole-graph structure is not appropriately captured and does not lead to accurate vector representation [4,48].

There are quite a few works for the whole-graph embedding. Some whole-graph embedding methods rely on the efficient calculation of graph similarities in graph classification tasks [4,32,35,44]. These methods are supervised and need a labeled dataset. Besides, they are designed for unattributed graphs. On the contrary, our graphs are attributed and unlabeled. Therefore, these methods do not apply to our problem, and we need an unsupervised method. Graph2Vec [33] is an unsupervised method that maximizes the likelihood of graph subtrees given graph embedding and generates vector representations. However, since this model uses subgraphs, the global graph information is not captured correctly [48]. Besides, this method is not applicable for graphs with attributes both on nodes and edges. In [48], for capturing the whole-graph structures, they take advantage of random walk for converting graphs to a set of sequences. Then an LSTM autoencoder is presented to learn graph representations. However, this method suits unattributed graphs.

Sentences are presented with a sequence of words. Each word in sentences is represented with an embedding vector. In other words, we have a sequence of vectors in sentences. In [7,50], LSTM variational autoencoder is used for text and sentence embedding and generation. The methods proposed in [7,48,50] motivate us to convert our graphs to sequences (like sentences which are a sequence of vectors) and propose a novel LSTM variational autoencoder model that suits our unlabeled attributed graphs with feature both on nodes and edges. In particular, we convert each floorplan (graph) into a set of sequences (Sect. 3.5), and we propose a generative model that maps our graphs (sequences) to a d-dimensional space $\theta : G \rightarrow R^d$. The proposed model is detailed in the next section.

3.4 Model

We present a novel LSTM variational autoencoder architecture illustrated in Fig. 1. LSTM is a special kind of recurrent neural network (RNN). It is designed for learning long-term dependencies by introducing state cell [20] to address the vanishing gradient in vanilla RNN with long-term sequences [5]. Autoencoders are a type of unsupervised neural network with two connected networks. The first network is an encoder that converts the inputs to latent vectors in a low-dimensional space. The second network is the decoder, which reconstructs the original input vector from latent vectors [24]. However, the vanilla autoencoders map each input to a constant vector. The embedding space with vanilla autoencoder is not continuous, and interpolation is not allowed. variational autoencoder (VAE) is a generative model designed to address vanilla autoencoders' limitations by learning the probability density function (PDF) of the training data [23]. The VAE generates a continuous embedding space in which vector operations are allowed.

As mentioned we have attributes both on nodes and edges with different dimensions. To address this difference in dimension, we consider a parallel LSTM VAE with two branches, one for node sequences and one for corresponding edge sequences. Let $S_n = s_{n_1}, s_{n_2}, ..., s_{n_i}$ be a node sequence and $S_e = s_{e_1}, s_{e_2}, ..., s_{e_{i-1}}$ be the corresponding

edge sequence. We aim to learn an encoder and decoder to map between the space of these two sequences and their continuous embedding $z \in R^d$ where d is a hyperparameter. In each branch, the encoder is defined by a variational posterior $q_\phi(z|S_n)$ and $q_\phi(z|S_e)$ and the decoder by a generative distribution $p_\theta(S_n|z)$ and $p_\theta(S_e|z)$, where θ and ϕ are learned parameters. For each branch, loss function has two terms (Eq. 1). The first term is reconstruction error that we used Mean Square Error (MSE). This term encourages the decoder to learn to reconstruct the data $\bar{S}_n, \bar{S}_e$. The second term which is a regularizer is Kullback–Leibler (KL) divergence to penalize loss if the encoder outputs representations that are different than a standard normal distribution $N(0, 1)$ [45]. In training, two branch loss functions are summed for back-propagation (Eq. 1).

$$\begin{aligned} Loss_{total} = &(||S_n - \bar{S}_n||^2 + KL[q_\phi(z|S_n), N(0, 1)]) \\ &+ (||S_e - \bar{S}_e||^2 + KL[q_\phi(z|S_e), N(0, 1)]) \end{aligned} \quad (1)$$

In both branches for the encoder, we have an LSTM layer with 256 units. In the following, we have two fully connected layers with dimension 16 for generating μ and σ and consequently the $d = 16$. Then we have the sampling, and finally, we have the decoder with one LSTM layer with 256 units for reconstructing the input sequences. The number of layers and number of units/neurons are set experimentally for best performance. Adam [22] is used as optimizer. Before training, each graph is converted to a set of sequences (Sect. 3.5), and these sequences are used as input for training. After training, each graph is represented by averaging its sequences' embedding vectors, Eq. 2 [48]. The $Nseq$ is the number of sequences, and RS is the corresponding latent vector to each sequence.

$$\Phi(G) = \frac{1}{Nseq} \sum_{n=1}^{Nseq} RS_n \quad (2)$$

3.5 Graphs to sequences

Graphs can be converted to sequences by methods including but not limited to random walk or breadth-first search (BFS). In [48], the random walk, BFS, and shortest path between all pairs of nodes are utilized. The experiments show that sequences generated by random walk lead to a better vector representation. The reason is that random walk captures more than immediate neighbors of nodes. The random walk is introduced in [36] for converting graphs to sequences. In this version, we pick a node, and then we choose one of its edges randomly to move to the next node. We repeat this procedure until we get a walk of some predefined length. (Length of a walk is defined by the number of nodes on the walk, and a shorter walk than the predefined length will be padded into the predefined length.) Later, two other versions are proposed.

Random Walk 1 In [15], the random walk is modified to have two parameters Q and P. Parameter Q is the probability of discovering the graph's undiscovered parts, and parameter P is the probability of returning to the previous node. We call the random walk proposed in [15] "random walk 1."

Random Walk 2 In [48] the random walk is modified by adding probability $1/D(N)$, where $D(N)$ is the degree of node N. We start from a node in this walk, and the next node will be selected by its probability $1/D(N)$. We call this random walk presented in [48] as "random walk 2."

We used both walks with different walk lengths to find the best performer walk and walk length. Besides node sequences, the edge sequences are captured at the same time. Both nodes and edge sequences are used for training the model.

