GlobalMapper: Arbitrary-Shaped Urban Layout Generation

Liu He
Purdue University
he425@purdue.edu

Daniel Aliaga
Purdue University
aliaga@purdue.edu

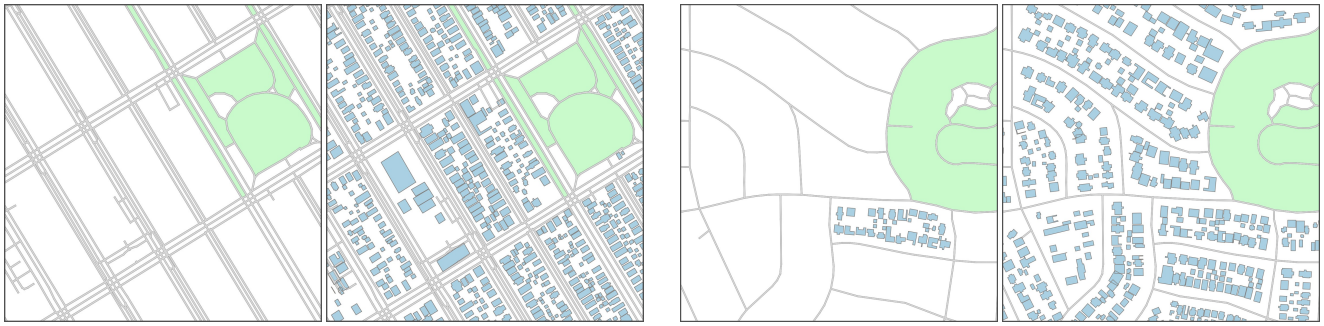

Figure 1: **Arbitrary Urban Layout Generation.** Our method generates realistic urban layouts given an arbitrary road network (Left Pair). Moreover, given learned prior from a building layout, our method conditionally generates similar urban layouts for the given arbitrary road network (Right Pair).

Abstract

Modeling and designing urban building layouts is of significant interest in computer vision, computer graphics, and urban applications. A building layout consists of a set of buildings in city blocks defined by a network of roads. We observe that building layouts are discrete structures, consisting of multiple rows of buildings of various shapes, and are amenable to skeletonization for mapping arbitrary city block shapes to a canonical form. Hence, we propose a fully automatic approach to building layout generation using graph attention networks. Our method generates realistic urban layouts given arbitrary road networks, and enables conditional generation based on learned priors. Our results, including user study, demonstrate superior performance as compared to prior layout generation networks, support arbitrary city block and varying building shapes as demonstrated by generating layouts for 28 large cities.

1. Introduction

Layout generation is of significant interest today in computer vision, computer graphics, and related fields. In particular, an urban building layout consists of a set of buildings arranged into arbitrarily shaped city blocks as defined by a network of interconnected roads. Such layouts are needed in order to provide urban configurations for entertainment, simulation, and scientific applications such as designing and evolving cities to address urban weather forecasting (Fig. 5), urban heat island modeling, and sky view factor analysis, etc.

The capture and modeling of complex urban building layouts has been pursued from several directions. 1) Computer vision and photogrammetry works seek to metrically reconstruct building layouts over large areas from satellite and/or aerial imagery [11, 40, 12, 21, 64, 70, 34, 36, 15]. Nonetheless, these approaches are constrained by image resolution, occlusion, and segmentation accuracy. 2) Urban procedural modeling has made significant strides in generating the hierarchical components of a city such as road networks, city blocks, parcels, buildings, and more (e.g., buildings [39, 67, 5], roads [42, 10], trees [22, 24, 31, 69], parcels [35, 57] and even entire cities [56, 4]). Unfortunately, the generation process requires time consuming and expert knowledge in developing custom procedural rules to create or update the content. 3) Inverse urban procedural modeling attempts to remove the need of explicit rule specification by encoding heuristics specific to cities [6, 68].

Recently, some solutions have addressed inverse modeling for layout generation; e.g., using graph-based meth-

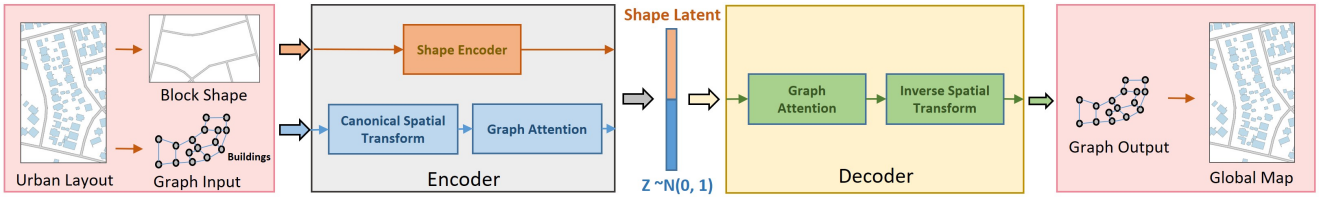

Figure 2: **Framework.** Our method is trained in a VAE framework conditioned on arbitrary block shapes (e.g., from a road network). The block shape is a binary mask encoded to latent space by a CNN-based encoder. Building layouts are represented by a grid topology graph and its positional, geometrical, and shape information is encoded by a spatial transform into a canonical representation. By using a graph attention network we codify the graph into a latent space. During decoding, the block shape latent is utilized as condition for decoder to produce building layouts. Our method is able to conditionally generate large urban layouts for arbitrary block shapes and building arrangements (i.e., we have processed 28 cities).

ods for structured object creation (StructureNet [38]), interior layout design (HouseGAN [41]), land lot generation (BlockPlanner [63]), and building generation (BuildingGAN [8]). However, none of these works focus on generating urban layouts containing arbitrary city block shapes and varying building footprints such as those appearing in cities worldwide.

Our work builds on three key observations.

- First, following the organization defined by procedural modeling arbitrarily-shaped buildings within an arbitrarily-shaped city block can be organized as a discrete spatial data structure of non-overlapping objects arranged into a few rows of buildings (e.g., one row for large buildings, two rows for dense smaller structures, and multiple rows for sparser areas having main buildings and auxiliary building structures).
- Second, capturing the aforementioned building layout configuration can be performed with at least a 2D message passing setup as afforded by a stubby grid graph (i.e., one row of connected nodes/buildings, two rows forming a ladder graph, or a few rows interconnected as a shallow regular grid, as in Fig. 3.
- Third, by using skeletonization we can perform a spatial transform to the aforementioned stubby graph topology to support diverse urban block shapes.

