International Journal of Computer Vision (2024) 132:3408–3427
https://doi.org/10.1007/s11263-024-02030-w

UrbanEvolver: Function-Aware Urban Layout Regeneration

Yiming Qin[1,3] · Nanxuan Zhao[2] · Jiale Yang[1] · Siyuan Pan[1] · Bin Sheng[1] · Rynson W. H. Lau[3]

Received: 22 August 2022 / Accepted: 9 February 2024 / Published online: 19 March 2024

Abstract

Urban regeneration is an important strategy for land redevelopment, to address the urban decay in cities. Among many tasks, urban layout is the foundation for urban regeneration. In this paper, we target a new task called *function-aware urban layout regeneration*, and propose *UrbanEvolver*, a function-aware deep generative model for the task. Given a target region to be regenerated, our model outputs a regenerated urban layout (*i.e.*, roads and buildings) for the target region by considering the function (*i.e.*, land use type) of the target region and its surrounding context (*i.e.*, the functions and urban layouts of the surrounding regions). UrbanEvolver first extracts implicit regeneration rules from the target function and the surrounding context by encoding them separately in different scales through the function-layout adaptive (FA) blocks, and then constrains the regenerated urban layout based on the learned regeneration rules. To enforce the regenerated layout to be valid and to follow the road structure, we design a set of losses covering both pixel-level and geometry-level constraints. To train our model, we collect a large-scale urban layout dataset covering more than 147 K regions under 1300 km^2 with rich annotations, including functions, region shapes, urban road layouts, and urban building layouts. We conduct extensive experiments to show that our model outperforms the baseline methods in generating practical and function-aware urban layouts based on the given target function and surrounding context.

Keywords Urban regeneration · Urban layout regeneration · Function-aware generative model

Communicated by Arun Mallya.

Yiming Qin, Nanxuan Zhao have contributed equally to this work.

Corresponding author: Bin Sheng. Rynson Lau and Bin Sheng lead this project.

✉ Bin Sheng
shengbin@sjtu.edu.cn

Yiming Qin
yiming_qin@sjtu.edu.cn

Nanxuan Zhao
nanxuanzhao@gmail.com

Jiale Yang
yangjiale@sjtu.edu.cn

Siyuan Pan
pansiyuan@sjtu.edu.cn

Rynson W. H. Lau
Rynson.Lau@cityu.edu.hk

1 Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China

2 Department of Computer Science, University of Bath, Bath, UK

3 Department of Computer Science, City University of Hong Kong, Kowloon, HKSAR, China

1 Introduction

Decaying and underused urban regions in a city may damage the image, liveability and productivity of the city (Amirtahmasebi et al., 2016). To address this problem, a complex urban regeneration plan is typically needed, of which coming out with an urban layout is one of the essential tasks. An urban layout basically contains the shape and position of roads and buildings. Given a target region for layout regeneration, there are many factors that can influence the result (Amirtahmasebi et al., 2016). Among them, the target function (*i.e.*, the land use type) and the surrounding context (*i.e.*, the functions, urban layouts and region shapes of the surrounding regions) play the key roles. The target function characterizes the layout of a region (Groenewegen et al., 2009). For example, an industrial region typically has a more regular and sparse layout, while a residential region tends to have an irregular and dense layout. The surrounding regions also influence the layout regeneration, especially near the boundary (Parish & Müller, 2001; Groenewegen et al., 2009). For example, the boundary of the target region tends to have dense and low-

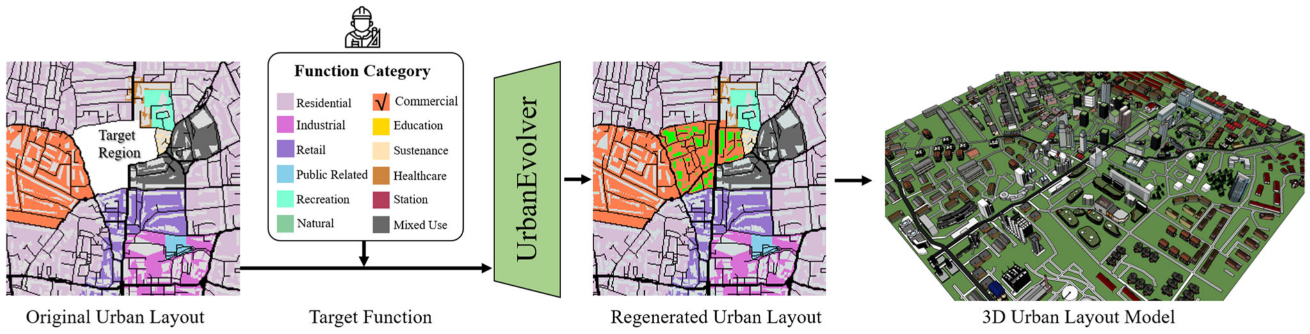

Fig. 1 Urban layout regeneration process. Given a target region with its function and surrounding context (where the color of each surrounding region indicates its function category) as shown on the left, our function-aware urban layout regenerator, *UrbanEvolver*, automatically regenerates the urban layout (*i.e.*, road and building layouts) of the target region as shown in the middle, which matches the user-specified function and the surrounding regions. 3D models can be further added to the target region as shown on the right

class flats if it is next to an industrial region, but spare houses if it is next to a park. However, urban layout regeneration is a tedious and time-consuming task, even for experienced designers. In this work, we aim to ease the burden by taking the first step to learn the function-aware layout regeneration from data.