4 Experiments

4.1 HouseExpo++ dataset

The original HouseExpo dataset includes 35, 126 2D floorplans. For each floorplan, the number of rooms, the bounding box of the whole floorplan, a list of vertices, a dictionary of room categories, as well as their bounding boxes are provided [28]. While we can use the provided bounding box of the rooms for segmentation, these bounding boxes are not accurate; therefore, we use them only for labeling. We compile these floorplans (JSON format) into images. Then, we segment the images to find the rooms, their possible connections, the direction of connections, and their square footage. The provided bounding boxes in the original dataset are used for assigning labels to room segments by the criterion of maximum overlapping. The described processes are done with our automated tool by image processing techniques. Moreover, we convert these JSON formatted floorplans to files in a readable format with our 3D crowd simulator (SteerSuite). Then by running simulations, we record the human behavior features (features are provided in Table 1). We call this augmented dataset as *HouseExpo++* that is publicly available in [3].

4.2 Training

As described in Sect. 3.5, we converted graphs to nodes and edge sequences. We utilized both mentioned walks with walk lengths 3, 5, and 7. For random walk 1, we set both Q and P to 0.5. For each graph, we run the random walk 11 times. Therefore, we have 11 sets of node and edge sequences for each graph. Out of 11, one set is considered as a proxy set (proxy graph). These proxy sets do not participate in training and are

only used later for evaluation. As mentioned, we have two random walks, and we run each of them with walk lengths 3, 5, and 7. In total, we have 6 different sets of sequences. On average, we have 5 nodes in each graph. However, because of randomness in random walks, all sequences are not valid. For example, it happens to have a sequence which is the repetition of two nodes. These types of irregularities are pruned. On average, we have 306440 node sequences and 306440 corresponding edge sequences in these 6 sets. The sequences with lengths less than the target length are padded with zeros. We trained three models considering a different set of features.

Model 1 In this model, we only considered the design semantic features on nodes. We removed the edge branch, and the model is trained only with nodes sequences. The dimension of node features is 11.

Model 2 In this model, we considered semantic design features both on nodes and edges. The model is trained with both branches. The features dimension on nodes is 11 and on edges is 4.

Model 3 In this model, we considered all semantic design features and human behavioral features. The model is trained with two branches. The features dimension on nodes is 20 and on edges is 4.

All three models have the same described architecture and loss function. Note that in Model 1, the edge branch is removed, and we have only the node branch's loss function. However, in the other two models, the loss is the summation of two branches' loss. The learning rate was set empirically as 0.001. All three models are trained on a machine with 32 GB RAM, 12 * 3.50 GHz cores CPU, and Quadro K620 GPU with 2 GB Memory. On average, each model takes about 4 hours for training with 50 epochs.

4.3 Quantitative evaluations

This section presents quantitative results from the experiments.

4.3.1 Nearest neighbor ranks

As mentioned, by embedding, the graphs are mapped to a continuous embedding space. The graphs with similar structures and properties should be close to each other in the embedding space. The closeness of two floorplans can be measured by the euclidean distance of the two corresponding embedding vectors (smaller distance denotes higher similarity). If this embedding space is well constructed, similar floorplans in terms of structure and semantic and behavioral properties should be closed. Therefore, for each graph (called query graph), we compute the euclidean distance from this graph to other graphs (including itself) and rank the other graphs for this graph according to the distance. We hypothesize that each graph should find itself as the first nearest neighbor and its proxy graph (a different set of sequences for the query graph) in close ranks. This study is independent of considered features on graphs and only shows the model's effectiveness for generating valid embedding vectors. Therefore, we use all three models to obtain the top 5 nearest neighbors for each floorplan in their corresponding learned embedding space. We calculate the average percentage of graphs that have themselves in the first rank and the percentage of proxy graphs in the other four ranks. Table 2 shows these average percentages with different walk lengths for both random walks.

As Table 2 shows, in both random walks, walk length 5 leads to better performance. Besides, random walk 2 is superior. By random walk 2 and walk length 5, each graph is in the first rank and 92% of proxy graphs in the second rank. Since proxy graphs are a different set of sequences on the graphs, a good percent of the proxy graphs should be present in top ranks if the model performs appropriately. Random walk 2 captures our graph's structure better, and consequently, it has better performance. In graphs (i.e., floorplans), we always have the central node (i.e., room) with a high degree. Then moving toward this node gives the sequences that capture our graph structure better. The walk length has a dependency on the size of the available graphs in the dataset. For us, walk length 5 is the suitable length since, in both random walks, the embedding performance is better compared to walk length 3 and 5. In [2], with vanilla LSTM Autoencoder, on average 84% of proxy graphs are in the second rank. However, this new model improves this percentage to 92%.

4.3.2 Clustering

As mentioned in the previous section, floorplans with similar properties are close in the embedding space. This similarity is in terms of floorplans structure, design semantics, and human behavioral features. There are many parametric methods for clustering like KMeans [49] that we need to give the number of clusters as the input parameter. Since we do not want to limit ourselves to a specified number of clusters, we used density-based spatial clustering of applications with noise (DBSCAN). It is a nonparametric clustering method based on density. Each dense region (close-packed points) represents a cluster, and the points in the low-density areas are marked as outliers. It has two parameters, the minimum number of points in each cluster, and the maximum distance between two samples in each cluster [12].

We sampled 1000 floorplans, and we run DBSCAN over their embedding vectors generated by our three models to cluster them. We calculated the standard deviation for the number of nodes, average node degrees, node types, edge types, and flow rate for each cluster. The average of mentioned metrics on all clusters is provided in Table 3. The number of nodes and average node degrees are almost simi-

Table 2 Average of nearest neighbor ranks (Sect. 4.3.1) with two types of random walks and walk length 3, 5, or 7 for our three models.