Altogether, our fully automatic approach for building layout generation uses a variational auto-encoder framework using graph neural networks. First, we define a graph-based representation of a block and building layouts that is able to capture the inherent styles of multiple cities globally. Second, the encoder makes use of multi-pass aggregation and combination message passing as well as a spatial transform to codify varying building shapes and layouts for arbitrary city block shapes (Fig. 1). Third, the decoder generates varying building shapes and layouts from the latent space conditioned on block shape (i.e., the distribution of building layouts depends on the block shape). Unlike most prior graph-based method, our scheme is able to accurately capture block shape, building shape type, positional data, and alignment as well. Finally, our synthesis process generates arbitrary maps based on the generated graph, potentially spanning a large continuous area (e.g. a entire city).

To the best of our knowledge, our method is the first to generate building layouts in arbitrary city blocks, and is able to span diverse styles of 28 large cities across North America (e.g., Chicago, Los Angeles, New York City... full list in Sec. 4.1). In Fig. 1, our method generates realistic urban layouts given arbitrary shaped road networks, and it provides controllable generation given building layout priors. We provide comparisons to several prior related methods [2, 17, 26, 63, 25, 20] and show that none of them supports arbitrary block shapes, various building shapes and layouts, nor have any been demonstrated at the scale of our method. In addition, we show examples of conditioned generation, manipulation, interpolation, and ablation study.

Our main contributions include:

- *Graph-based Representation* for arbitrarily shaped city blocks and building layouts amenable to deep learning.
- *Canonical Spatial Transformation* to encode building layouts into a shape-independent canonical representation,
- *Controllable Generation* of realistic urban layouts for arbitrary road networks (e.g. random, or conditioned on a learned prior).

2. Related Works

The task of urban layout analysis from images as well as design and modeling has a long history. The seminal paper of [45] fomented urban procedural modeling, including that of building masses [39], urban layouts [1, 16, 35], and parcel generation (e.g., [57], Esri CityEngine). This last work, as well as [7], describe that the building layouts of a city block usually consist of one of two fundamental styles, both of which are afforded by our approach but without the need

to manually provide the procedural template. In addition, interior layout approaches have also been developed such as for furniture layout [66], mid-scale interior layouts [14] and building interiors [37].

Recently, with the rise of deep learning [61, 50, 49, 53, 51], numerous other data-driven and network-based layout generation approaches have been proposed, yielding improved performance and automation once trained. LayoutGAN [33] and LayoutVAE [26] present general 2D layout planning tools. Alternative works [47, 54, 28] focus on document or graphic layout generation. Recent advancements in self-attention networks (e.g. Transformer) have been applied to natural scene and document layout generation by [17, 2, 65]. Latest diffusion models are also introduced to document layout generation [25, 20]. Other methods have focused on urban scenarios. For example, Building-GAN [8] generates 3D building structures, House-GAN [41] arranges rooms within a single floor of a house. Works by Ritchie *et al*. [48] and Wang *et al*. [60, 59] synthesize the layout of indoor scenes. Further, [62, 44] generate interior plans for residential buildings. However, all previous works assume layout generated on a rectangular canvas. Moreover, the position and geometry of the predicted bounding box is scaled to a finite set of categories (e.g. image coordinates [26], vocabulary table [17, 2, 25, 20]). It benefits from rectilinear alignment in some tasks (e.g. document layout generation), but constrains the randomness of building geometry as it may appear anywhere in a city block. To the best of our knowledge, our method is the first to handle arbitrary block-shapes/canvas and to include varying building/bounding-box shapes.

The approaches most relevant to our work are the building-layout generator of [3] and the BlockPlanner method of [63]. Bao *et al*. [3] assumes that a set of hard and soft constraints are provided as input, and then generates a single building layout and proposes a methodology to evaluate the "goodness" of a layout with respect to the provided constraints. While our goal is also to generate building layouts, we seek a multi-building layout spanning an entire city block based on a conditioned input or a provided style, and then perform this task at scale (e.g., a fragment or more of a city). Further, we wish the style, which is analogous to the user-provided constraints in [3], to be inferred from example data and not provided manually.

BlockPlanner [63] generates only rectangular building shapes inside rectangular city blocks. Their graph-based solution generates an approximate solution and then uses a refinement step to obtain zero-gap building output, which is an assumption of theirs (i.e., buildings in dense parts of cities). In comparison, our method adds canonical spatial transform to support arbitrary block shapes, can represent various building shapes, uses a grid topology to afford more complex building layouts rather than only a ring topology, employs attention-based networks instead of strict message passing, supports more buildings per block, and is demonstrated on 28 large cities instead of only 1 (New York City).

Nonetheless, in Sec. 4.2 we perform explicit comparisons to the domain-specific method BlockPlanner [63], to the natural scene layout method LayoutVAE [26], to self-attention networks Gupta *et al*. [17] and its variational modification [2]. In all cases, our approach outperforms these existing methods qualitatively and quantitatively.

3. Building Layout Generation

We describe our approach to deep building layout generation (Fig. 2). First, we summarize our graph representation and normalization process for arbitrary city blocks. Second, we provide details on our encoding phase. Third, we describe our conditional decoding/generation phase. Finally, we provide information on our synthesis phase as well as auxiliary algorithms.

3.1. Graph Representation

A urban layout is defined by a network of roads that forms city blocks and within each city block a set of buildings. In Fig. 3, each city block B is represented by a graph G of the multiple buildings within the block and by a set of block shape features m.

Buildings. We represent the building layout of an arbitrary city block B using a stubby grid graph G (e.g., the graph has many more columns than rows and each vertex typically has 2 or 3 incident edges). We define the graph as $G = \{V, E\}$, where V is a set of node vertices $V = \{v_i\}$, $i \in [1, N]$ that each vertex v_i describes each building, and E is a set of edges $E = \{e_{ij}\}$, $i, j \in [1, N]$ between spatially adjacent nodes/buildings. N is defined as the maximum building count in one block. We empirically set $N = 120$ to handle >99% percent of real city blocks in our dataset.