Traditional works (Parish and Müller (2001); Weber et al. (2009); Lipp et al. (2011)) generate urban layouts in vector format through hand-crafted rules, which have limited representation power. Recent works (Belli & Kipf, 2019; Chu et al., 2019; Mi et al., 2021) adopt data-driven methods for generating urban road layouts in vector format. However, because of the surging computational resources for modeling the relationship among road nodes, these methods can only generate urban road layouts based on limited local statistics. It becomes even more difficult to adapt to model the context (*i.e.*, function and the surrounding region) in our task.

To this end, we propose *UrbanEvolver*, a function-aware urban layout regeneration method. Given a target region to be regenerated, *UrbanEvolver* regenerates the layout of the target region conditioned on the target function and the surrounding context, as shown in Fig. 1. It first implicitly learns the regeneration rules (*e.g.*, the characteristics of different target functions) from the target function and the surrounding context. To achieve this, our model first extracts features from the functional map and the surrounding context and then fuses these features through our novel function-layout adaptive (FA) block. However, it is difficult to learn the regeneration rules directly from the natural vector data which are represented with an adjacency matrix of thousands of dimensions. We build pixel-level data by rendering the region shapes/functions as a functional map and the urban layouts as an urban layout map to facilitate efficient learning, while leaving the connectivity information of the target region in vector format to facilitate accurate learning. Unlike the functional map in (Ovsjanikov et al., 2012), our functional map is a bird's eye view land-use (function) planning map that indicates the functions of the target region and surrounding regions. Hence, we regenerate the urban layout of the target region in a hybrid way by utilizing both vector and pixel data for balancing between computational efficiency and accurate supervision. Through training, our model learns how to extract effective fused features, i.e., implicit representation, and map them to urban layouts.

To supervise the model training, we design a set of losses, including both geometry-level and pixel-level constraints, taking advantage of knowledge from the two domains. While the geometry-level losses help constrain the regenerated results based on road networks and junctions, the pixel-level losses aim at increasing the realism and diversity of the regenerated layouts.

To facilitate learning and evaluation, we have collected a large-scale urban layout dataset that covers more than 147K regions from the Open Street Map (OSM) (OpenStreetMap contributors, 2017). Our dataset has rich annotations, including region shapes and functions, urban road layouts, and urban building layouts. We evaluate our method on our dataset for urban layout regeneration, based on visual and structural quality. Results show that our proposed model can regenerate a valid and function-aware urban layout conditioned on the target function and surrounding context. Our method also outperforms the baseline methods in regenerating urban layouts. Our main contributions are as follows:

- We make the first effort to study the function-aware urban layout regeneration task, and propose an end-to-end function-aware deep model, UrbanEvoler, for urban layout regeneration.
- We propose a function-layout adaptive (FA) block to extract implicit urban layout regeneration rules, and a set of pixel-level and geometry-level loss functions to help ensure the generation of valid urban layouts while minimizing the computational costs.

- We collect a large-scale urban layout dataset covering 147K regions of 1300 km^2 with rich annotations, including region shapes and functions, urban road layouts, and urban building layouts.
- We demonstrate the effectiveness of our method on urban layout regeneration through extensive experiments. We show that our model can capture the urban layout characteristics of different target functions. We also show the practical usage of our model via several applications.

2 Related Work

2.1 Urban Layout Design

Urban layout is important to the design of a city, and has attracted a lot of research. Earlier works mainly considered city modeling using traditional procedural modeling techniques, while recent works focused on urban road layout generation with deep learning techniques. We discuss in detail these two directions of work.

Traditional City Modeling. Parish and Müller (2001) proposed a city generation method based on the L-system. It generated urban layouts from scratch and procedurally created them based on hand-crafted rules. Their method was refined by Weber et al. (2009), which procedurally generated a city with realistic geometric configurations such as parcel boundaries, street widths and building footprints. On the other hand, Lechner et al. (2006) procedurally generated the distribution of land uses, placement of buildings and roads of the city based on an agent-based method when given a terrain description for the whole city. Lipp et al. (2011) proposed a method for editing and merging urban layouts, using both procedural modeling and manual editing. This allows intuitive manipulation, including drag, drop, translation and rotation, on urban layouts. Given a region and design elements such as major roads, lakes and parks, Yang et al. (2013) utilized hierarchical domain splitting to generate street layouts and parcel layouts.

All these existing city generation methods are mainly based on procedural modeling, and generate urban layouts from scratch without taking into account the surrounding context. Procedural modeling methods are limited to hand-crafted rules, which are inflexible and less expressive, requiring a lot of human laboring efforts. Unlike city generation, urban layout regeneration aims to regenerate a new urban layout for a target region, which is surrounded by other existing regions with different/same functional purposes. The new layout for the target region should be constrained by the target function and the surrounding context, rather than just ensuring the validity within the target region only. To address this new task, we propose a deep learning based approach in this work to learn the complex regeneration rules from the target function and the surrounding context, and build a function-aware regenerative model to regenerate the urban layout of the target region based on the learned regeneration rules.