	Walk length	Rank of query floorplan within 5 NN	Rank of proxy graph within 5 NN
Random walk 1	3	[100, 0, 0, 0, 0]	[0, 51, 2, 1, 1]
	5	[100, 0, 0, 0, 0]	[0, 76, 4, 2, 2]
	7	[100, 0, 0, 0, 0]	[0, 56, 2, 1, 1]
Random walk 2	3	[100, 0, 0, 0, 0]	[0, 85, 3, 2, 1]
	5	[100, 0, 0, 0, 0]	**[0, 92, 2 , 1, 1]**
	7	[100, 0, 0, 0, 0]	[0, 88, 3, 1, 1]

For each graph, we compute the euclidean distance from this graph to other graphs (including itself and a proxy graph which is a different set of sequences of the query graph) and rank the other graphs for this graph according to the euclidean distance. Each graph should find itself as the first nearest neighbor and its proxy graph in close ranks. We calculate the percentage of graphs that have themselves in the first rank and the percentage of proxy graphs in the other top four ranks (e.g., "[0, 76, 4, 2, 2]" in the table denotes that 76% graphs have their proxy as the top 2 nearest neighbor, 4% as top 3, 2% as top 4, and 2% as top 5). This analysis showcases that walk length 5 can lead to better performance, and random walk 2 is superior

Table 3 Average of standard deviation for number of nodes, node degrees, node types, edge type and flow rate in clusters out of 1000 samples in our three models

	Number of nodes	Average of node degrees	Node types	Edge types	Flow rates
Model 1	0.164	0.092	1.11	2.12	1.71
Model 2	0.168	0.091	1.06	1.34	0.68
Model 3	0.161	0.093	1.07	1.41	0.34

lar in all three models. The reason is that all the corresponding features to these metrics are available in three models. The node type is a common feature in all three models. However, in models 2 and 3, the performance is better. It is because of integrating edge types in these two models. Edge type is not a considered feature in model 1, and we have the worse performance in model 1. Flow rate is only available in model 3, which we have the best performance. However, model 2's performance for the flow rate is satisfactory and shows that integrating edge direction helps find more similar floorplans with similar flow rates. Figure 4 shows the resulting clusters over 1000 samples in model 3. For visualization, the vectors' dimension is reduced to two by t-distributed stochastic neighbor embedding (TSNE) [30], and two sample floorplans from one of the clusters show the graph properties are encoded accurately with our model.

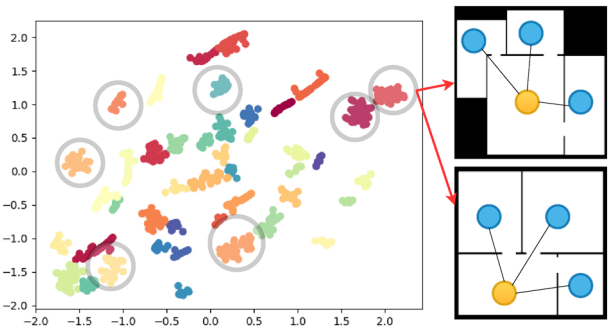

Fig. 4 Clusters after running DBSCAN over 1000 random samples. Two samples from one of the clusters are shown. They have the same number of nodes, same node degrees, and similar room types (blue nodes are bedrooms, and yellow nodes are living rooms). Besides, the average square footage for the top floorplan is 23.49, and for the bottom floorplan is 25.38. This shows that the embedding space indeed captures the design semantics of floorplans

4.4 Qualitative evaluations

This section presents qualitative results from the experiments.

4.4.1 Nearest neighbors (NNs)

As described, we trained three models with a different set of features. Each model makes an embedding space. We selected three random floorplans and found their top 5 nearest neighbors in each model's corresponding embedding space. Figure 5 shows the query floorplans and their top 5 nearest neighbors. For each sample, the first row shows the NNs in the first model's embedding space, the second line shows the NNs in the second model's embedding space, and the third row shows the NNs in the third model's embedding space. As shown in the image, with the first model, the floorplans have the same structure in terms of room numbers, room (node) degrees, and room types. However, the room arrangements are not similar. In the second model, since the edge features are added, the high-rank NNs follow the same arrangement, and with moving toward low-rank NNs, the arrangement similarity is decreased. However, they have a similar structure yet. In the third model, the human behavior features are added, and floorplans with similar behavioral features get close to query floorplans. The last row for each sample shows the visualization of the crowd flow rate. The numbers inside floorplans in the first and second row show the square footage of each room. In the third row, the num-