Each vertex v_i stores a building's geometric and semantic feature information. In particular, the per-vertex (or building) features are the following:

- e_i is the binary existence flag of this building (e.g., a value of one implies the building exists, else zero).
- (x_i, y_i) is the physical position of building center.
- (w_i, h_i) is the width and height of the building's oriented 2D bounding box (the width dimension is parallel to the main axis),
- s_i is an integer representing the building shape type (described below), and
- a_i is the occupancy ratio of the building area to the oriented 2D bounding box area.

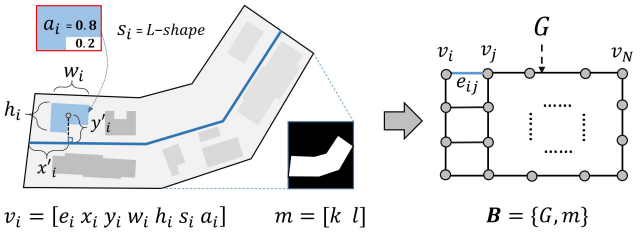

Figure 3: **Graph Representation.** Our method represents any city block B using a normalized stubby graph representation G (i.e., a grid graph with many columns and few rows), and a set of graph-level features m. **Canonical spatial transformation** finds the main axis (blue line) of an arbitrary city block, and transforms each node's spatial information by the relative distance to the main axis.

The aforementioned features $\{s_i, a_i\}$ represent each building by a shape type and an occupancy ratio. In a preliminary analysis, we found that buildings in our test areas can be fit with IoU=95.78% and Hausdorff Distance $= 1.36m$ (i.e., the average of maximum distance between the contour of the actual building and our best-fit polygonal building shape is $1.36m$.) using one of four fundamental parameterized shape types: *Rectanglar, L-shape, U-shape and X-shape*. During the preprocessing phase, we use Powell optimization to determine the best-fitting building shape type and calculate an occupancy ratio (see Supplemental for more details). Both variables enable our later-described synthesis step (Sec. 3.3) to generate a synthetic building footprint.

Blocks. The block shape features m are composed of a binary mask k and the mask scale l. We typically use a mask resolution of 64x64 pixels. The longest side of each block's oriented bounding box is rotated to horizontal and scaled to fit within the mask (see Fig. 3).

3.2. Variational Learning

In this section, we first describe the spatial transform of an arbitrary urban layout into a canonical form, and then describe the shape encoder setup which enables using block shapes as a condition of the decoder setup (Fig. 2). We also discuss our variational model structure and training scheme.

Canonical Spatial Transform. In order to support a wide variety of urban layouts at arbitrary scales, we introduce a transformation parameterized by block shape feature m. It maps each building graph G to a canonical representation G'. Prior urban procedural modeling work [55] indicates that urban layouts follow typical conventions which we exploit to define the canonical representation. In particular, using the contour of block mask k we compute a block's 2D skeleton and modify its endpoints to obtain an extended skeleton as a compact polyline structure forming a main axis (Blue line in Fig. 3). Then, for each building in G we compute its normalized position $(x'_i, y'_i$ in Fig. 3) relative to one of the main axis polyline segments, and to one of the minor axis perpendicular to a main axis line segment. The length of main axis and minor axis is utilized to normalize building's position and size, ultimately producing the canonical G'. This spatial transform provides a regularized spatial representation of building layout regardless of block shape as compared to direct scaling by most former relative methods [63, 26, 17, 2]. Additional details are in Supplemental.

Shape Encoder. To obtain a latent representation of block shape feature m, a CNN-based autoencoder is trained to reconstruct the binary mask k. The block scale l is stacked with the binary mask as the second channel of the input image. The encoder model has 4 layers of convolutional layers and the corresponding decoder model is the mirror image of this. The model is able to reconstruct a binary mask with IoU > 0.98. During our variational training, the latent representation m' encoded by the shape encoder can be utilized as a conditional prior of the decoder.

Conditional Variational Training. As a conditional VAE, our model aims to represent the conditional data distribution of $p(G'|m')$, which is intractable to compute. By variational learning, the model approximates the distribution by maximizing its evidence lower bound (ELBO). The objective function is as follows:

$$\mathcal{L}(\theta, \phi) = \mathop{\mathbb{E}}_{q_\phi(z|G',m')}[\log p_\theta(G'|z, m')] - \mathrm{KL}[q_\phi(z, m'|G')\|p(z)] \tag{1}$$

where the approximate posterior $q_\phi(z|G', m')$ is parameterized by ϕ, and the decoder $p_\theta(G'|z, m')$ is a deep neural network parameterized by θ. The prior distribution $p(z)$ is a standard Gaussian distribution in our training. Particularly, our model should be able to encode and reconstruct G' conditioned by m'.

Graph Attention. We use a graph attention network [58] (GAT) as the backbone of our encoder and decoder. Graph attention networks perform weighted multilayer messaging passing between connected nodes. In particular, the edges of our 2D grid graph topology enable message passing between buildings next-to and in-back-of other buildings. We denote node features in the canonical graph as f_i. Our formulation can be represented by:

$$f_i^t = GAT(f_i^{t-1}) \tag{2}$$

We collect all individual node feature vectors into feature matrix $F^t = \{f_i^t\}, i \in [1, N], t \in [1, T]$ and apply equation Eq. 2 a total of T times. Then, the feature matrices from all T iterations are aggregated as $[F^t, F^1, ..., F^T]$ to form a 512 dimensional normal distribution that is sampled by the z of a variational reparameterization.

The decoder $p_\theta(G'|z, m')$ is conditioned by m'. As Fig. 2 indicates, we concatenate m' with z as the input

of our decoder. Initially, a sampled latent space vector z is pushed through a MLP to produce an initial feature matrix $\hat{F}^0 = \{\hat{f}_i^0\}, i \in [1, N]$, which consists of single feature vector for each node vertex/building. Then, a similar multi-layer messaging passing, using Eq. 2, is performed T times yielding $\hat{F}^T$. Finally, each vector component $\hat{f}_i^T$ within $\hat{F}^T$ is decoded by independent MLPs for predictions of each building feature element in $\hat{G}'$ (e.g., $\{e_i, x_i, y_i, w_i, h_i, s_i, a_i\}$). Predicted features are calculated by reconstruction loss. The weighted sum of each loss component is back-propagated to graph attention networks for weight updating.