Urban Road Layout Generation. Chen et al. (2008) utilized a tensor field approach to generate the whole street network. However, it required expertise to manually create a tensor field in order to obtain a desired urban road layout. Aliaga et al. (2008) generated a complete street network conditioned on examples of aerial-view imagery. They utilized a random walk based algorithm to generate the street network according to the attributes of the intersection points. Groenewegen et al. (2009) procedurally generated a street network based on high-level user inputs, such as city size, location and historic background. Nishida et al. (2016) also proposed an example-driven procedural urban road generation method. It generated urban road layouts by merging patches extracted from examples. An edge and point merging method was used to ensure connectivity among the patches.

In recent years, deep learning based methods have been introduced to urban road layout design. StreetGAN (Hartmann et al., 2017) is an example-based road layout generation method. It synthesized urban road layouts by reproducing the style of the reference patches from scratch. However, this method has difficulties in generating coherent global patterns and may not produce valid road structures. (Chu et al., 2019) treated an urban road layout as a graph, in which a node represents a junction and an edge represents a road. However, their method generated road layouts based on local statistics and city style, rather than huge and complex geographic data. RoadNetGAN (Owaki & Machida, 2020) used GANs to generate a road network based on an input real urban road layout.

While the above methods focus on generating urban road layouts, most of the deep learning-based methods focus more on generating style-based urban road layouts, such as London-style, New York-style or Beijing-style, or local statistics. Unlike these methods, our deep generative model regenerates urban layouts including urban road and building layouts in an end-to-end manner. Our regenerated results are constrained by the target function and the surrounding context, rather than local heuristic connectivity.

2.2 General Layout Design

In layout design, different conditions are added to the design process in order to meet different layout design requirements. We divide layout design into 2D and 3D layout design tasks.

2D Layout Design. There are many tasks in the domain, and one of the most important tasks is graphic design (including documents, posters and webpages) layout generation. González-Morcillo et al. (2010) utilized evolutionary computation and fuzzy logic to maximize the graphic design quality through cost functions and design principles and constraints. O'Donovan et al. (2014) proposed a single-page graphic design method based on nonlinear inverse optimization learned from example layouts. Yang et al. (2016) formulated visual-textual presentation typography as an energy optimization problem in perception and semantics. Pang et al. (2016) rearranged the Web elements on an input webpage according to the user-specified element reading order. LayoutGAN (Li et al., 2019) arranged the graphic elements to a good quality layout conditioned on semantic and spatial relations. Zheng et al. (2019) proposed the first multi-modal interactive graphic layout generation model conditioned on both visual and textual contents. Based on Zheng et al. (2019), Ueno and Satoh (2021) produced continuous and gradual magazine layout style changes conditioned on two given layout styles and an interpolation coefficient. Vinci (Guo et al., 2021a) produced a poster layout given a product image and taglines.

There are also other layout design tasks. For textural layout, based on a visual saliency map of a natural image, SmartText (Zhang et al., 2020) generated a textual layout over the natural image. For photo layout, Atkins (2008) arranged photos on a rectangular canvas based on designed criteria. Ryu et al. (2010) placed photos in the grid based on temporal and spatial information. For image scene layout, Lee et al. (2018) proposed a model to place a new object instance with a valid layout by considering the scene context of the input semantic map. Qiao et al. (2019) completed an image given only a few foreground objects. Given a labeled set of objects, LayoutVAE (Jyothi et al., 2019) predicted the locations and sizes of the objects to produce a natural image scene layout. Hudson and Zitnick (2021) synthesized images through a sequential process that contains two stages: a planning phase and an execution phase.

3D Layout Design. For house layout, Merrell et al. (2010) utilized a Bayesian network trained on real-world data to design floor plans for residential buildings. Merrell et al. (2011) incorporated the layout guidelines as terms in a density function and proposed a hardware-accelerate Monte Carlo sampler to generate the furniture layout based on the density function. Fisher et al. (2011) represented indoor scenes as graphs, and then defined a kernel between relationship graphs to compare common virtual substructures and capture the similarity between their corresponding scenes. Fisher et al. (2012) introduced Bayesian networks and Gaussian mixtures to build a probabilistic model for 3D object arrangements from examples. Yu et al. (2011) encoded the relationships of furniture objects as a cost function and optimized it using a Metropolis-Hastings state search step. Fisher et al. (2015) synthesized an indoor scene layout conditioned on a coarse geometric scene representation and human activities. Henderson et al. (2017) generated furniture layouts by exploiting the learned joint (co-occurrence) statistics from a database, using the room shape, size and object placement as conditions. HouseGAN (Nauata et al., 2020) utilized a graph as conditions, where nodes represent room types and edges represent room relationships, to generate house layouts. (Nauata et al., 2021) improved on (Nauata et al., 2020) and introduced a new condition, 2D segmentation mask, to enable iterative design refinement. Graph2Plan (Hu et al., 2020) first retrieved a house layout from the database based on the user-specified room counts and other constraints, and then generated different types of house layouts conditioned on the building boundary. For building layout, Building-GAN (Chang et al., 2021) generated multi-story building layouts conditioned on a program graph. Gupta et al. (2021) proposed LayoutTransfomer for diverse domains layout generation conditioned on the learned contextual relationships, such as 2D graphic layout and 3D objects layout.