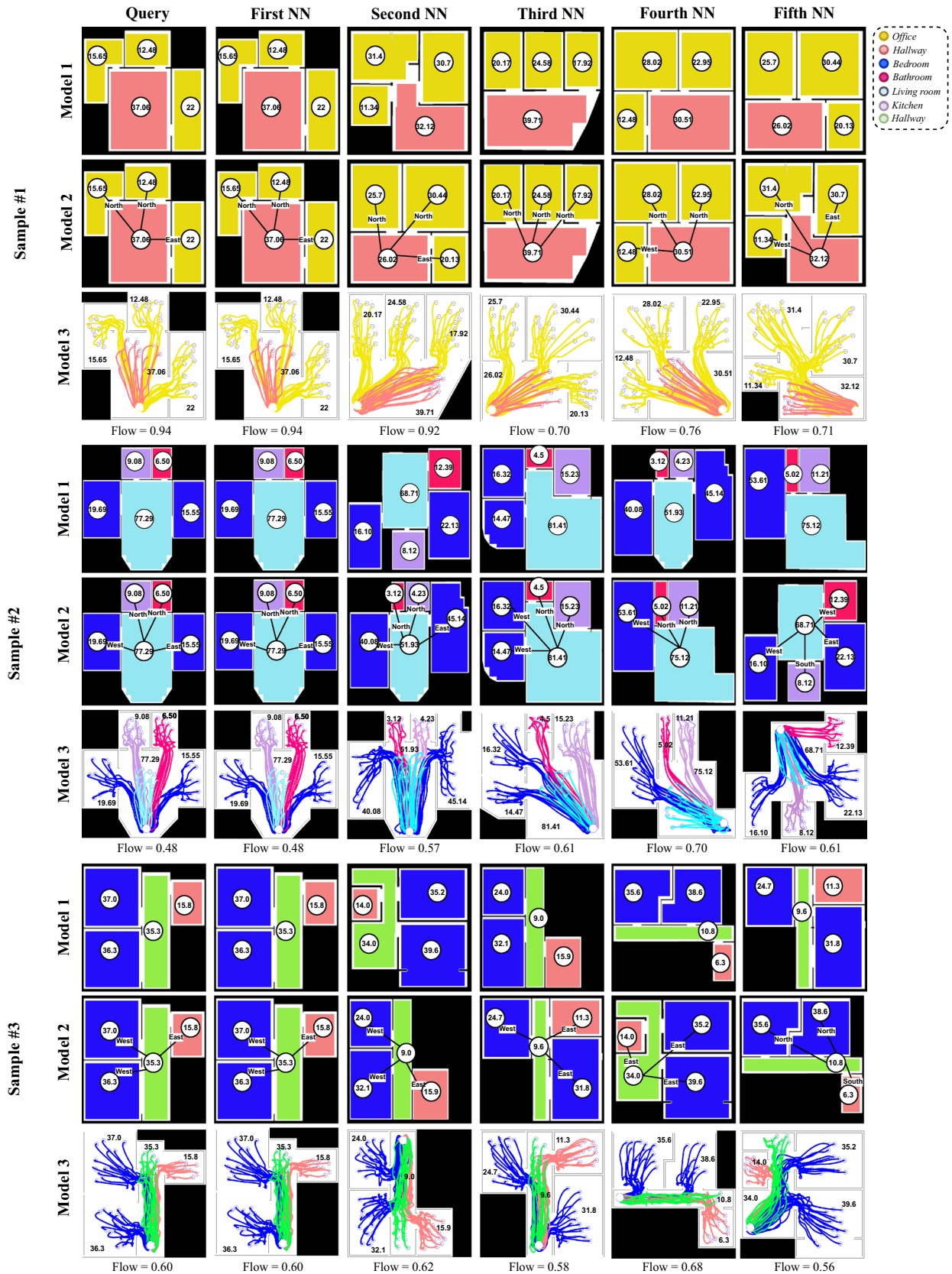

Fig. 5 Top 5 NNs for three floorplans found with three models. The first row shows the NNs with the first model, the second row shows the NNs with the second, and the third row shows the NNs with the third model. The room's color denotes the room type, and the numerical value inside each node denotes the square footage of the room. In the third row of each sample, the crowd flow is visualized, and the numbers depict flow rates. See Sect. 4.4.1 for details

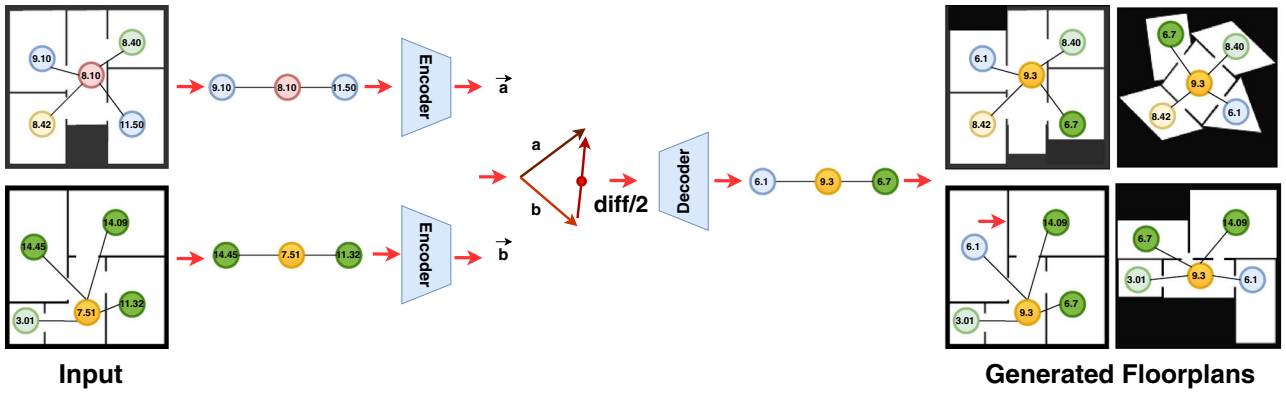

Fig. 6 Floorplan generation with interpolating in embedding space. We select two random sequences from two random floorplans, and after encoding them, we calculate the difference of their embedding vectors. Adding half of their difference to the base vector and decoding it gives us a new sequence. These new sequences can be used for generating new floorplans. See Sect. 5 for details

bers depict flow rates. Please note, as mentioned in Sect. 3.2, the north is at the top, and other directions are recognized correspondingly.

5 Floorplan generation

Given that variational autoencoders are generative, we study the skill of our model for generating new floorplans. Generating new floorplans can be done with sampling from the posterior distribution of sequences or with homotopies [7,50].

5.1 Sampling from posterior distribution

VAE learns the data distribution instead of deterministic mapping. Therefore, we can sample from these posterior distributions for generating new data. As mentioned in Sect. 4.2, for each graph, we run a random walk 11 times to generate 11 sets of node and edge sequences. To generate a new floorplan, we select a floorplan and a set from its 11 sets of node and edge sequences. By decoding the samples from the posterior distribution of these sequences, we get new sequences. There could be different strategies to produce a new floorplan with these newly generated sequences. We select the node with the highest degree that is repeated in all sequences as the central node. Therefore, the arrangement of other rooms can be fixed relatively. These new sequences give us information about room types and square footage. Our method does not encode the rooms' geometry shape; therefore, we can assume the room shapes are similar to the initially selected floorplan or any arbitrary shapes that satisfy the generated square footages (Fig. 6). Generating new floorplans with this approach is limited and gives us similar floorplans regarding the number of rooms and room types. The only changes in new floorplans are the square footage and geometry shape of rooms. The square footage generated in the new sequences does not have the same value for each room. However, considering the original sequences as references, the average generated square footage can be used for a new floorplan.

5.2 Homotopies

VAE makes a continues embedding space, and it allows interpolation in this space. We used the concept of homotopy that means the set of points on the line between two embedding vectors. Instead of a set of random points, we limit our experiment to the point in the middle of the line. We can select two random sequences from two random floorplans, and after encoding them, we can calculate the difference of their embedding vectors. Adding half of their difference to the base vector and decoding it gives us a new sequence. Figure 7 shows an example. We can generate new floorplans by replacing the newly generated sequence with the old random sequence from the original floorplan. It can happen between any other random sequences, and in this way, we can generate more derivative samples. Different strategies for interpolation and generating new floorplans could be used here. With homotopy, we do not have the mentioned limitations in sampling from the posterior distribution. Floorplans with varying room types can be generated as a result of interpolation, and the only limitation is the geometry shape of rooms, which is not encoded. We can assume the geometry shapes are the same as reference floorplans or any arbitrary shapes that satisfy the generated square footages to address this limitation.

