BlockPlanner: City Block Generation with Vectorized Graph Representation

Linning Xu[1]*, Yuanbo Xiangli[1]*, Anyi Rao[1], Nanxuan Zhao[1,3], Bo Dai[2], Ziwei Liu[2], Dahua Lin[1,3,4]
[1]The Chinese University of Hong Kong [2]S-Lab, Nanyang Technological University
[3]Centre of Perceptual and Interactive Intelligence [4]Shanghai AI Laboratory
{xl020,xy019,ra018,dhlin}@ie.cuhk.edu.hk nanxuanzhao@gmail.com {bo.dai,ziwei.liu}@ntu.edu.sg

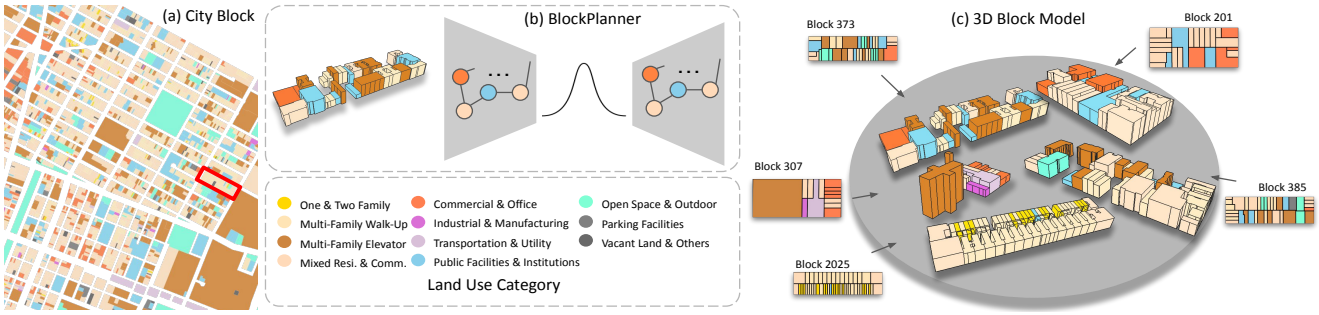

Figure 1: City block forms the basic urban fabric of a city. We illustrate (a) sample blocks taken from Manhattan, NYC. Colors are assigned according to land use categories. We propose (b) *BlockPlanner*, a novel system for generating realistic and diverse (c) 3D city blocks from scratch with valid land use categories. The cores of *BlockPlanner* are: a vectorized city block representation, a graph-based VAE, and a series of losses to prevent geometric violations.

Abstract

City modeling is the foundation for computational urban planning, navigation, and entertainment. In this work, we present the first generative model of city blocks named BlockPlanner, and showcase its ability to synthesize valid city blocks with varying land lots configurations. We propose a novel vectorized city block representation utilizing a ring topology and a two-tier graph to capture the global and local structures of a city block. Each land lot is abstracted into a vector representation covering both its 3D geometry and land use semantics. Such vectorized representation enables us to deploy a lightweight network to capture the underlying distribution of land lots configurations in a city block. To enforce intrinsic spatial constraints of a valid city block, a set of effective loss functions are imposed to shape rational results. We contribute a pilot city block dataset to demonstrate the effectiveness and efficiency of our representation and framework over the state-of-the-art. Notably, our BlockPlanner is also able to edit and manipulate city blocks, enabling several useful applications, e.g.*, topology refinement and footprint generation.*

1. Introduction

The surging demand for city modeling [24, 19] appears in many fields, including urban planning [29, 10, 4], autonomous driving simulation [12], and game design [5, 1, 2], attracting lots of research attention in the last decades. As the fundamental component of urban areas, *city blocks* provide the order and structure to a city, forming the basic unit of a city's urban fabric, illustrated in Fig. 1. However, conventional procedural modeling methods adopt heuristic rules to approximate the subdivision of city blocks [29, 32], which may fail to reflect the realistic and dynamic structures. Consequently, we want to learn city block generation from resourceful urban planning data that captures both the 3D geometry and their land use compositions.

A more relevant problem to ours is indoor floorplan generation. Series of works [27, 15, 36] have recently been inspired by the huge success in rasterized image synthesis [21, 8, 30]. For example, HouseGAN [27] adopts a generative adversarial network (GAN) to produce diverse house layouts given a relation graph. The model predicts a binary mask for each room, indicating its position and size relative to the house. However, these solutions cannot be directly adapted to city block generation for the following reasons.

First of all, the rasterized representation is not expressive for city blocks. As city blocks bear intricate structural details and regularities, representing land lot instances as binary masks significantly impair their geometrical traits, as well as the relative position between land lots. Additionally, once rasterized, small land lots are likely to be dropped

*Equal contribution.

by the generative model as a small probability event. One needs to increase the resolution of rasterized images to fully capture the details and diversity in a city block, as a city block on average accommodates more than 30 land lots, whereas the number of rooms in a house is usually less than 10. Consequently, more training time and computing resources are required. Secondly, the spatial configuration of a city block has unique constraints on the topology and function of land lots. For example, the generated land lots should be accessible from streets and ideally aligned. Furthermore, with much more instances, the searching space for the relation among lots grows exponentially.

To this end, we propose a novel block generation framework, referred to as *BlockPlanner*. Following the nature of city blocks, we propose to represent city blocks in a vectorized manner. We come up with a unified ring topology as the backbone of various land lots arrangements in a block and construct two-tier graphs to represent city block individuals. While the block as a whole serves as the parent node, each land lot is represented as a child node, associated with a set of geometry parameters and geometric attributes, as well as the land use semantics. Their adjacency relations are reflected by the edge matrix. Such representation captures both the global structure of a city block and the local relationships among its land lots, enabling our development of a lightweight model to generate diverse city blocks with valid structural details. Our model is constructed following the scheme of a variational autoencoder (VAE) [22]. A set of losses are carefully designed to penalize various geometric violations for ensuring valid topologies. To facilitate city block generations and more general studies, we construct a new dataset named *NYC-Block Dataset*, collected and organized from the public government resources [6]. Our experiments demonstrate *BlockPlanner*'s capability of producing diverse and realistic city blocks with reasonable land use compositions. We further show that *BlockPlanner* can benefit applications such as city footprint generation and indoor scene layout generation.