For other tasks, Campen et al. (2012) constructed quad layouts on manifold surfaces based on careful construction of the layout graph's combinatorial dual. Umetani et al. (2012) proposed an interactive design framework to provide active guidance for designing geometrically and physically valid models. Panozzo et al. (2013) optimized the force layouts both geometrically and topologically to find a self-supported structure. Ripon et al. (2013) solved the multi-objective facility layout problem using the variable neighborhood search. Peng et al. (2014) tiled a domain with a set of deformable templates using a two-step layout algorithm including a discrete step and a continuous step. Chen et al. (2015) utilized a multi-objective genetic algorithm to optimize the wind farm layout to increase the power output.

Unlike all these layout design works, we focus on urban layout regeneration in this paper, and condition our regenerated urban layout on the given target function and surrounding context.

3 Our Dataset

To learn the urban layout regeneration task, a well-designed urban layout dataset is needed. Although there are some publicly available datasets (Amazon Web Services, 2016; Belli & Kipf, 2019) for urban layout design, they only contain urban road layout information without other annotations. However, urban layout regeneration requires not only urban road layout but also functions (*i.e.*, land use types), region shapes and urban building layout, which existing datasets do not provide. Thus, we collect a large-scale urban layout dataset with rich annotations to facilitate our task.

Initial Collection. We collect data from Open Street Map (OSM) (OpenStreetMap contributors, 2017), which contains diverse annotations to build our dataset. As an open-sourced website, the annotations are labeled by contributors voluntarily. Thus, OSM often contains incomplete markers in underpopulated regions. In order to obtain complete and rich annotations, we collect our dataset mainly from the Greater London area, using a scale of 1:200. We utilize scripts and OSM's application programming interface (API) to obtain geographical data.

We select 12 functions from OSM: commercial, industrial, residential, retail, education, sustenance, healthcare, public related, recreational, natural, station, and mixed-use. These 12 functions are commonly used in cities (not countryside) according to (OpenStreetMap, contributors, 2017; Lenormand et al., 2015). A function indicates the purpose for which an area of land is being used, and different functions can be classified based on the layout image, road density, building density, 4-way crossing proportion and connectivity index (Alhalawani et al., 2014). Each function may be assigned to different regions of a city, while each region represents a connected area covered by a single function.

In the initial raw format, the road layout, building layout and region shapes are all represented by a sequence of latitude and longitude coordinates.

Pre-processing. From the raw data, we note that if a wide road crosses a single functional region, the region can be annotated as multiple regions. To remove this problem, we employ a polygon merging method (Žalik, 2001) to merge adjacent fragmented regions into a one. Specifically, if two regions share the same function and their distance is smaller than a given threshold, they are merged into a single region along the nearest edge.

As mentioned earlier, to train our model, we use the vector data of the target region for accurate supervision and the pixel-level data of the surrounding regions for efficient computation. Thus, we also represent our dataset in pixel format. Compared with the vector data, the pixel-level data can be more efficiently processed by CNNs. For the urban road layout, we plot the roads on a binary image, where a 1 indicates a road pixel. Similar to the urban road layout, we also plot all buildings with each sample on a binary image, and the function on another image (*i.e.*, each pixel stores a value ranging between [0, 11]). We show an example of our pre-processed dataset in Fig. 2 and the statistics of the whole dataset in Table 1.

In summary, our large-scale urban layout dataset contains more than 147K regions covering 1,300 km^2 with rich annotations, including function, region shape, urban road layout, and urban building layout.

4 Our Method

Given a target region R, we aim to regenerate the urban layout L of region R, based on a given target function t and surrounding context S. We define the output urban layout $L = \{L_r, L_b\}$ as a collection of road layout L_r and building layout L_b. While the target function t determines the main properties of the final layout, the surrounding context S (*i.e.*, the functions and layout of the surrounding regions) guides the generation of local details, especially for the layout near the boundary of the target region. We follow the general strategy (Parish & Müller, 2001) by first predicting the road layout, and then the building layout afterwards.

To guide the regeneration process, we first discuss how we extract implicit regeneration rules from the inputs $\mathcal{R}$ from $(t, S) \Rightarrow \mathcal{R}$ in Sect. 4.1. We then discuss how we regenerate the layout of the target region based on these extracted rules in Sect. 4.2. The overall framework of our UrbanEvolver is shown in Fig. 3.

4.1 Regeneration Rules Extraction

Since an urban layout may contain thousands of elements, such as roads and buildings, directly learning the regeneration rules from vector data consumes a large amount of computational resources. For example, a large number of irregular buildings need an ultra-high dimensional matrix to store. We thus render the vector data in pixel format for a more efficient learning of the regeneration rules. We represent the target function t and functions of the surrounding regions together as a functional map $M_f = \{M_{fs} + t \times R\}$ and the urban layout of surrounding regions as a surrounding layout map M_{ls}, where M_{fs} indicates the functional map of surrounding regions.