6 User study

In this section, we present a user study to evaluate the quality and efficiency of our models of graph embeddings. Three different embedding models are tested: (1) trained with design

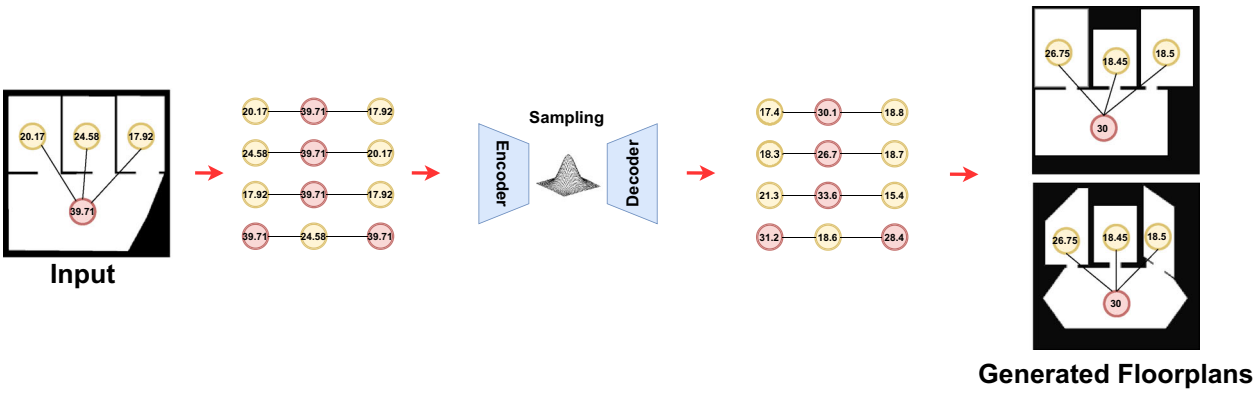

Fig. 7 Floorplan generation with sampling from posterior distributions. To generate a new floorplan, we select a floorplan from our dataset and a set from this floorplan's 11 sets of node and edge sequences. By decoding the samples from the posterior distribution of these sequences, we obtain new sequences and then map them into new floorplans

Table 4 Demographic information and domain knowledge ratings of expert participants (self-reported)

Gender	Sex	Age	Country of Residence
Demographic information			
Female: 4 (40%)	Female: 4 (40%)	18 – 24 years old: 4 (40%)	China: 1 (10%)
Male: 6 (60%)	Male: 6 (60%)	25 – 34 years old: 4 (40%)	United States: 4 (40%)
		35 – 44 years old: 2 (20%)	Canada: 5 (50%)

	Poor	Below average	Average	Above average	Excellent	Avg. scale
Domain knowledge						
Ability to interpret architectural or interior designs?	0 (0%)	1 (10%)	1 (10%)	7 (70%)	1 (10%)	3.80
Prior experience with architecture or interior designs?	2 (20%)	0 (0%)	1 (10%)	6 (60%)	1 (10%)	3.40
Prior experience in urban planning and design?	3 (30%)	0 (0%)	2 (20%)	5 (50%)	0 (0%)	2.90
Prior understanding of computational tools for architectural design space exploration?	2 (20%)	0 (0%)	3 (30%)	5 (50%)	0 (0%)	3.10
Prior understanding of pedestrian movement flow or crowd flow?	0 (0%)	2 (20%)	4 (40%)	3 (30%)	1 (10%)	3.30

semantic features only on nodes, (2) design semantic features both on nodes and edge, and (3) design semantic both on nodes and edges and behavioral features. Given a floorplan (input), we query five similar floorplans (nearest neighbors) from each embedding model.

6.1 Hypothesis

Our hypothesis is twofold: (a) the user-perceived sequence of floorplans as top five nearest neighbors match with the sequence captured by our model as nearest neighbors and (b) users perform better in their perceived sequence of top five nearest neighbors for models (2) and (3) than model (1) which is only trained with design semantic features.

6.2 Apparatus

Floorplans are presented as 2D blueprints (e.g., a top-down skeletal view of an environment layout). The users (e.g., study participants) viewed these blueprints as high-resolution images on their own computer screens via an online survey. For model (1), each room in a floorplan is annotated with room dimension (e.g., square footage area) and color-coded with respect to its room type. The annotation for model (2) is similar to model (1) with the addition of edges between

rooms and their respective directions (e.g., north, east, west, and south). For model (3), we showed color-coded trajectories of virtual occupants from the rooms they spawned into the exit, along with the square footage area of each.

6.3 Participants

Ten (10) domain experts from the architecture community (4 female and 6 male) voluntarily participated in the user study. Table 4 shows the demographic information and domain knowledge of the experts. On average, all the participants had above average experience and expertise in interpreting architecture designs and were knowledgeable of computational tools for design space exploration (self-reported). In addition, every participant was asked for consent before the start of the study.

6.4 Procedure and task

The user study is conducted as an online survey and delivered in four parts. In part (a), users are asked to provide their demographic information and report the domain knowledge and expertise in architecture and urban design. In part (b), users are presented with five different input floorplans. For each input floorplan, a sequence of 5 nearest neighbors is presented in a randomized order, retrieved using model (3), and presented to the users "without" any visual annotations. Users are asked to interactively reorder the given sequence of floorplans (e.g., via drag and drop) based on their perceived "similarity" of these floorplans with respect to the input floorplan. The ordering sequence is arranged such that, more a floorplan is toward the left in the order, the nearest it gets to the input floorplan. In parts (c), (d), and (e), the nearest neighbors are retrieved using models (1), (2), and (3), respectively, for the identical five input floorplans which are used in part (b). In parts (c), (d), and (e), the floorplans are presented to the users "with" visual annotations for their respective features. We estimated that the user study would take up to 15 minutes at maximum to complete.