In summary, the main contributions of our work are:

- We propose *BlockPlanner*, a system that realizes diverse and valid city block generation, where a novel vectorized two-tier graph representation, a lightweight graph VAE, and a set of losses capable of geometric reasoning are contributed. A pilot city block dataset containing real-world city blocks with rich annotations is alongside collected, which would facilitate future research on city understanding.

- We demonstrate the efficiency and the generalizability of our approach, which sets the foundation for large-scale city modeling with high fidelity in both geometry and functional semantics.

2. Related Work

City Modeling and Urban Representation. Procedural modeling is the early approach for the generation of city layouts, including street networks and 3d building models [29, 26]. While these methods guarantee valid topologies with user-specified attributes and can be deployed to large urban areas, expert efforts are required to design hand-coded rules that accord with certain regulations both visually and semantically. Another line of researches exploits two distinctive attributes of city images: 1) the rich geometric structures inherited in the man-made world [40, 39]; and 2) the underlying functional semantics that are tightly connected with human activities [35, 16]. Both attributes have shown important roles in urban view recognition and understanding, yet how to incorporate them effectively into the generative model counterparts remains an open question.

Generative Models for Structured Layout. Layouts have the nature of being topologically constrained in the global structure, meanwhile exhibiting consistency in local shape attributes. The generation of layouts hence must consider both perspectives. Various graph-based generative models have been proposed [25, 23, 37] to enforce rational structure in 3D shape generation. LayoutVAE [20] adopts a sequential framework and tries to capture the structural logic of a scene by placing elements iteratively. Similarly, Wang *et al*. [33] synthesize indoor scenes by inserting objects one-by-one. Recently, in the context of indoor floorplan generation, multiple attempts have been made to enforce local spatial consistency by converting floorplans into rasterized images [27, 15, 36]. Features such as zero-gap adjacency and wall boundaries between rooms are then reflected in pixel-level, thus can be effectively learned by deep neural networks. Nevertheless, these approaches require either the outline of the floorplan or the connectivity between rooms, which to some extent simplify the challenge by providing hints of structure. Modeling floorplan or layout in rasterized form has certain drawbacks. The rasterized layout is not scaleable nor precise. Future processing is inevitable to make it suitable for computer-aided design.

Inspired by recent success in using vectorized representation for synthesizing images and vector graphics [9, 13, 31], we propose a novel framework called *BlockPlanner* for city block generation.

3. NYC-Block Dataset

To launch city block generation at large scale, a pilot dataset *NYC-Block Dataset* is collected and organized on top of urban planning resources published by city agencies of New York City, US [6]. We sort out records related to valid tax lots and group them by city blocks, resulting in $28,838$ entries in total. A typical city block sample is presented in Fig. 2. By definition, a *city block* is a tract of land

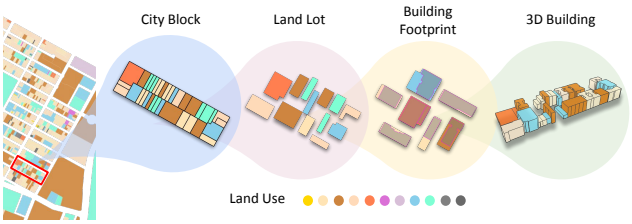

Figure 2: Visualization of a typical city block in *NYC-Block*.

	Land Lots per CityBlock					CityBlocks
	Avg.	Std.	Min.	Med.	Max.	Count
BK	36.35	25.30	1	37	524	7,610
SI	30.69	30.76	1	24	494	4,070
MN	21.71	17.85	1	19	243	1,965
BX	29.59	22.93	1	27	244	3,031
QN	26.68	17.77	1	26	273	12,162
NYC	29.77	23.07	1	27	524	28,838

Table 1: Statistics of *NYC-Block*, listed separately for five boroughs: Brooklyn (BK), Staten Island (SI), Manhattan (MN), Bronx (BX), Queens (QN), and the entire NYC.

bounded on all sides by streets. The spatial configuration and layout of a city block are defined based on its internal arrangement of *land lots*. There is also a height limitation specified for each land lot, where buildings constructed within should not exceed. Apart from geometry, each land lot is associated with its unique *land use* type.

NYC-Block Dataset is large-scale, comprehensive, and high quality, in terms of the following aspects: 1) Wide coverage: it covers all five boroughs across NYC, including both urban and suburban areas, where a great diversity of block shapes and land use configurations are included. Statistics are shown in Tab. 1; 2) Hierarchical structures: each land lot is associated with its parent block, district, and borough, allowing us to conduct a series of merging and splitting operations on different levels, and build connections between the geo-related attributes and the instance-level lot attributes; 3) Rich annotations: along with accurate lot geometries stored as polygons with coordinates of each vertex, each land lot is associated with 93 meta attributes covering from functional uses, land use regularizations, administrative planning, year built and modified, etc. All the information is provided by the official NYC urban planning department, which guarantees the facticity and accuracy in reflecting real city block distributions; 4) Easy generalization ability: the dataset can be easily enriched by linking different databases conditioning on the unique geo-location keys. The consolidated *NYC-Block Dataset* containing detailed geometry and multi-level hierarchy and semantics opens new possibilities for future research on city understanding and cross-modal analysis.

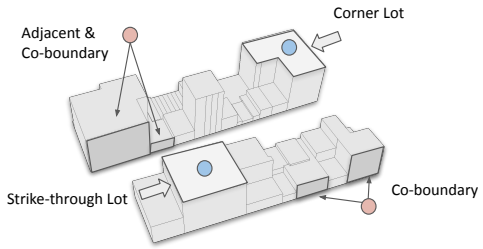

Figure 3: Illustration of two important geometric relationships between lots: Adjacency & Co-boundary, and two special lot types in spatial: Corner lot & Strike-through lot.