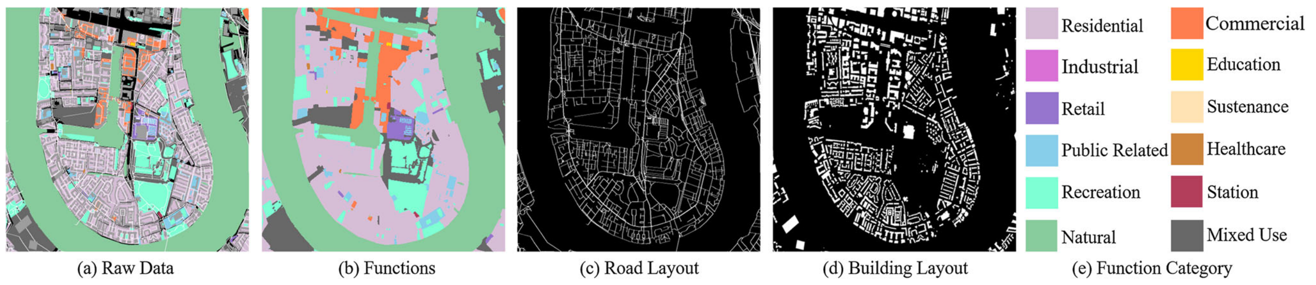

Fig. 2 A visual example of our collected dataset. Our dataset contains rich annotations on functions and region shapes, road layout and building layout

Table 1 The statistics of our dataset

Function		Region			Road	Building	
		Total area	Top 3 NN.	Count	Density	B.C.R.	Count
Residential	(RES)	608.34	REC,MU,NAT	12,285	Dense	27.14	604,082
Industrial	(IND)	151.02	RES,MU,REC	1789	Sparse	36.82	13,596
Commercial	(COMM)	105.18	RES,MU,REC	2694	Normal	38.71	21,309
Mixed-use	(MU)	98.25	RES,REC,NAT	22,388	Normal	5.53	46,280
Recreational	(REC)	97.75	RES,MU,NAT	33,416	Normal	7.86	120,541
Natural	(NAT)	72.60	RES,REC,MU	9473	Sparse	5.59	18,689
Retail	(RETAIL)	65.53	RES,REC,MU	5693	Normal	56.40	56,021
Public-related	(PUB)	44.39	RES,REC,MU	41,920	Sparse	29.14	58,044
Education	(EDU)	22.83	RES,REC,MU	1199	Sparse	40.99	3,373
Healthcare	(HEALTHC)	18.74	RES,REC,MU	2124	Sparse	35.69	5041
Station	(STN)	18.19	RES,EDU,COMM	614	Sparse	16.47	1868
Sustenance	(SUS)	14.54	RETAIL,RES,REC	13,829	Normal	35.17	29,197

Total area–the total area (km^2) occupied by each function. *Top 3 NN.*–the top 3 nearest neighbourhood functions of each function. This value provides guidance to the user for selecting the target function. *Density*–the density level of the road layout belonging to each function. We rank the level based on (Zhang et al., 2015). *B.C.R.*–the building coverage ratio (%) reflects the building density of each function. *Count*–the total number of regions or buildings belonging to a function