6.5 Independent and dependent variables

Input floorplans and the retrieved nearest neighbors from the models are the primary independent variables. The rearranged sequences of floorplans by the users are the only dependent variables.

6.6 Results

Figure 8 shows the user-ordered sequences of the nearest neighbors for the three models from the user study. The colored bars for each neighbor of an input floorplan represent the number of users who correctly perceived the neighbor's order in the given sequence. Overall, about 28.68% of the neighbors are accurately ordered in their sequences based on their perceived similarity with respect to input floorplans for model (1), 59.28% for model (2), and about 77.6% for model (3), collectively by all the users. These results highlight that users least performed when they had to perceive the similarity between floorplans by considering the design semantic features alone. In contrast, they performed comparatively better when presented with the neighbors annotated with edge and/or behavioral attributes. The users performed the best for the model (3) when presented with the floorplans visually annotated with design semantics (e.g., room types), edge (e.g., the movement direction of the agents), and behavioral (e.g., movement flow of the agents) features. The findings from the user study suggest that both of our hypotheses stand valid.

We also wanted to analyze the users' performance in perceiving the ordering sequence of the neighbors when floorplans are not visually annotated with their respective features. To test this, we used the input floorplans and their neighbors from model (3) and presented them as the model (0) in the user study. These floorplans were presented to the users without any visual annotations. This was to analyze how important the visual annotation of the features is and its significance to assist users in perceiving the neighbors in their correct order. Interestingly, about 45.68% of the neighbors were accurately ordered in their sequences based on their perceived similarity with respect to input floorplans for model (0). This result revealed that the annotations for design semantic features alone are not a good representative to convey the spatial feature information of the floorplans. As well, that the users better perceive the floorplans retrieved from the embedding space that is trained not only with the design semantic feature alone but also with the additional edge and/or dynamic behavioral features.

7 Conclusion

This paper aims to represent floorplans with numerical vectors such that design semantic and human behavioral features are encoded. Precisely, the framework consists of two components. In the first component, an automated tool is designed for converting floorplan images to attributed graphs. The attributes are designed semantic and human behavioral features generated by simulation. In the second component, we proposed a novel LSTM variational autoencoder for both embedding and generating floorplans. The qualitative, quantitative, and expert evaluation shows our embedding framework produces meaningful and accurate vector representations for floorplans, and its abilities for generating new floorplans are showcased. Besides, we make our dataset public to facilitate the research in this domain. This dataset

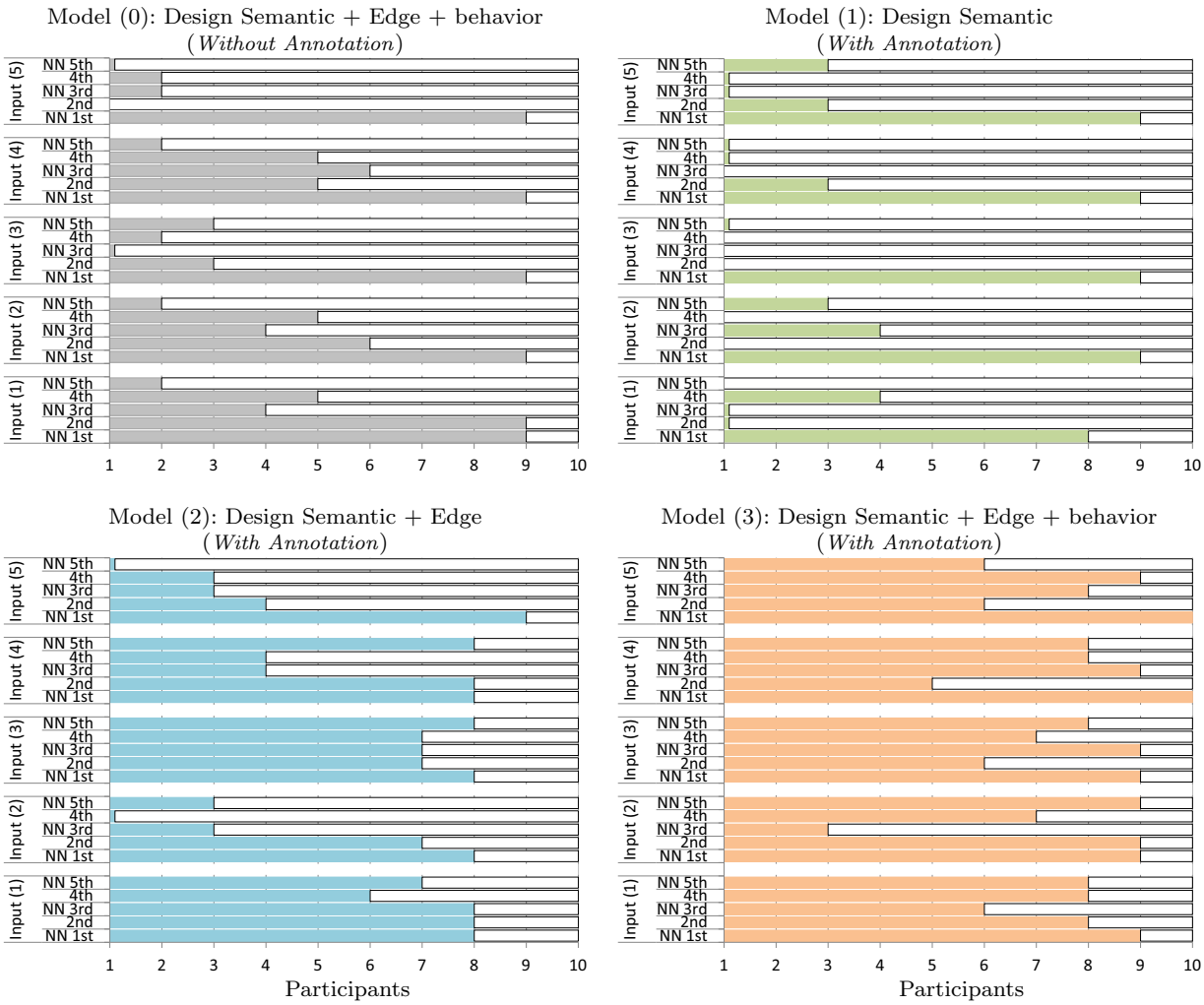

Fig. 8 Accuracy of user-ordered sequences of the nearest neighbors. Colored bars for each neighbor of an input floorplan represent the number of users who correctly perceived the neighbor's order in the given sequence. Gray bars are for the nearest neighbors, which are queried using model (3) but were presented to users without any annotations, Green bars are for nearest neighbors who are queried using model (1) and were presented with annotations, blue bars using model (2)—with annotations, and orange bars using model (3)—with annotations

includes both the extracted design semantic features and simulation-generated human behavioral features.