4. BlockPlanner

The overview of *BlockPlanner* is shown in Fig. 4. During the inference time, random codes can be sampled from the gaussian distribution and generate diverse and realistic city blocks in an unconditional way. As a city block is usually bounded by two sets of parallel streets, blocks with approximate rectangle shapes are of our main interest. We first introduce how a city block is canonicalized and represented in a graph structure in Sec. 4.1. Then we explain the training paradigm in Sec. 4.2, and elaborate the designed training loss functions with geometric constraints in Sec. 4.3.

4.1. City Block Representation

We aim to seek a representation depicting accurate geometry and land use semantics of a city block, whilst effectively capturing its global structure and the local relationship among its land lots components.

Canonical View. To eliminate the orientation and scale difference caused by terrain, city blocks are firstly transformed to a canonical view. The heading direction is corrected by placing its longer block side to be well-aligned with the horizontal. We normalize the geometry coordinates to $[-1, 1]$, representing the relative displacement in 2D space compared to the centroid of the block. The building heights are normalized in a range of $[0, 1]$ within a city block.

Ring Topology for Global Structure. Rather than treating the city block as an unordered set of land lots, we propose a novel *ring topology* as the backbone of various lot configurations, with lot instances indexed in a unified order. In this way, we circumvent the inefficiency and potential errors brought by the matching process in loss computation; and encourage the model to concentrate more on local connections, which greatly benefits their geometric reasoning.

Specifically, abide by urban planning regulations, land lots in a block are configured to be accessible from at least one of the surrounding streets to guarantee accessibility, forming the intrinsic *co-boundary* relationship in block layout, as illustrated in Fig. 3. Stem from this fact, for each city block, we arrange its land lots into a ring topology and order them from the land lot in the left-upper corner. From the starting lot, we traverse along the four boundaries of

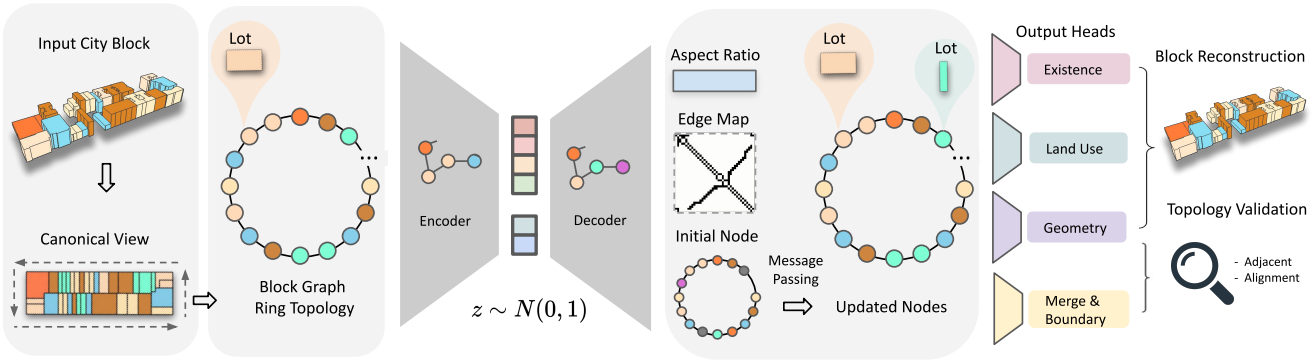

Figure 4: The overall framework of *BlockPlanner*. The model learns to generate city blocks under a VAE training scheme. Each city block is represented by a graph with lots arranged in a ring topology. After encoding the input graph into a 128-d latent code, the decoder first predicts 1) the aspect ratio of the block shape, 2) the edge map indicating adjacent relations, and 3) the initial feature for each lot node. Then the initial features are updated through iterative message passing and output 1) existence of lots, 2) the land use category, 3) the geometry parameters, and 4) the merged and boundary attributes, with four linear heads. These attributes are further used to calculate the reconstruction error and the geometry violations, plus the conventional variational loss.

the block in a counter-clockwise manner, turning the encountered land lots into an ordered list according to their *boundary accessibility* s_b. For a standard rectangle shape block, $s_b \in \{0, 1, 2, 3\}$, representing its four boundaries. Land lots are then assigned to the boundaries they reside on. To preserve continuity, corner lots and strike-through lots that touch more than one side, as shown in Fig. 3, are repetitively counted. The duplicated occurrences on later appeared times are distinguished with a *merging parameter* $s_m \in \{0, 1\}$, indicating whether the lot should be merged.

Block-Lot Representation with Graph. We formulate a city block as a two-tier graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, $\mathcal{V} = \{B, \{L_i\}\}$, where B is the parent node standing for the city block, and each child node L_i denotes a land lot belonging to this city block. Further generalization about this graph structure is discussed in supplementary. Here, each lot L_i is associated with two attributes: the *land lot geometry* g_i and the *land use semantics* s_i. We adopt a 3D bounding box representation for g_i with five parameters $\{x_c, y_c, w, h, n\}$, where (x_c, y_c) is the coordinate of its 2D box centroid, while (w, h, n) denote the size of its 3D building envelope. Here we eliminate the subscript i for notation brevity. The block node B is associated with an *aspect ratio* $r_y = W/H$, where W and H are the width and height of a city block. The edge matrix $\mathcal{E}$ represents the adjacency relations between land lots. We set the edge value $e_{i,j} = 1$ if the lot pair (L_i, L_j) are adjacent in 2D, and 0 otherwise.

4.2. Generate City Blocks with Graph VAE

We adopt a novel framework based on powerful graph neural networks (GNN) and variational autoencoder (VAE) [38, 25, 22] to train *BlockPlanner*. The designs of our graph encoder and decoder are explained in the following. More network illustrations are provided in supplementary.