Inverse Spatial Transform. With block shape feature m, the spatial information of $\hat{G}'$ (e.g., building position, width, and height) is inversely transformed back to its original scale and location.

3.3. Synthesis

In this phase, we generate building shapes within each generated city block. Our goal is to produce building footprints that are similar (but not necessarily identical) to the original building coverages, positions, and shapes. Further as shown in Sec. 4.4, we can produce such an output using only a small fraction of the real-world data.

Our method synthesizes the aforementioned four building shape types (*Rectangular, L-shape, U-shape, X-shape*) using a parameterized function (see Supplemental for details). For the buildings of a city block, the shape parameters of each generated building type s_i are randomly perturbed within a predefined range until a configuration best satisfying the generated occupancy value a_i is found within a maximum number of iterations. Then, the generated building footprint is rotated so that its width dimension is again parallel to the block's main axis.

4. Experiments

4.1. Training

Datasets. We collected datasets from 28 large cities across North America (Chicago, D.C., New York City, Atlanta, Dallas, Los Angeles, Miami, Seattle, Boston, Providence, Baltimore, San Diego, San Francisco, Portland, Milwaukee, Minneapolis, Austin, New Orleans, Phoenix, Denver, Toronto, Vancouver, Pittsburgh, Tampa, San Jose, Norfolk, Austin, and Houston). The total number of city blocks is 119,236, containing 2,513,697 buildings, which is 187-times bigger than the training set of [63]. We split the data 80% for training, and 20% for validation. We obtained the datasets by downloading 2D layouts from OpenStreetMap (OSM) [43]. Minimal clean-up was done to simplify parallel multi-lane roads to single edges and to remove self-looping edges.

Training Details. We performed an end-to-end training of our pipeline. After various experiments, we set learning rate $lr = 0.001$, and each training typically converges in 10^5 iterations (9 hours on a single NVIDIA A5000 GPU). We determined the best configuration of loss functions to be using $L2$ loss for all geometry features $\{x_i, y_i, w_i, h_i\}$, and cross entropy loss for categorized features $\{e_i, s_i\}$, and KL-divergence loss for z. The relative loss weight for geometry features, categorized features, and KL-divergence loss is 4.0, 1.0, 0.5, respectively. We found additional regularization losses (i.e., a loss penalizing overlaps [63]) do not further reduce errors. The average inference time is 5.81ms per city block by a single A5000 GPU.

4.2. Comparisons

We compared our approach to 4 related layout generation methods: LayoutVAE [26], Gupta *et al*. [17], VTN [2], and BlockPlanner [63]. For first two methods, we utilized the codes provided by the author's repository of [17]. We provided the correct bounding box count for LayoutVAE and only retrained its BBoxVAE portion to predict bounding box geometry. For [17], we transferred our dataset to COCO format and retrained the network using default settings in repository. During generation, we setup their top-k sampling as k=5 to produce diverse layout outputs. For VTN [2], we modified the codes of [17] by adding a variational training framework. For BlockPlanner [63], we only received partial model structure source code from the authors. We re-implemented the rest of the method to the best of our ability, and retrained it on our dataset by the suggested settings in their paper. Additionally, comparisons to diffusion models [25, 20] are provided in Supplemental.

Quantitative Results. We generated 1000 urban layouts by ours and by each existing method, and compared to the same amount of real urban layouts. Our method is conditioned on the road networks from real urban layouts. The LayoutVAE is conditioned on the building count we provided. VTN is conditioned on the corresponding real building layouts. Gupta *et al*. [17] generation is fully random. The set-to-set comparison is evaluated by 6 quantitative metrics. Overlap index [32] is the percentage of total overlapping area among generated building layouts within the urban block. The Out-Block index is the percentage of generated building layout area that is out of the urban block contour. Wasserstein distance (WD) indicates the similarity between two distributions. Specifically, we computed the WD between the distributions of building counts and bounding box geometry (location and size, centered around the origin). Lower score indicates generated urban layouts provide better similarity to the real world distribution of building counts and geometry arrangement. We also computed FID score [23] to evaluate diversity and quality of the generated urban layouts.

Inspired by DocSim index [47], we design a similarity

Method	L-Sim↑	Overlap↓ (%)	Out-Block↓ (%)	FID↓	WD↓ (bbx)	WD↓ ($count$)
LayoutVAE [26]	4.49	33.39	11.15	94.54	7.24	-
BlockPlanner [63]	14.92	9.46	<u>2.24</u>	<u>39.27</u>	6.20	**0.03**
Gupta *et al*. [17]	<u>17.59</u>	3.61	7.58	47.06	<u>2.23</u>	6.12
VTN [2]	17.65	<u>1.49</u>	7.97	46.71	2.78	3.98
Ours	**22.45**	**1.42**	**0.89**	**14.94**	**1.45**	<u>0.06</u>

Table 1: **Quantitative Results.** We generate 1000 urban layouts and compare to the same amount of real urban layouts. Best values are in bold, second best values are underlined. Our method outperforms other existing methods in 5 metrics.

metric LayoutSim (L-Sim) to evaluate the geometry similarity between pairs of urban layouts. We first rotate each of the two urban blocks to make the long side of its oriented bounding box horizontal, and translate to its center location without changing scale. Then we assign a matching score for all pairs of buildings from two different urban layouts. Pairs of buildings that are approximately overlapping and with similar sizes will have higher scores. Then we compute the maximum weight matching among all possible building pairs by Hungarian method [30]. The average matching score is the L-Sim between two urban layouts. The formula for matching score between building b_1 and b_2 is as followed:

$$S(b_1, b_2) = MinArea(b_1, b_2) \cdot 2^{-c\|b_1 - b_2\|_2} \quad (3)$$

The multiplier $MinArea(b_1, b_2)$ returns the minimum area of two buildings. Exponent factor $\|b_1 - b_2\|_2$ is the $L2$ norm between the positions of two buildings. Matching large buildings will tend to increase the matching score. Large position difference will decrease the value of the matching score. We set $c = 0.02$ to scale the relatively large position difference value in the real world.

As Tab. 1 shows, our method clearly outperforms existing methods in 5 metrics and achieves the second best in WD of building counts. Note that we provide building count (the label set as described in original paper [26]) to LayoutVAE. So it is unfair to us regarding the WD distance on building count. We don't report the value in the table.