This contribution holds promise to pave the way for novel developments in automated floorplan clustering, exploration, comparison, and generation. By encoding latent features in the floorplan embedding, designers can store multi-dimensional information of a building design to identify floorplan alterations that share similar or different features quickly. While in this work, we encode features derived from dynamic crowd simulations of building occupancy. The proposed approach can virtually scale to encode any static or dynamic performance metric.

Acknowledgements This research has been partially funded by grants from ISSUM, Ontario Graduate Scholarship, and in part by NSF awards: IIS-1703883, IIS-1955404, and IIS-1955365. The authors would also like to thank Mathew Schwartz for helping in editing and proofreading the manuscript.

References

1. Ahonen, T., Hadid, A., Pietikainen, M.: Face description with local binary patterns: application to face recognition. TPAMI **12**, 2037–2041 (2006)
2. Azizi, V., Usman, M., Patel, S., Schaumann, D., Zhou, H., Faloutsos24, P., Kapadia, M.: Floorplan embedding with latent semantics and human behavior annotations. In: Proceedings of the Symposium on Simulation for Architecture and Urban Design, pp. 43–50 (2020)
3. Azizi, V., Usman, M., Zhou, H., Faloutsos, P., Kapadia, M.: Houseexpo dataset augmented with crowd behavioral features. https://github.com/VahidAz/Floorplan_dataset (2020)

4. Bai, Y., Ding, H., Qiao, Y., Marinovic, A., Gu, K., Chen, T., Sun, Y., Wang, W.: Unsupervised inductive graph-level representation learning via graph-graph proximity. arXiv preprint arXiv:1904.01098 (2019)
5. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. **5**(2), 157–166 (1994)
6. Berseth, G., Usman, M., Haworth, B., Kapadia, M., Faloutsos, P.: Environment optimization for crowd evacuation. Computer Anim. Virtual Worlds **26**(3–4), 377–386 (2015)
7. Bowman, S.R., Vilnis, L., Vinyals, O., Dai, A.M., Józefowicz, R., Bengio, S.: Generating sentences from a continuous space. CoRR **abs/1511.06349** (2015). arxiv:1511.06349
8. Cai, H., Zheng, V.W., Chang, K.C.: A comprehensive survey of graph embedding: Problems, techniques and applications. CoRR **abs/1709.07604** (2017). arxiv:1709.07604
9. Chaillou, S.: Ai+ Architecture: Towards a New Approach. Harvard University, Cambridge (2019)
10. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection (2005)
11. Dutta, A., Llados, J., Pal, U.: Symbol spotting in line drawings through graph paths hashing. In: DAR, pp. 982–986. IEEE (2011)
12. Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. Kdd **96**, 226–231 (1996)
13. Feng, T., Yu, L.F., Yeung, S.K., Yin, K., Zhou, K.: Crowd-driven mid-scale layout design. ACM Trans. Graph. 35(4) (2016)
14. Gonzalez, J.E., Xin, R.S., Dave, A., Crankshaw, D., Franklin, M.J., Stoica, I.: Graphx: graph processing in a distributed dataflow framework. In: 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14), pp. 599–613 (2014)
15. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: KDD, pp. 855–864. ACM (2016)
16. Haworth, B., Usman, M., Berseth, G., Kapadia, M., Faloutsos, P.: Evaluating and optimizing level of service for crowd evacuations. In: Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games, pp. 91–96 (2015)
17. Haworth, B., Usman, M., Berseth, G., Kapadia, M., Faloutsos, P.: On density-flow relationships during crowd evacuation. Computer Anim. Virtual Worlds **28**(3–4), e1783 (2017)
18. de las Heras, L.P., Fernández, D., Fornés, A., Valveny, E., Sánchez, G., Lladós, J.: Runlength histogram image signature for perceptual retrieval of architectural floor plans. In: Workshop on Graphics Recognition, pp. 135–146. Springer (2013)
19. Heylighen, A., Neuckermans, H.: A case base of case-based design tools for architecture. Computer-Aided Des. **33**(14), 1111–1122 (2001)
20. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
21. Hu, R., Huang, Z., Tang, Y., van Kaick, O., Zhang, H., Huang, H.: Graph2plan: Learning floorplan generation from layout graphs. arXiv preprint arXiv:2004.13204 (2020)
22. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2014)
23. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
24. Kramer, M.A.: Nonlinear principal component analysis using autoassociative neural networks. AIChE J. **37**(2), 233–243 (1991)
25. Kumar, P., Huang, H.H.: G-store: high-performance graph store for trillion-edge processing. In: SC'16, pp. 830–841. IEEE (2016)
26. Lambert, G., Gao, H.: Line moments and invariants for real time processing of vectorized contour data. In: International Conference on Image Analysis and Processing, pp. 347–352. Springer (1995)
27. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR, vol. 2, pp. 2169–2178. IEEE (2006)
28. Li, T., Ho, D., Li, C., Zhu, D., Wang, C., Meng, M.Q.H.: Houseexpo: A large-scale 2d indoor layout dataset for learning-based algorithms on mobile robots (2019)
29. Low, Y., Gonzalez, J., Kyrola, A., Bickson, D., Guestrin, C., Hellerstein, J.M.: Distributed graphlab: A framework for machine learning in the cloud. arXiv preprint arXiv:1204.6078 (2012)
30. Maaten, L.V.D., Hinton, G.: Visualizing data using t-sne. J. Mach. Learn. Res. **9**, 2579–2605 (2008)
31. Merrell, P., Schkufza, E., Koltun, V.: Computer-generated residential building layouts. In: ACM SIGGRAPH Asia 2010 papers, pp. 1–12 (2010)
32. Mousavi, S.F., Safayani, M., Mirzaei, A., Bahonar, H.: Hierarchical graph embedding in vector space by graph pyramid. Pattern Recognit. **61**, 245–254 (2017)
33. Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., Jaiswal, S.: graph2vec: Learning distributed representations of graphs. arXiv preprint arXiv:1707.05005 (2017)
34. Nauata, N., Chang, K.H., Cheng, C.Y., Mori, G., Furukawa, Y.: House-gan: Relational generative adversarial networks for graph-constrained house layout generation (2020)
35. Niepert, M., Ahmed, M., Kutzkov, K.: Learning convolutional neural networks for graphs. In: International Conference on Machine Learning, pp. 2014–2023 (2016)
36. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk. Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '14 (2014). doi:https://doi.org/10.1145/2623330.2623732
37. Richter, K., Heylighen, A., Donath, D.: Looking back to the future. an updated case base of case-based design tools for architecture. pp. 285–292 (2007)
38. Rodrigues, E., Gaspar, A.R., Gomes, Á.: An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, part 1: methodology. Computer-Aided Des. **45**(5), 887–897 (2013)
39. Rodrigues, E., Gaspar, A.R., Gomes, Á.: An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, part 2: Validation and performance tests. Computer-Aided Des. **45**(5), 898–910 (2013)
40. Sabri, Q.U., Bayer, J., Ayzenshtadt, V., Bukhari, S.S., Althoff, K.D., 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, pp. 50–60 (2017)
41. Sharma, D., Chattopadhyay, C.: High-level feature aggregation for fine-grained architectural floor plan retrieval. IET Computer Vis. **12**(5), 702–709 (2018)
42. Sharma, D., Chattopadhyay, C., Harit, G.: A unified framework for semantic matching of architectural floorplans. In: Pattern Recognition, pp. 2422–2427. IEEE (2016)
43. Sharma, D., Gupta, N., Chattopadhyay, C., Mehta, S.: Daniel: A deep architecture for automatic analysis and retrieval of building floor plans. In: DAR, vol. 1, pp. 420–425. IEEE (2017)
44. Shervashidze, N., Schweitzer, P., Leeuwen, E.J.V., Mehlhorn, K., Borgwardt, K.M.: Weisfeiler-lehman graph kernels. J. Mach. Learn. Res. **12**, 2539–2561 (2011)
45. Simonovsky, M., Komodakis, N.: Graphvae: Towards generation of small graphs using variational autoencoders. In: International Conference on Artificial Neural Networks, pp. 412–422. Springer (2018)
46. Singh, S., Kapadia, M., Faloutsos, P., Reinman, G.: An open framework for developing, evaluating, and sharing steering algorithms. In: International Workshop on Motion in Games, pp. 158–169. Springer (2009)
47. Sohn, S.S., Zhou, H., Moon, S., Yoon, S., Pavlovic, V., Kapadia, M.: Laying the foundations of deep long-term crowd flow prediction. In: Proceedings of the European Conference on Computer Vision (ECCV) (2020)