Encoder. The encoding process starts from the lot level. For a lot node L_i, the geometry and semantic embedding $f_{i,g}, f_{i,s}$ are obtained via two separate encoders:

$$f_{i,g} = e_g(\{x_c, y_c, w, h, n\}), \quad f_{i,s} = e_s(s_l). \tag{1}$$

A one-hot *positional encoding* $f_{i,PE}$ indicating the lot order index i in the ring is concatenated to the initialized feature $f_i^{(0)}$ as $\{f_{i,g}, f_{i,s}, f_{i,PE}\}$. Given the edge matrix $\mathcal{E}$, the initial feature $f_i^{(0)}$ gets updated via T iterations of message passing by averaging over the neighboring lots. At each iteration t, $f_i^{(t)}$ is updated as follows,

$$f_i^{(t)} = \frac{1}{M_i} \sum_{\{e_{i,j}\} \in \mathcal{E}} h^{(t)} \left([f_i^{(t-1)}, f_j^{(t-1)}] \right), \tag{2}$$

where M_i is the number of adjacent lots for L_i, and $h^{(t)}$ encodes the concatenation of $f_i^{(t-1)}$ and $f_j^{(t-1)}$. This module aims to enrich each lot feature with its surrounding lot nodes, as their spatial geometries and functionalities are mutually constrained and influenced. After each iteration t, we summarize the graph embedding $z^{(t)}$ representing the block node feature via max-pooling over the entire lot nodes. The final embedding z for the block node B is obtained at last by concatenating all $z^{(t)}$ obtained from each iteration and pass through the final aggregation layer.

Decoder. On the decoder side, we develop a set of specialized geometric reasoning modules, to bridge the gap between low-level geometry and high-level topological reasoning. Specifically, the block node embedding z is processed by three decoders to predict: the edge matrix, the initial feature for each land lot, and block geometry parameter, *i.e.*, aspect ratio, from which a rough block graph can then be inferred. Formally, they are predicted by:

$$\tilde{\mathcal{P}} = d_{edge}(z), \quad \{\tilde{f}_i^{(0)}\} = d_{lot}(z), \quad \tilde{r}_y = d_r(z), \tag{3}$$

where $\tilde{\mathcal{P}}$ represents how likely the two land lots are connected. An edge with $\tilde{P}(i,j) \geq 0.5$ is treated as existing, and $\tilde{\mathcal{E}} = \tilde{\mathcal{P}} \geq 0.5$. Similar to the encoder, a one-hot position encoding is attached to the initialized decoded features as $\tilde{f}_i^{(0)} = \{d_{lot}(z), f_{i,PE}\}$. We follow Eq. 2 to update $\tilde{f}_i^{(t)}$ iteratively based on the predicted edge matrix $\tilde{\mathcal{E}}$.

The final lot node embedding $\tilde{f}_i = \tilde{f}_i^{(T)}$ is evaluated from multiple geometry regularization aspects. 1) We first decode the geometry of each lot with $(\tilde{x}_c, \tilde{y}_c, \tilde{w}, \tilde{h}, \tilde{n}) = g_{box}(\tilde{f}_i)$. 2) Given the maximal number N of possible lots in a block, we predict the actual existence probability of each lot with $p_i = \sigma(g_{lot}(\tilde{f}_i))$, where σ is a sigmoid function. Lots with $p_i \geq 0.5$ are treated as exists; 3) Meanwhile, the land use semantic is decoded with $\tilde{s}_l = g_{label}(\tilde{f}_i)$; 4) To further enforce the geometry constraints, two additional attributes are predicted $\tilde{s}_m = g_{merge}(\tilde{f}_i)$, and the boundary accessibility $\tilde{s}_b = g_{bound}(\tilde{f}_i)$, as mentioned in Sec. 4.1.

4.3. Loss Function Design

To enforce both geometric and semantic constraints, we design the following loss functions.

1) **Reconstruction loss** $\mathcal{L}_r$ measures the disparity in geometry parameters between the generated block and the ground truth using $\mathcal{L}_1$ loss. Given the unified land lots ordering under the ring topology, the reconstruction loss on the entire block can be computed by directly summing up the losses on its land lots components L_i; plus the difference in block aspect ratio r_y:

$$\mathcal{L}_r = \sum_i ||L_i - \tilde{L}_i||_1 + ||r_y - \tilde{r}_y||_1. \tag{4}$$

2) **Existence loss** $\mathcal{L}_x$ examines both the existence of node and edge, defined as:

$$\mathcal{L}_x = \mathcal{L}_{lot} + \mathcal{L}_{edge} + M_e||\mathcal{E} - \tilde{\mathcal{E}}||_1. \tag{5}$$

As existence is a binary code, binary cross entropy loss is used for both $\mathcal{L}_{lot}$ and $\mathcal{L}_{edge}$. Since the positive pairs in the edge matrix are sparse, we add an additional term by highlighting the ground truth positive edge pairs with a weighted binary mask M_e multiplied to the $\mathcal{L}_1$ loss.

3) **Land use semantics loss** $\mathcal{L}_s$ evaluates the captured semantics. We cast land use category prediction as a classification problem and use the conventional cross entropy loss.

4) **Geometric validation loss** $\mathcal{L}_g$ plays a critical role in our formulation. It consists of three parts:

$$\mathcal{L}_g = \mathcal{L}_{adj} + \mathcal{L}_{bound} + \mathcal{L}_{merge}. \tag{6}$$

$\mathcal{L}_g$ covers both the *topology attribute* loss and their *inconsistency* with the actually decoded geometries. The boundaries of land lots are characterized by $(x_c \pm \frac{w}{2}, y_c \pm \frac{h}{2})$. Specifically, $\mathcal{L}_{adj}$ penalizes when two adjacent land lots do not touch, implemented by pairing land lots according to their orders inside the ring topology and the street boundaries, followed by minimizing the L_1 distances of the colliding edges of these adjacent land lots. $\mathcal{L}_{bound}$ evaluates whether the boundary accessibility is correctly predicted and whether the bounding box edge of a land lot is tightly aligned with the predicted boundary; $\mathcal{L}_{merge}$ is specialized to take care of the corner and strike-through type of lots to reduce potential duplication.