Qualitative Results. Fig. 4 shows urban layouts generated by our method and other methods (LayoutVAE [26], BlockPlanner [63], and VTN [2]). For improved fairness, we provide ground truth building shape types and occupancy ratio values to our competitors since they did not consider building shapes. Nonetheless, the shown visual quality indicates that our method is still better able to capture arbitrary block shape and provide good generated output. In particular, Blockplanner [63] struggles to produce densely distributed urban layouts as claimed. It might be due to two reasons. First, the model is retrained on a larger and diverse dataset (119,236 blocks, 28 cities) compared to the original paper (637 blocks, 1 city); second, our urban layout generation task differs to its original land lot generation task (i.e., most buildings are not supposed to touch each other as in their original case). VTN [2] produces rectilinear building layouts regardless of block shapes. Strictly-rectilinear layouts are common in document layouts, but far from realistic in urban layouts.

User Study. We conducted a user study for perceptual realism using our method and the above three prior methods. The study was performed in a two-alternative forced choice (2AFC) manner. We generated 18 comparisons, each containing a layout generated by our method and the same layout generated by one of the three prior methods (thus 6 layout pairs for each prior method). The urban locations do not overlap across the three sets. During the study, we presented two urban layouts side-by-side from different methods. Users were asked to choose the one that looks more realistic to the best of their ability. We also included 2 random duplicate questions for quality checking. Users that answered differently to the same question were discarded.

In total 62 survey results were received. We discarded replies that were finished too fast (less than 2 minutes), that always answered the same choice, and that did not pass the quality check. Replies from the remaining 50 valid users are summarized. Our method was preferred by 97.3% of answers over LayoutVAE [26], 86.7% over BlockPlanner [63], and 88.0% over VTN [2].

4.3. Ablation Study

We performed a comprehensive ablation study of adding and removing components of our solution to our method and to prior methods. While the previous comparisons show the superiority of our full method to prior methods as published, we further analyzed the effect of providing our canonical spatial transformation (CST) and also shape encoder (SE) to our competitors. In particular, CST was added to all competitors, and the vector produced by SE was also concatenated to the input vector of Gupta *et al*. [17] and to the latent bottleneck vector of VTN [2] and BlockPlanner [63] before decoding. For ablations on our model structure, we experimented with alternative graph convolutional layers (e.g. GraphSAGE [18], transformer-based [52], and GCN [29]) with depth $T = 3$. In addition, we varied attention depth ($T \in 1, 2, 3$) in our model structure.

We performed one-to-one comparisons using 1000 validation urban layouts, which is a comparison process differ-

Figure 4: **Qualitative Results.** Given the same road network, all above methods generate urban layouts multiple times and we present the most similar one compared to the real data (similarity is evaluated by L-Sim index defined in Section 4.2. LayoutVAE [26] fails in all types of block shapes. VTN [2] produces rectilinear layouts regardless of block shapes. Blockplanner [63] suffers overlapping and struggles under irregular block shapes. As compared to real data, our method is able to capture the layout styles more faithfully.

ent to the set-to-set comparison in Sec. 4.2. This more restrictive comparison task comprehensively evaluates robustness of overlap, position, and coverage rate. Position error is the average percent of building position shifting over the block length. Coverage error is the difference of total building area coverage rate compared to the ground truth rate.

Ablation results in Tab. 2 illustrate that CST and SE both help robustness to arbitrary block shapes. It especially helps reduce Out-Block index and position error. This indicates our novelty has the potential to broadly benefit layout generation and improve adaptability to arbitrary canvas shapes. Moreover, even when both mechanisms are added to all competitors, our method still performs clearly better. Self-ablation results show that graph attention networks outperform other alternative convolutional layers, and the deeper structure improves model performance.

4.4. Controllable Map Generation

Given a pre-trained model, we are able to generate diverse and realistic urban layouts for cities around the world. Apart from the random/conditional generation showed in Fig. 1 and several more in Supplemental, we present several interesting generation options including sparse prior generation, semantic manipulation, and interpolation.

Sparse Priors Generation. Our approach supports conditioned generation given sparse building layouts as prior. In particular, given a road network and a few initial blocks of building layouts, we can generate more blocks with a similar style. Fig. 5 uses only a random 5% of prior building layouts to generate a city: the latent vector of an empty block is calculated as the distance-weighted sum of the nearest k blocks given as prior ($k = 5$ works well). Normal noise is also added to the latent vector for diversity.

This generation ability is useful to provide data equity for many/most cities globally, where complete layout information is not available. To evaluate this, we showcased a well-modeled city. Fig. 5 models Chicago in which we trained building height from [27] as additional building features so that we can generate 3D building masses (still from only 5% of prior). Then, we extracted a 3D urban mor-

Ablations	Overlap↓	Out-Block↓	Pos-E.↓	Cov-E.↓
LayoutVAE [26]	28.14	11.08	18.52	24.02
BlockPlanner [63]	5.87	3.47	8.04	8.41
Gupta *et al*. [17]	3.54	7.43	10.50	0.49
VTN [2]	2.34	7.24	9.42	2.56
[26] + CST	28.00	8.12	15.09	24.34
[63] + CST	3.20	2.08	4.09	6.11
[17] + CST	4.15	5.18	7.69	1.90
[2] + CST	2.32	5.61	6.67	1.97
[63] + CST&SE	2.12	1.93	3.47	2.78
[17] + CST&SE	3.89	4.34	5.65	1.73
[2] + CST&SE	2.25	4.52	5.89	1.88
Ours - CST	4.86	2.23	4.40	3.94
Ours - SE	1.39	2.03	2.69	2.00
Ours - CST&SE	7.09	3.03	7.87	9.88
Ours (Xformer [52])	1.08	4.40	6.00	0.38
Ours (SAGE [18])	1.36	5.04	6.84	0.73
Ours (GCN [29])	1.57	3.44	6.02	0.76
Ours (T=1)	1.38	1.30	3.41	0.88
Ours (T=2)	1.25	1.31	3.32	0.74
Ours* (T=3)	**1.06**	**1.25**	**3.10**	**0.36**

Table 2: **Ablations.** We report metrics (all in %) among all alternative ablations. T is number of stacked graph attention layers. More ablations in Supplemental.