48. Taheri, A., Gimpel, K., Berger-Wolf, T.: Learning graph representations with recurrent neural network autoen-coders. KDD Deep Learning Day (2018)
49. Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S., et al.: Constrained k-means clustering with background knowledge. Icml **1**, 577–584 (2001)
50. Wang, W., Gan, Z., Xu, H., Zhang, R., Wang, G., Shen, D., Chen, C., Carin, L.: Topic-guided variational autoencoders for text generation. arXiv preprint arXiv:1903.07137 (2019)
51. Weber, M., Liwicki, M., Dengel, A.: A. scatch-a sketch-based retrieval for architectural floor plans. In: Frontiers in Handwriting Recognition, pp. 289–294. IEEE (2010)
52. Wu, W., Fu, X.M., Tang, R., Wang, Y., Qi, Y.H., Liu, L.: Data-driven interior plan generation for residential buildings. ACM Trans. Gr. (TOG) **38**(6), 1–12 (2019)

Vahid Azizi is a Ph.D. student at Rutgers University in the Department of Computer Science. He earned his master's degree in computer science at Jacobs University. His research interests lie in deep learning, computer vision, and graph representation learning. Projects that he has been working on includes modeling and understanding crowd dynamics, enabling intelligent and automatic floorplan design, recommendation systems, and social network analysis.

Muhammad Usman is a postdoctoral fellow in the Department of Electrical Engineering and Computer Science at York University. He received his Ph.D. degree (2020) and M.Sc. degree (2016) in computer science from York University. His research interests include human–building interaction, crowd steering behaviors in virtual environments, computer-aided architectural design tools, spatial analysis of environments, spatial visualizations, and virtual reality.

Honglu Zhou is a Ph.D. student at Rutgers University in the Department of Computer Science. Her research interests lie in deep learning, computer vision, and graph representation learning. Projects that she has been working on include modeling and understanding crowd dynamics, enabling intelligent and automatic floorplan design, and predicting online information spread. She is currently researching on how to augment deep neural networks with reasoning capabilities to enrich higher-level spatiotemporal understating.

Petros Faloutsos is a Professor at the Department of Electrical Engineering and Computer Science at York University and an affiliate Scientist at the UHN-Toronto Rehabilitation Institute. Before joining York, he was a faculty member at the Computer Science Department at the University of California at Los Angeles, where in 2002 he founded the first computer graphics laboratory at UCLA. Faloutsos received his Ph.D. degree (2002) and his M.Sc. degree in computer science from the University of Toronto, Canada, and his BEng degree in electrical engineering from the National Technical University of Athens, Greece.

Mubbasir Kapadia is the Director of the Intelligent Visual Interfaces Lab and an Assistant Professor in the Computer Science Department at Rutgers University. Previously, he was an Associate Research Scientist at Disney Research Zurich. Kapadia's research lies at the intersection of artificial intelligence, visual computing, and human–computer interaction, with a mission to develop intelligent visual interfaces to empower content creation for human-aware architectural design, digital storytelling, and serious games.