5) **Variational regularization loss** $\mathcal{L}_v$ is added as a conventional technique to obtain a smooth latent space following a standard normal distribution. Our final loss is a weighted combination of the above:

$$\mathcal{L} = w_r\mathcal{L}_r + w_x\mathcal{L}_x + w_s\mathcal{L}_s + w_g\mathcal{L}_g + w_v\mathcal{L}_v. \tag{7}$$

5. Experiments

Implementation Details. Since city blocks in boroughs like Brooklyn and Queens are usually monotone with little variation (see statistics in supplementary), in this work we narrow down to a representative Manhattan subset selected from *NYC-Block*, which is more diverse and complicated, while our methods can be naturally generalized to other areas and larger datasets. Inferred from the statistics in Tab. 1, we set the maximum number of lots N within a block to be 50 to filter out extraordinarily large blocks; and the minimum lot number to be 5 to preserve enough composition complexity, resulting in 637 block samples (15,541 land lots) to conduct experiments on. Detailed network structure and hardware configuration can be found in supplementary.

Evaluation Metrics. We compare the generated city blocks to real distribution with two sets of metrics, which measure the fidelity in both visual quality and structural statistics.

To evaluate visual quality, 1) *Realism* measures the fidelity of rasterized blocks via user rating. The final score ranges from 0 to 100, with higher score denotes higher fidelity. User study details can be found in supplementary. 2) *FID* score [14, 27, 28] is adopted to measure the diversity of the rasterized 2D samples.

To evaluate layout validity, we additionally compare a set of *land use statistics* between the empirical distributions of real and fake samples, where 3) $\boldsymbol{S_c}$ stands for the number of land lots per-block; 4) $\boldsymbol{S_l}$ is the empirical count of land lots over different land use types; and 5) $\boldsymbol{S_t}$ is the empirical *transition probability* of the land use category between one-step neighboring lots within the same block. We report the L_1 distance between the discretized distributions for evaluation. Note that, we use these statistical metrics only in our ablation studies, given that the overall visual quality of the generated samples is reasonable enough.

Comparison Methods. Since we are the first to study end-to-end city block generation, our proposed *BlockPlanner* is compared with two general graph-based layout/structure

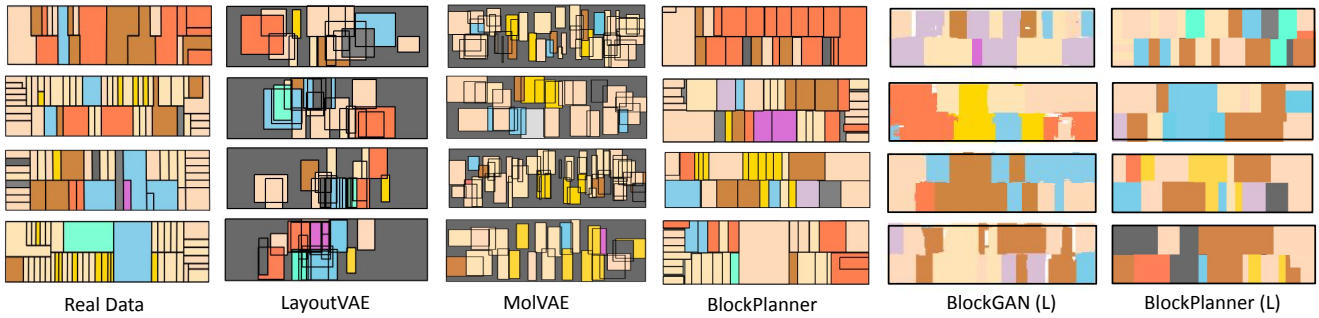

Figure 5: Qualitative evaluations. Column 2-3: LayoutVAE and MolVAE fail to capture valid block topology, while our *BlockPlanner* efficiently generates diverse and valid block layouts with reasonable land use categories. Column 5-6: marker (L) indicates land use level view. Though pixel-based BlockGAN generates sensible structure for large lots, the artifacts such as the fuzzy boundary are significant. In contrast, our vectorized representation captures much more accurate geometry even in small areas. (Dimgray color indicates the unfilled regions.)

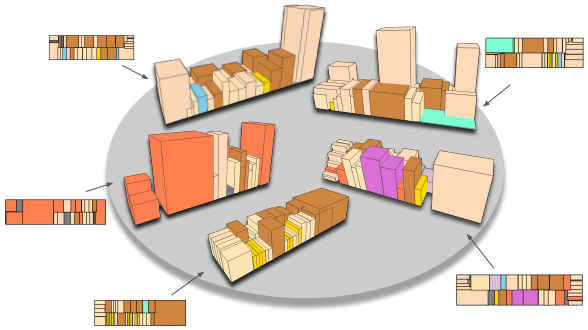

Figure 6: 3D City blocks generated by *BlockPlanner* by extruding the height dimension with lot attribute n.

Settings	# FLOPS/M (↓)	Realism (↑)	FID (↓)
LayoutVAE	11	21.5	205.9
MolVAE	18	33.8	146.6
BlockGAN	805,297	63.4	133.1
BlockPlanner (Ours)	373	**95.6**	**36.9**

Table 2: Quantitative evaluations. FID and Realism measure visual quality; FLOPS measures computing efficiency.

generation frameworks, namely LayoutVAE [20] and MolGAN [11]; and a SOTA method in indoor floorplan generation, named HouseGAN [27]. We make minimal input-level adjustments to fit the city block generation scenario. The hyperparameters are obtained via grid-search with optimal performance. We implement LayoutVAE following the original paper [20]; substitute MolGAN with its VAE counterpart that appears to be more stable during training; and transform HouseGAN into an unconditional generative framework. To be compatible with HouseGAN's input, we rasterize our city block data in 2D and feed it into the original HouseGAN model. We treat each land use assembly as an instance and output a binary mask for each land use, denoted as BlockGAN. Detailed implementations can be found in supplementary.

5.1. Benchmarking Results

We use *Realism* and *FID* to measure the visual quality of the generated blocks and compare the model efficiency using *FLOPS* computed at inference time, as a lightweight generative model is always preferred in large-scale city modeling. The quantitative comparison is provided in Tab. 2 and the qualitative results are shown in Fig. 5.