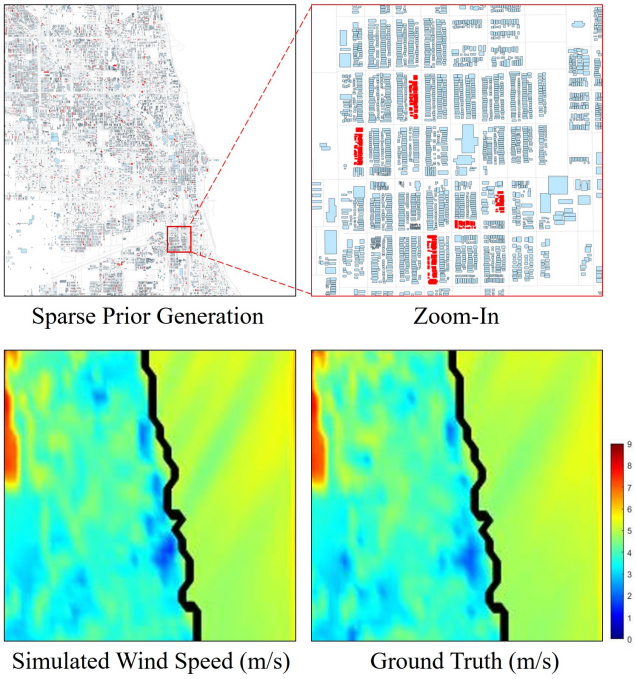

Figure 5: **Sparse Prior Generation.** Top: we show a generated Chicago from only 5% prior data (shown in red), and a Zoom-In. Blank areas indicate non-building structures (e.g., parks, rivers, roads). Bottom: generated layout used by a local weather forecasting model [9] yields almost identical wind speed simulation compared to the ground truth.

phology suitable for an urban weather forecasting system, Urban WRF [9]. The model generated by our method from only 5% prior yields comparable local weather forecasting results as that provided by the ground truth. Our average per-pixel wind-speed prediction error is 0.23m/s (details in Supplemental). Additional results are available in [46].

Semantic Manipulation. A semantic editing of building layouts can be performed by exploiting plausible disentanglements. For example, we labeled building row numbers (e.g., from 1 to 4) of 20K test urban layouts. We found latent vectors for each row-label group form reasonable clusters. If we translate a latent vector of a 1-row building layout towards that of the 2-row building layout cluster center, the layout will gradually have another row of buildings. Results are provided in Supplemental for up to 4 rows of buildings.

Interpolation. Our method can generate urban layouts by interpolating latent space vectors between two layouts. In Supplemental, we show results from linearly interpolating either building layout latent vectors or block shape latent vectors from one to another. The intermediate layouts correspond to an intuitive style interpolation.

5. Conclusions and Future Work

We have presented a controllable graph-based method to generate plausible urban layouts with arbitrary block shapes and varying building shapes. Our approach exploits multi-layer message passing using graph attention networks to encode and decode/generate the relationship between adjacent building structures. As opposed to image and pixel based methods, our technique exploits that urban building layouts are discrete, uses a stubby grid graph topology to support multiple rows of building structures, fits buildings to a taxonomy of parameterized building shapes in order to support a variety of building forms, and uses a skeletonization algorithm to map arbitrary city block shapes to a canonical form in order to better support a deep learning based approach. Our results, including user study, show superior performance to prior methods (e.g., LayoutVAE [26], BlockPlanner [63], Gupta *et al*. [17], and VTN [2]), analyzes different message passing schemes [29, 19, 13, 58], and demonstrates many examples in large cities.

Our approach does have some limitations. First, our approach cannot represent city block contours with interior boundaries. Second, we do not support all possible building shapes. Third, only up to a fixed maximum number of buildings are supported in a block. Our method can handle common dead-end roads, such as cul-de-sac's (see Supplemental), but not all cases of dead-end roads.

As future work, we would like to ingest additional building semantics to produce more elaborate building structures, support further out-of-distribution layouts, and scale our method to cities worldwide.

6. Acknowledgement

This research is at least partially supported by NSF grants 1835739 and 2106717, and by Purdue Institute of Digital Forestry (esp. Dr. Songlin Fei). In addition, we thanks UT Austin collaborators Dr. Dev Niyogi and Harsh Kamath for height dataset. Thanks to Haoteng Yin, and Zhiquan Wang for their valuable suggestions.

References

[1] Daniel G Aliaga, Carlos A Vanegas, and Bedrich Benes. Interactive example-based urban layout synthesis. *ACM transactions on graphics (TOG)*, 27(5):1–10, 2008. 2

[2] Diego Martin Arroyo, Janis Postels, and Federico Tombari. Variational transformer networks for layout generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13642–13652, 2021. 2, 3, 4, 5, 6, 7, 8

[3] Fan Bao, Dong-Ming Yan, Niloy J. Mitra, and Peter Wonka. Generating and exploring good building layouts. *ACM Trans. Graph.*, 32(4), jul 2013. 3

[4] Bedrich Benes, Xiaochen Zhou, Pascal Chang, and Marie-Paule R Cani. Urban brush: Intuitive and controllable urban layout editing. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, pages 796–814, 2021. 1

[5] Manush Bhatt, Rajesh Kalyanam, Gen Nishida, Liu He, Christopher May, Dev Niyogi, and Daniel Aliaga. Design and deployment of photo2building: A cloud-based procedural modeling tool as a service. In *Practice and Experience in Advanced Research Computing*, pages 132–138. 2020. 1

[6] Martin Bokeloh, Michael Wand, and Hans-Peter Seidel. A connection between partial symmetry and inverse procedural modeling. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, New York, NY, USA, 2010. Association for Computing Machinery. 1

[7] Matthew Carmona. *Public places urban spaces: The dimensions of urban design*. Routledge, 2021. 2

[8] Kai-Hung Chang, Chin-Yi Cheng, Jieliang Luo, Shingo Murata, Mehdi Nourbakhsh, and Yoshito Tsuji. Building-gan: Graph-conditioned architectural volumetric design generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11956–11965, 2021. 2, 3