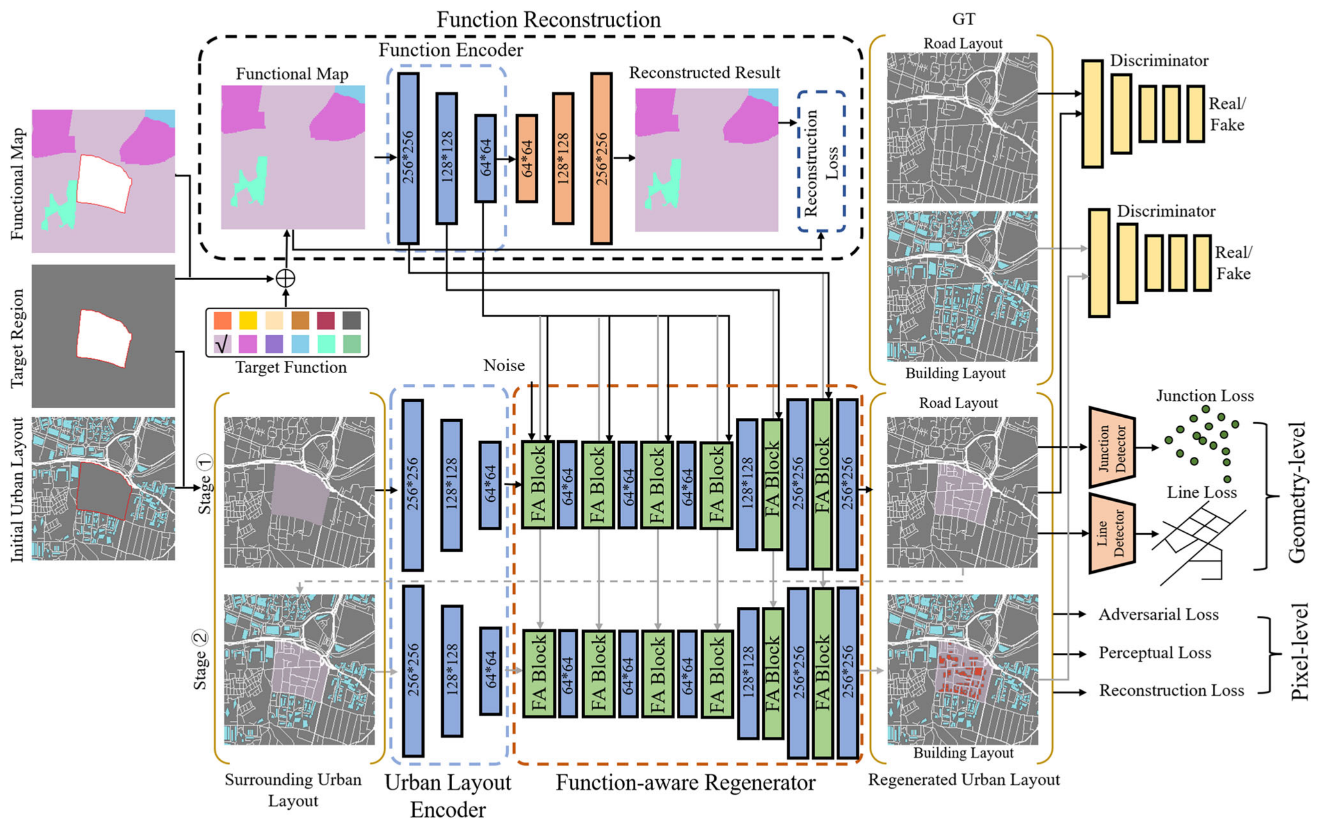

Fig. 3 Overview of UrbanEvolver. Given a target region, UrbanEvolver regenerates the urban layout of this target region conditioned on the specified target function and surrounding context. UrbanEvolver generates the layout in two stages, by first generating the road layout, and then the building layout. It first extracts the functional map features from a pre-trained function encoder, learned through a reconstruction task. The functional map features are then fused with urban layout features through FA blocks for layout regeneration. A series of geometry-level and pixel-level losses are designed for training

When encoding these two maps, a straightforward solution is to use a simple unified encoder, but is not suitable for our task. Since the functional map serves as a high-level signal to determine the global urban layout style of the target region and the surrounding urban layout map guides the synthesis of local urban layout details, a simple unified encoder shows poor performances in extracting information on different domains (which is validated in Sect. 5). We thus extract features separately from the functional map and from the layout map through different encoders, as explained below.

Function Encoder. We represent the function (or land use type) of each region as a functional map M_f, and encode visual clues using CNNs. The function encoder contains three conv layers with kernel size 3×3 followed by ReLU to extract multi-scale features. Formally, we obtain the functional map features as:

$$F_{f}{}_{n=1}^{K} = \sigma(W^n(M_f) + b^n), \tag{1}$$

where ${F_f}^n$, σ, W^n and b^n are the functional map features, the activation layer, the weights and bias at n^{th} layer, respectively.

Urban Layout Encoder. We represent the surrounding urban layout as a surrounding urban layout map M_{ls}. We use the same structure as the function encoder to encode M_{ls} to obtain the surrounding urban layout features F_{ls}. As we generate the road layout and building layout separately, we use two separate layout encoders of the same architecture but different inputs. For the road layout branch, the input is the initial surrounding urban layout map M_{ls}. For the building layout branch, the input is a combination of M_{ls} and the generated road layout of the target region.

Auxiliary Task - Function Reconstruction. During training, we find that the convergence rates are different between the function and urban layout encoders. As the information densities of the functional map and urban layout are different, we introduce an auxiliary task called function reconstruction to help with the learning of functional map features. This auxiliary task aims to reconstruct the input functional map through an autoencoder, using the same function encoder introduced above. In the reconstruction, we set three conv layers with kernel size 3×3 and two up-sample layers with scale factor 2 to obtain the reconstructed functional map. The function encoder is fixed after this pretraining. This further allows the two-stage regeneration subnets to better disentangle the (surrounding) urban layout features, achieving better performances.

4.2 Function-Aware Urban Layout Regeneration

The layout regeneration contains two stages, following the standard of previous works (Parish & Müller, 2001; Lipp et al., 2011). It first regenerates the road layout, based on

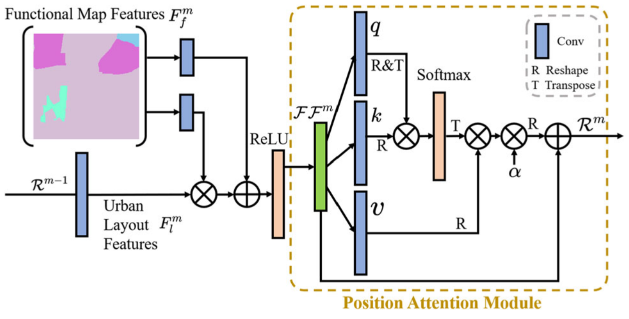

Fig. 4 Structure of a function-layout adaptive (FA) block

which it then generates the building layout. This is because the building layout is often determined by the road layout, and two-stage regeneration allows more dynamic manipulation by users. As shown in Fig. 3, we adopt two networks of the same architecture but different weights for these two stages. For each stage, we use the implicit regeneration rules encoded in the functional map features F_f and the (surrounding) urban layout features F_{ls} to guide the regeneration process.