Comparison with Graph-based Methods. The block layouts generated by LayoutVAE are drastically different from real data. Land lots majorly concentrate around the center of the block with a high overlapping ratio. It appears that LayoutVAE fails to grasp the global structure of city blocks. This can be ascribed to its sequential generation manner, which enforces the model to focus on local patterns only. The generated results from MolVAE are more structured compared to LayoutVAE. However, it fails to capture the detailed relationship between pairs of land lots, and the sense of alignment is weak. This is because the design of MolVAE lacks spatial reasoning capability. In its initially applied scenario, *i.e.* molecule synthesis, only atom types and bonds between atoms are of its primary concern, while city block requires much stronger geometry representation in order to synthesize valid topology.

Comparison with Pixel-based Methods. BlockGAN produces more reasonable configurations compared to the aforementioned methods. Although it can output rational global block structure, the generated mask at the instance level is fuzzy with inaccurate geometry. Different from an indoor floorplan, a land lot usually occupies a small tract and has delicate 2D geometry. The power of rasterized representation is hence limited in capturing all the details. Despite the impressive results on the indoor floorplan, it is less practical for our scenario. A workaround is to increase the input resolution, but with the price of using more computing resources and training time. On the other hand, benefit from the vectorized canonical graph representation and geometry reasoning constraints, *BlockPlanner* is able to capture both the block-level structure and lot-level geometry with

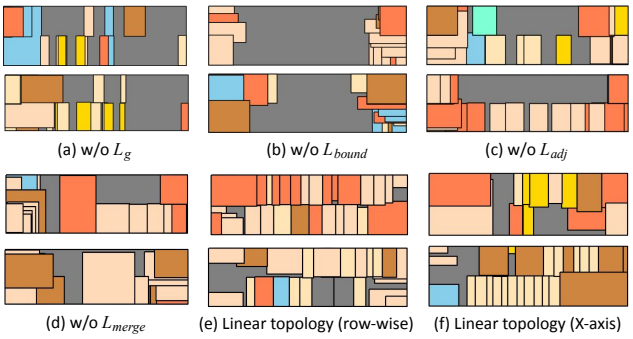

Figure 7: Different effects of visual quality degradation by ablating on different design modules, illustrated with the typical artifacts of each setting.

Ablation Setting		FID (↓)	S_l (↓)	S_t (↓)	S_c (↓)
$\mathcal{L}_g$ Losses	-	92.8	0.29	0.74	0.49
	+ $\mathcal{L}_{adj}$	64.9	0.34	0.64	0.53
	+ $\mathcal{L}_{adj}$ + $\mathcal{L}_{bound}$	55.5	0.33	0.64	0.44
	+ $\mathcal{L}_{adj}$ + $\mathcal{L}_{bound}$ + $\mathcal{L}_{merge}$ ($\star$)	**36.9**	**0.18**	**0.46**	**0.27**
Message Pass	0	50.9	0.75	0.68	0.42
	1	47.3	0.62	0.63	0.43
	2	41.6	0.26	0.53	0.38
	3 ($\star$)	**36.9**	**0.18**	**0.46**	**0.27**
Topology	Row-wise	47.3	0.12	**0.43**	0.31
	X-axis	44.8	**0.11**	0.53	0.62
	Ring ($\star$)	**36.9**	0.18	0.46	**0.27**
Position Enc	w/o PE	47.2	0.41	0.67	0.45
	w/ PE ($\star$)	**36.9**	**0.18**	**0.46**	**0.27**

Table 3: Quantitative comparisons on ablating: (a) $\mathcal{L}_g$ items; (b) Number of iterations in message passing; (c) Adopting alternative topology; (d) Using position encoding on encoder and decoder. We use ($\star$) to indicate full model.

reasonable computing resources. Quantitatively, it outperforms BlockGAN in both *FID* and *Realism* with $\sim 1000\times$ lower *FLOPS*. To further demonstrate the superior of *BlockPlanner*, we provide the 3D extruded geometry of the generated city blocks in Fig. 6, which exhibits diversity and fidelity in both topology and the land use semantics.

5.2. Ablation Studies

Effect of Geometry Constraints. The role of each geometry constraint $\mathcal{L}_{adj}$, $\mathcal{L}_{bound}$ and $\mathcal{L}_{merge}$ is individually studied. Fig. 7 (a)-(d) depict the effect of each loss term. $\mathcal{L}_{bound}$ helps spread land lots to four boundaries and enforce their alignment; $\mathcal{L}_{adj}$ decreases gaps between land lots; $\mathcal{L}_{merge}$ effectively prevents overlaying at corners and helps predict correct geometry for strike-through and corner land lots. Tab. 3 (a) offers quantitative results where the best performance is achieved when all constraints are imposed.

Influence of Message Passing. We vary the number of iterations performing message passing in our model. Results are shown in Tab. 3 (b). It can be noticed that increasing the number of iterations from 0 to 3 leads to better results, implying that a deeper fusion of neighboring information benefits the learning process.

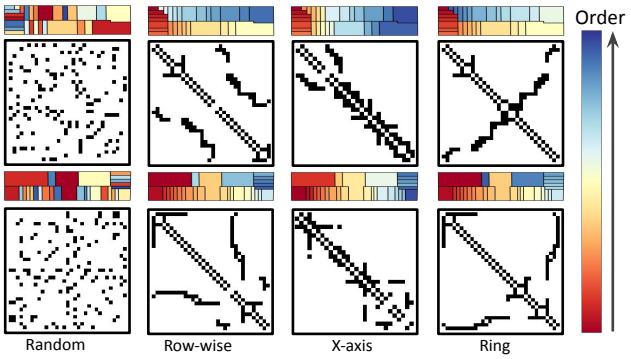

Figure 8: Edge matrices from different global topologies. From left to right, *Random* (no pre-defined topology) gives patternless edge matrices; *Row-wise* and *X-axis* (linear topology) lead to discontinuity due to jumps between lots; *Ring* exhibits regular and continuous edge matrices.