[9] Fei Chen, Hiroyuki Kusaka, Robert D. Bornstein, Jason Ching, C. Sue B. Grimmond, Susanne Grossman-Clarke, Thomas Loridan, Kevin W. Manning, Alberto Martilli, Shiguang Miao, David J. Sailor, Francisco Salamanca, Haider Taha, Mukul Tewari, Xuemei Wang, Andrzej A. Wyszogrodzki, and Chao-Lin Zhang. The integrated wrf/urban modelling system: development, evaluation, and applications to urban environmental problems. *International Journal of Climatology*, 31, 2011. 8

[10] Guoning Chen, Gregory Esch, Peter Wonka, Pascal Müller, and Eugene Zhang. Interactive procedural street modeling. In *ACM SIGGRAPH 2008 papers*, pages 1–10. 2008. 1

[11] Dominic Cheng, Renjie Liao, Sanja Fidler, and Raquel Urtasun. Darnet: Deep active ray network for building segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7431–7439, 2019. 1

[12] Ilke Demir, Krzysztof Koperski, David Lindenbaum, Guan Pang, Jing Huang, Saikat Basu, Forest Hughes, Devis Tuia, and Ramesh Raska. Deepglobe 2018: A challenge to parse the earth through satellite images. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 172–17209. IEEE, 2018. 1

[13] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *CoRR*, abs/2012.09699, 2020. 8

[14] Tian Feng, Lap-Fai Yu, Sai-Kit Yeung, KangKang Yin, and Kun Zhou. Crowd-driven mid-scale layout design. *ACM Trans. Graph.*, 35(4), jul 2016. 3

[15] Adnan Firoze, Cameron Wingren, Raymond A Yeh, Bedrich Benes, and Daniel Aliaga. Tree instance segmentation with temporal contour graph. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2193–2202, 2023. 1

[16] Saskia A Groenewegen, Ruben M Smelik, Klaas Jan de Kraker, and Rafael Bidarra. Procedural city layout generation based on urban land use models. *Short Paper Proceedings of Eurographics 2009*, 2009. 2

[17] Kamal Gupta, Justin Lazarow, Alessandro Achille, Larry S Davis, Vijay Mahadevan, and Abhinav Shrivastava. Layouttransformer: Layout generation and completion with self-attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1004–1014, 2021. 2, 3, 4, 5, 6, 8

[18] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017. 6, 8

[19] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1025–1035, Red Hook, NY, USA, 2017. Curran Associates Inc. 8

[20] Liu He, Yijuan Lu, John Corring, Dinei Florencio, and Cha Zhang. Diffusion-based document layout generation. *arXiv preprint arXiv:2303.10787*, 2023. 2, 3, 5

[21] Liu He, Jie Shan, and Daniel Aliaga. Generative building feature estimation from satellite images. *IEEE Transactions on Geoscience and Remote Sensing*, 2023. 1

[22] Liu He, Haoxiang Yang, and Yuchun Huang. Automatic pole-like object modeling via 3d part-based analysis of point cloud. In *Remote Sensing Technologies and Applications in Urban Environments*, volume 10008, pages 233–248. SPIE, 2016. 1

[23] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 5

[24] Yuchun Huang, Ping Ma, Zheng Ji, and Liu He. Part-based modeling of pole-like objects using divergence-incorporated 3-d clustering of mobile laser scanning point clouds. *IEEE Transactions on Geoscience and Remote Sensing*, 59(3):2611–2626, 2020. 1

[25] Naoto Inoue, Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. Layoutdm: Discrete diffusion model for controllable layout generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10167–10176, 2023. 2, 3, 5

[26] Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. Layoutvae: Stochastic scene layout generation from a label set. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9894–9903, 2019. 2, 3, 4, 5, 6, 7, 8

[27] Harsh G Kamath, Manmeet Singh, Lori A Magruder, Zong-Liang Yang, and Dev Niyogi. Globus: Global building heights for urban studies. *arXiv preprint arXiv:2205.12224*, 2022. 7

[28] Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. Constrained graphic layout generation via latent optimization. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 88–96, 2021. 3

[29] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 6, 8

[30] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 6

[31] Bosheng Li, Jacek Kałużny, Jonathan Klein, Dominik L Michels, Wojtek Pałubicki, Bedrich Benes, and Sören Pirk. Learning to reconstruct botanical trees from single images. *ACM Transactions on Graphics (TOG)*, 40(6):1–15, 2021. 1

[32] Jianan Li, Jimei Yang, Aaron Hertzmann, Jianming Zhang, and Tingfa Xu. Layoutgan: Generating graphic layouts with wireframe discriminators. *arXiv preprint arXiv:1901.06767*, 2019. 5

[33] Jianan Li, Jimei Yang, Aaron Hertzmann, Jianming Zhang, and Tingfa Xu. Layoutgan: Synthesizing graphic layouts with vector-wireframe adversarial networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(7):2388–2399, 2021. 3

[34] Weijia Li, Lingxuan Meng, Jinwang Wang, Conghui He, Gui-Song Xia, and Dahua Lin. 3d building reconstruction from monocular remote sensing images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12548–12557, 2021. 1

[35] Markus Lipp, Daniel Scherzer, Peter Wonka, and Michael Wimmer. Interactive modeling of city layouts using layers of procedural content. *Comput. Graph. Forum*, 30:345–354, 04 2011. 1, 2

[36] Jisan Mahmud, True Price, Akash Bapat, and Jan-Michael Frahm. Boundary-aware 3d building reconstruction from a single overhead image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 441–451, 2020. 1

[37] Paul Merrell, Eric Schkufza, and Vladlen Koltun. Computer-generated residential building layouts. In *ACM SIGGRAPH Asia 2010 Papers*, SIGGRAPH ASIA '10, New York, NY, USA, 2010. Association for Computing Machinery. 3

[38] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas Guibas. Structurenet: Hierarchical graph networks for 3d shape generation. *ACM Transactions on Graphics (TOG), Siggraph Asia 2019*, 38(6):Article 242, 2019. 2

[39] Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. Procedural modeling of buildings. In *ACM SIGGRAPH 2006 Papers*, pages 614–623. 2006. 1, 2

[40] P. Musialski, P. Wonka, D. G. Aliaga, M. Wimmer, L. Van Gool, and W. Purgathofer. A survey of urban reconstruction. *Computer Graphics Forum*, 32(6):146–177, Sept. 2013. 1