A simple way is to directly concatenate these features before sending to the decoder. However, as mentioned above, the functions and the surrounding urban layout play different roles in layout regeneration. The function features learned from the function encoder focus on the high-level semantics, while the surrounding urban layout features learned from the urban layout encoder focus on the local details of the urban layout. Thus, directly using the concatenation operation is not an effective way of fusing features in diverse distributions (Zheng, 2015), as it can weaken both the global control by the target function and local details inferred from the surrounding context.

Instead, we propose the function-layout adaptive (FA) block to effectively extract the regeneration rules, inspired by (Park et al., 2019). The structure of the FA block is shown in Fig. 4. Given the functional map features and the surrounding urban layout features, the FA block adaptively fuses the features to guarantee the local details while facilitating the guidance from a high-level function. For each block, to maintain the semantic control, the functional map features are first passed into two different conv layers to obtain two modulation parameters γ and β. These two parameters are fused with the regenerated urban layout features F_l in an element-wise way as:

$$\mathcal{FF}^m = \gamma^m \frac{Sigmoid(F_l^m) - \mu^m}{\sigma^m} + \beta^m, \tag{2}$$

where μ and σ are the mean and standard deviation of $Sigmoid(F_l)$. m is the index of the FA block. Note that $F_l^m = F_{ls}$ when $m = 1$.

As the features in various locations may not contribute to the regeneration rules equally, we further introduce the position attention module (Fu et al., 2019) to our FA block. This can model global contextual relationships and improve semantic consistency among features. Formally, we obtain the regeneration features $\mathcal{R}^m$ by:

$$\mathcal{S}_{ji} = \frac{\exp(q_i \cdot k_j)}{\sum_{i=1}^{N} \exp(q_i \cdot k_j)}, \tag{3}$$

$$\mathcal{R}_j^m = \alpha \sum_{i=1}^{N} \left(\mathcal{S}_{ji} v_i\right) + \mathcal{FF}_j^m, \tag{4}$$

$$F_l^{m+1} = Conv(R^m), m < K, \tag{5}$$

where i and j are the position indexes. $N = H \times W$ and K is the total number of FA blocks. The weight α is learnable and set to be strictly positive during training. It is gradually increased from 0, allowing the network to focus gradually from the local to the global features (Zhang et al., 2019). The features q, k and v are calculated through different conv layers with $\mathcal{FF}^m$ as input. Rather than only considering regeneration features in a single scale, our model progressively incorporates the regeneration features in different scales as shown in Fig. 3. We first stack four FA blocks followed by ReLU and conv layers to progressively obtain the implicit regeneration rules from functional map features and surrounding layout features. To get the output of the same size as the input, we then set the up-sample layers with scale factor 2 to gradually increase the output size. After up-sampling the output size, we still add the FA blocks followed by ReLU and conv layers. Besides, to increase the diversity of the regenerated results, we inject a noise vector sampled from a normal distribution in the first FA block. The regeneration features $\mathcal{R}^K$ derived from the last FA block represent our learned regeneration rules. Finally, we obtain the regenerated urban layouts L with the same size as the input. Our regenerated urban layouts are in binary format. We can transform them into vector format. For the road layout, we use the line detector (Pautrat et al., 2021) to get the nodes of the road and further obtain coordinate sequences of the nodes. To filter noisy nodes, we merge nodes with a distance less than 2 m. For building layouts, we obtain the contours of the buildings (Suzuki et al., 1985) and then the coordinate sequences of contours. To avoid irregular shapes, we also merge adjacent nodes of the contours with a distance less than 0.5m. In this way, we obtain continuous and rational urban layouts in vector format.

4.3 Loss Functions

To train our model, we design a set of losses mainly of two groups: geometry-level losses and pixel-level losses. Before discussing them in detail, we first introduce the function loss to enable auxiliary function reconstruction for extracting functional map features. Note that this is independently trained with the following function loss:

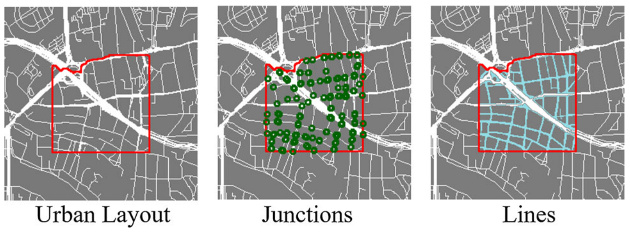

Fig. 5 Junctions (*i.e.*, dark green circles) and lines (*i.e.*, blue lines) of the regenerated urban layout in the target region (within the red boundaries) (Color figure online)

$$\mathcal{L}_f = \left\| M_f - O_f \right\|_1, \tag{6}$$

and the weights are fixed for two-stage urban layout regeneration, where O_f is the predicted functional map and M_f is the ground truth.