Choice of Global Topology. Recall that our novel city block representation fits the global block structure into a ring topology. Such design reduces data variance compared to those constructed without global topology as illustrated in Fig. 8, which significantly eases the learning process. We also show that the ring topology is superior to linear topologies. Fig. 8 provides two examples where land lots are arranged row-wise and along X-axis respectively. One can observe that the according edge matrices exhibit discontinuity due to jumps between lots, which in turn lead to artifacts in generated samples as shown in Fig. 7 (e)-(f). Quantitative results are provided in Tab. 3 (c). We further study the effect of position encoding imposed on land lots under the ring topology. Tab. 3 (d) shows that both the visual quality and semantics validity improve when position encoding is deployed, where the global structure defined by ring topology is further strengthened.

6. Applications

BlockPlanner supports a variety of geometry editing and manipulation tasks benefiting from its vectorized representation. Meanwhile, it can also be generalized to other layout generation domains. For more discussion and implementation details, we refer the readers to the supplementary.

6.1. Editing and Manipulation

Topology Refinement. *BlockPlanner* allows us to flexibly and accurately control the output block shape and land lots configuration, for further editing or processing, in order to enhance the final visual quality. For example, we can adopt an optimization process to predict a minimal editing effort to obtain a zero-gap layout. Specifically, we learn a set of refinement parameters X that control the translation and scaling magnitude of each lot geometry. The optimized result can be obtained by minimizing the objective $D_{gap}(r(\mathcal{G}))+\lambda|X|_2$, where $r(\cdot)$ indicates the rendered layout, and $D_{gap}(\cdot)$ is the accumulative gap areas. We use

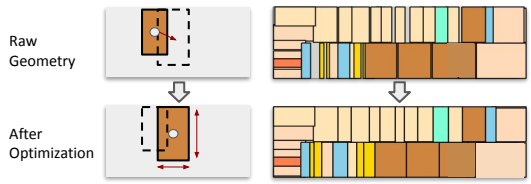

Figure 9: For cases where the generated block has gaps in between, we can effectively correct them by running the post-optimization step to minimize such artifacts.

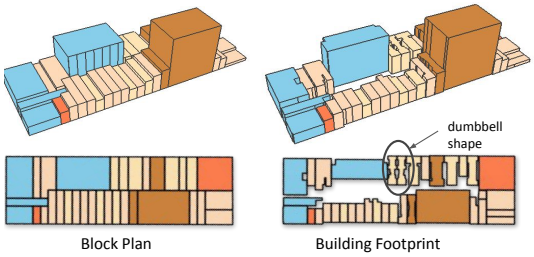

Figure 10: From block plan to building footprint. A strong correlation can be observed among footprint shapes within neighboring lots, conditioned on the land use category, such as the classic "dumbbell" shape for the old law tenement.

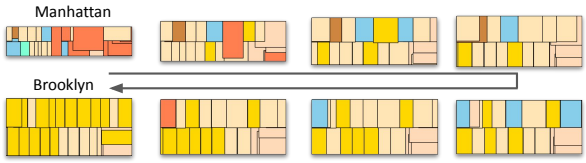

Figure 11: Network Interpolation [34] between two models separately trained on Manhattan (MN) and Brooklyn (BK) subsets (ordered clockwise). From MN to BK, block length along Y-axis gradually increases with land lots shape becoming more homogeneous. Functionally, residential lots gradually dominate the block.

$\lambda|X|_2$ to constrain the magnitude of changes, reflecting our desire to ask each lot to maintain its original geometry as much as possible. See Fig. 9 for example cases.

Footprint Generation. With *BlockPlanner*, large-scale city blocks can be easily generated. We showcase one-step further to synthesize plausible building footprints upon the generated block plans, *i.e.*, to generate 3D building models reside in each land lot. We formulate this task as a classical image-to-image translation problem conditioned on the generated city blocks. See supplementary for our adapted pix2pix [17] implementation.

From Manhattan to Brooklyn. Can the learned latent code distribution reveal the characteristics of a local area, so that it can also serve as a good indicator to study urban similarity and differences? Driven by this curiosity, we finetune a model on a filtered subset with $2,804$ Brooklyn blocks, which is pre-trained on the Manhattan subset. The linearly interpolated results using DNI [34] are analyzed in Fig. 11. Interestingly, the smooth transition reflects the underlying block patterns exhibiting in the two boroughs, revealing a tight correlation between the machine-identified city block configurations and the urban social factors.

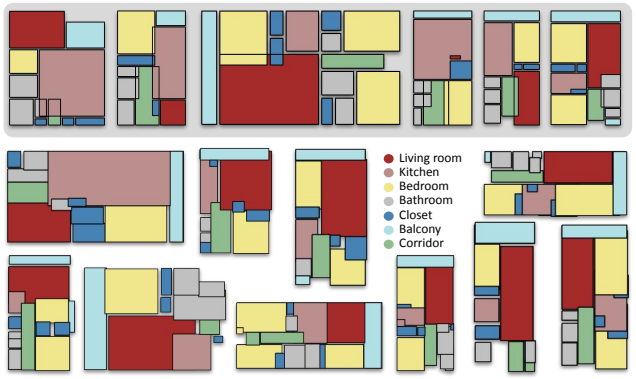

Figure 12: *Top*: Real floorplans from HouseGAN dataset. *Bottom*: Ours generated floorplans, with diverse configurations and realistic looking.

6.2. Extension to Indoor Scenes

With little modification, our vectorized representation for city blocks can be adapted to other layout generation domains. Take the indoor floorplan as an example, for the rooms and furniture that do not touch any boundary, we associate them with an additional void boundary dimension under the ring topology. We train *BlockPlanner* on a subset of HouseGAN [27] dataset, where we select houses with at least 10 rooms to showcase the capability of our framework in generating layouts with a large number of nodes. While in their original experimental setting, most houses have room numbers less than 10. See Fig. 12 for both ground truth samples and our generated floorplans.