[41] Nelson Nauata, Kai-Hung Chang, Chin-Yi Cheng, Greg Mori, and Yasutaka Furukawa. House-gan: Relational generative adversarial networks for graph-constrained house layout generation. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 162–177, Cham, 2020. Springer International Publishing. 2, 3

[42] G. Nishida, I. Garcia-Dorado, and D. G. Aliaga. Example-driven procedural urban roads. *Comput. Graph. Forum*, 35(6):5–17, sep 2016. 1

[43] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2017. 5

[44] Wamiq Para, Paul Guerrero, Tom Kelly, Leonidas J Guibas, and Peter Wonka. Generative layout modeling using constraint graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6690–6700, 2021. 3

[45] Yoav I. H. Parish and Pascal Müller. Procedural modeling of cities. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, page 301–308, New York, NY, USA, 2001. Association for Computing Machinery. 2

[46] Pratiman Patel, Rajesh Kalyanam, Liu He, Daniel Aliaga, and Dev Niyogi. Deep learning-based urban morphology for city-scale environmental modeling. *PNAS nexus*, 2(3):pgad027, 2023. 8

[47] Akshay Gadi Patil, Omri Ben-Eliezer, Or Perel, and Hadar Averbuch-Elor. Read: Recursive autoencoders for document layout generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 544–545, 2020. 3, 5

[48] Daniel Ritchie, Kai Wang, and Yu-An Lin. Fast and flexible indoor scene synthesis via deep convolutional generative models. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6175–6183, 2019. 3

[49] Yichen Sheng, Yifan Liu, Jianming Zhang, Wei Yin, A Cengiz Oztireli, He Zhang, Zhe Lin, Eli Shechtman, and Bedrich Benes. Controllable shadow generation using pixel height maps. In *European Conference on Computer Vision*, pages 240–256. Springer, 2022. 3

[50] Yichen Sheng, Jianming Zhang, and Bedrich Benes. Ssn: Soft shadow network for image compositing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4380–4390, 2021. 3

[51] Yichen Sheng, Jianming Zhang, Julien Philip, Yannick Hold-Geoffroy, Xin Sun, He Zhang, Lu Ling, and Bedrich Benes. Pixht-lab: Pixel height based light effect generation for im-

age compositing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16643–16653, 2023. 3

[52] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020. 6, 8

[53] Yizhi Song, Zhifei Zhang, Zhe Lin, Scott Cohen, Brian Price, Jianming Zhang, Soo Ye Kim, and Daniel Aliaga. Objectstitch: Object compositing with diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18310–18319, 2023. 3

[54] Sou Tabata, Hiroki Yoshihara, Haruka Maeda, and Kei Yokoyama. Automatic layout generation for graphical design magazines. In *ACM SIGGRAPH 2019 Posters*, pages 1–2. 2019. 3

[55] Carlos A Vanegas, Daniel G Aliaga, and Bedrich Benes. Building reconstruction using manhattan-world grammars. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 358–365. IEEE, 2010. 4

[56] Carlos A. Vanegas, Ignacio Garcia-Dorado, Daniel G. Aliaga, Bedrich Benes, and Paul Waddell. Inverse design of urban procedural models. *ACM Trans. Graph.*, 31(6), nov 2012. 1

[57] Carlos A Vanegas, Tom Kelly, Basil Weber, Jan Halatsch, Daniel G Aliaga, and Pascal Müller. Procedural generation of parcels in urban modeling. In *Computer graphics forum*, volume 31, pages 681–690. Wiley Online Library, 2012. 1, 2

[58] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. 4, 8

[59] Kai Wang, Yu-An Lin, Ben Weissmann, Manolis Savva, Angel X. Chang, and Daniel Ritchie. Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Trans. Graph.*, 38(4), jul 2019. 3

[60] Kai Wang, Manolis Savva, Angel X. Chang, and Daniel Ritchie. Deep convolutional priors for indoor scene synthesis. *ACM Trans. Graph.*, 37(4), jul 2018. 3

[61] Lei Wang, Yuchun Huang, Jie Shan, and Liu He. Msnet: Multi-scale convolutional network for point cloud classification. *Remote Sensing*, 10(4):612, 2018. 3

[62] Wenming Wu, Xiao-Ming Fu, Rui Tang, Yuhan Wang, Yu-Hao Qi, and Ligang Liu. Data-driven interior plan generation for residential buildings. *ACM Trans. Graph.*, 38(6), nov 2019. 3

[63] Linning Xu, Yuanbo Xiangli, Anyi Rao, Nanxuan Zhao, Bo Dai, Ziwei Liu, and Dahua Lin. Blockplanner: City block generation with vectorized graph representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5077–5086, 2021. 2, 3, 4, 5, 6, 7, 8

[64] Ziqiang Xu, Chunyan Xu, Zhen Cui, Xiangwei Zheng, and Jian Yang. Cvnet: Contour vibration network for building extraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1383–1391, 2022. 1

[65] Cheng-Fu Yang, Wan-Cyuan Fan, Fu-En Yang, and Yu-Chiang Frank Wang. Layouttransformer: Scene layout generation with conceptual and spatial diversity. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3732–3741, 2021. 3

[66] Lap-Fai Yu, Sai-Kit Yeung, Chi-Keung Tang, Demetri Terzopoulos, Tony F. Chan, and Stanley J. Osher. Make it home: Automatic optimization of furniture arrangement. In *ACM SIGGRAPH 2011 Papers*, SIGGRAPH '11, New York, NY, USA, 2011. Association for Computing Machinery. 3

[67] Xiaowei Zhang, Wufei Ma, Gunder Varinlioglu, Nick Rauh, Liu He, and Daniel Aliaga. Guided pluralistic building contour completion. *The Visual Computer*, 38(9-10):3205–3216, 2022. 1

[68] Xiaowei Zhang, Christopher May, and Daniel Aliaga. Synthesis and completion of facades from satellite imagery. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 573–588, Cham, 2020. Springer International Publishing. 1

[69] Xiaochen Zhou, Bosheng Li, Bedrich Benes, Songlin Fei, and Sören Pirk. Deeptree: Modeling trees with situated latents. *arXiv preprint arXiv:2305.05153*, 2023. 1

[70] Stefano Zorzi, Shabab Bazrafkan, Stefan Habenschuss, and Friedrich Fraundorfer. Polyworld: Polygonal building extraction with graph neural networks in satellite images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1848–1857, 2022. 1