7. Conclusion

In this work, we propose *BlockPlanner* to tackle the problem of city block generation, accompanied by a newly collected *NYC-Block Dataset*. *BlockPlanner* models city blocks using a novel and effective vectorized representation and trains a lightweight graph-based generative model under a VAE scheme, where a set of losses capable of geometrical reasoning are designed to enforce the intrinsic constraints on city blocks. We show that the generated city blocks are realistic in visual and valid in configuration, and allow future manipulation at the lot level. With such merits, *BlockPlanner* opens a new direction for large-scale city modeling in an end-to-end manner.

Acknowledgment This work has been supported by the Centre of Perceptual and Interactive Intelligence, the General Research Fund (GRF) of Hong Kong (No. 14205719), Theme-based Research Scheme 2020/21 (No. T41-603/20-R), NTU NAP, and under the RIE2020 Industry Alignment Fund–Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in-kind contribution from the industry partner(s).

References

[1] Cities: Skylines on steam. https://store.steampowered.com/app/255710/Cities_Skylines/. 1

[2] Cyberpunk 2077. https://www.cyberpunk.net/. 1

[3] Doitt. https://www1.nyc.gov/site/doitt/residents/gis-2d-data.page. 12

[4] Esri: Cityengine. https://www.esri.com/en-us/arcgis/products/esri-cityengine. 1

[5] Gta5: Grand theft auto v. https://www.rockstargames.com/V/. 1

[6] Pluto. https://www1.nyc.gov/site/planning/index.page. 2

[7] Geoff Boeing. Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems*, 65:126–139, 2017. 11

[8] A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis. *ArXiv*, abs/1809.11096, 2019. 1

[9] Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. Deepsvg: A hierarchical generative network for vector graphics animation, 2020. 2

[10] Hang Chu, Daiqing Li, David Acuna, Amlan Kar, Maria Shugrina, Xinkai Wei, Ming-Yu Liu, Antonio Torralba, and Sanja Fidler. Neural turtle graphics for modeling city road layouts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4522–4530, 2019. 1

[11] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018. 6, 13

[12] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017. 1

[13] David R Ha and D. Eck. A neural representation of sketch drawings. *ArXiv*, abs/1704.03477, 2018. 2

[14] 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. In *Advances in Neural Information Processing Systems*, volume 30, 2017. 5, 14

[15] Ruizhen Hu, Zeyu Huang, Yuhan Tang, Oliver Van Kaick, Hao Zhang, and Hui Huang. Graph2plan: Learning floorplan generation from layout graphs. *ACM Trans. Graph.*, 39(4), July 2020. 1, 2

[16] Huaiyi Huang, Yuqi Zhang, Qingqiu Huang, Zhengkui Guo, Ziwei Liu, and Dahua Lin. Placepedia: Comprehensive place understanding with multi-faceted annotations. In *European Conference on Computer Vision*, pages 85–103. Springer, 2020. 2

[17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017. 8, 15

[18] Allan B Jacobs et al. Great streets. Technical report, University of California Transportation Center, 1993. 11

[19] Jane Jacobs. *The death and life of great American cities*. Vintage, 2016. 1

[20] Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. Layoutvae: Stochastic scene layout generation from a label set. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9895–9904, 2019. 2, 6, 13

[21] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *Proc. CVPR*, 2020. 1

[22] Diederik P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014. 2, 4

[23] Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. Grass: Generative recursive autoencoders for shape structures. *ACM Trans. Graph.*, 36(4), July 2017. 2

[24] Kevin Lynch. *The image of the city*, volume 11. MIT press, 1960. 1

[25] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas J Guibas. Structurenet: Hierarchical graph networks for 3d shape generation. *arXiv preprint arXiv:1908.00575*, 2019. 2, 4, 13

[26] 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. 2

[27] 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 *European Conference on Computer Vision*, pages 162–177. Springer, 2020. 1, 2, 5, 6, 8, 13, 14

[28] Wamiq Para, Paul Guerrero, Tom Kelly, Leonidas Guibas, and Peter Wonka. Generative layout modeling using constraint graphs. *arXiv preprint arXiv:2011.13417*, 2020. 5, 14

[29] Yoav IH Parish and Pascal Müller. Procedural modeling of cities. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 301–308, 2001. 1, 2

[30] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1

[31] Leo Sampaio Ferraz Ribeiro, Tu Bui, J. Collomosse, and Moacir P. Ponti. Sketchformer: Transformer-based representation for sketched structure. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14141–14150, 2020. 2

[32] 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

[33] Kai Wang, Manolis Savva, Angel X. Chang, and Daniel Ritchie. Deep convolutional priors for indoor scene synthesis. *ACM Trans. Graph.*, 37(4), July 2018. 2

[34] Xintao Wang, Ke Yu, Chao Dong, Xiaoou Tang, and Chen Change Loy. Deep network interpolation for continuous imagery effect transition. In *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1692–1701, 2019. 8, 15

[35] Zhecheng Wang, Haoyuan Li, and Ram Rajagopal. Urban2vec: Incorporating street view imagery and pois for multi-modal urban neighborhood embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1013–1020, 2020. 2

[36] Wenming Wu, Xiao-Ming Fu, Rui Tang, Yuhan Wang, Yu-Hao Qi, and Ligang Liu. Data-driven interior plan generation for residential buildings. *ACM Transactions on Graphics (TOG)*, 38(6):1–12, 2019. 1, 2

[37] Zhijie Wu, Xiang Wang, Di Lin, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Sagnet: Structure-aware generative network for 3d-shape modeling. *ACM Trans. Graph.*, 38(4), July 2019. 2

[38] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020. 4

[39] Yichao Zhou, Jingwei Huang, Xili Dai, Linjie Luo, Zhili Chen, and Yi Ma. Holicity: A city-scale data platform for learning holistic 3d structures. *arXiv preprint arXiv:2008.03286*, 2020. 2

[40] Yichao Zhou, Haozhi Qi, and Yi Ma. End-to-end wireframe parsing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 962–971, 2019. 2