Geometry-level Losses. We keep the vector data in the target region for accurate supervision. Geometry-level losses mainly constrain the geometric structure within the target region based on lines (*i.e.*, roads) and junctions (*i.e.*, crossings), which are two key components of the urban layout, as shown in Fig. 5.

- ***Junction loss***: It measures how close the number and position of junctions in the regenerated results are to those in the ground-truth. Each junction is represented as a single centered pixel with a value of 1, and we rely on a junction detector (Pautrat et al., 2021) $\mathcal{D}_J$ to obtain the junctions in the regenerated urban road layout L_r. We use the binary cross entropy loss:

$$\mathcal{L}_j = -(G_j \cdot \log \mathcal{D}_J(L_r) + (1 - G_j) \cdot \log(1 - \mathcal{D}_J(L_r))), \tag{7}$$

  where G_j is the ground-truth of junctions in the target region.
- ***Line loss***: It penalizes incorrect lines in the regenerated results. To obtain the lines, we use a line detector (Pautrat et al., 2021) $\mathcal{D}_L$, and compute the differences between the regenerated lines and ground-truth lines in hough space (Zhao et al., 2021), as:

$$\begin{aligned}\mathcal{L}_l = &-(\{\mathcal{H}(G_l) \cdot \log(\mathcal{H}(\mathcal{D}_L(L_r)))+ \\ &(1 - \mathcal{H}(G_l)) \cdot \log(1 - \mathcal{H}(\mathcal{D}_L(L_r)))\}),\end{aligned} \tag{8}$$

  where $\mathcal{H}(.)$ is the deep hough transformation to transfer the lines from pixel space into parameter space, and G_l is the ground-truth of lines in the target region.

Pixel-level Losses:

- ***Reconstruction loss***: We use the $\mathcal{L}_1$ loss to encourage the model to predict per-pixel layout accurately. We compute the difference between the generated urban layout O_l in pixel format and the ground truth M_l. Note that to encourage modeling the connectivity among boundary regions, we also generate the surrounding regions and define $O_l = P_{ls} + L$, where P_{ls} is the reconstructed surrounding urban layout. We further add a term to penalize layout discrepancy within the target region, as:

$$\mathcal{L}_r = \|M_l - O_l\|_1 + \lambda_r \, \|R \times (M_l) - L)\|_1 , \tag{9}$$

  where λ_r is the penalty weight. It is set to 5 empirically.
- ***Perceptual loss***: This loss measures the feature distance between the ground-truth M_l and the regenerated urban layout O_l extracted by a pre-trained VGG19 (Simonyan & Zisserman, 2015) network. We denote ϕ_i as the activation function of the $i-th$ layer of the pretrained network, and define the loss as:

$$\mathcal{L}_p = \frac{1}{K}\sum_{i=1}^{k} \frac{1}{C_i H_i W_i} \|\phi_i(O_l) - \phi_i(M_l)\|_2^2 , \tag{10}$$

  where K, C_i, H_i, W_i are the numbers of activation layers and channels, height and width of a particular activation. We select $relu\_1\_2$, $relu\_2\_2$, $relu\_3\_3$, $relu\_4\_3$ layers, following the previous work (Johnson et al., 2016).
- ***Adversarial loss***: We also add adversarial loss to predict realistic layout following the distributions of ground truths:

$$L_a = -\mathbb{E}_{O_l}[\log(1 - D(O_l)] , \tag{11}$$

  where D is the discriminators.

In summary, our final loss L is formulated as:

$$\mathcal{L} = \lambda_{lj}\mathcal{L}_j + \lambda_{ll}\mathcal{L}_l + \lambda_{lr}\mathcal{L}_r + \lambda_{lp}\mathcal{L}_p + \lambda_{la}\mathcal{L}_a, \tag{12}$$

where $\lambda_{lj} = 10$, $\lambda_{ll} = 10$, $\lambda_{lr} = 20$, $\lambda_{lp} = 3$ and $\lambda_{la} = 2$ in our experiments.

5 Experiments

In this section, we compare our method with other state-of-the-art methods qualitatively in Sec. 5.1 and quantitatively in Sec. 5.2. We also demonstrate the effectiveness of our model design through an ablation study in Sec. 5.3.

Implementation Details. We implement our model using Pytorch, and we train our model on two NVIDIA RTX 2080TI for about five days. We set the bath size as 6 and the initial learning rate as 0.0001. We use the Adam optimizer with $\beta_1 = 0.0$ and $\beta_2 = 0.9$. We randomly sample 13,250 maps with a field-of-view (FoV) $1,000 \times 1,000 m^2$ as the training dataset, and 1,500 maps with the same FoV (which do not overlap with the training samples) as the test dataset, from our collected dataset. For each map, we select the central region with a single function as the target region and the other regions as the surrounding regions. Notably, the target regions are divided by diverse roads resulting in a wide variety of region shapes, as shown in Fig. 6, thus enhancing the robustness of our model. We limit the target region to a maximum of 25% of the entire map.

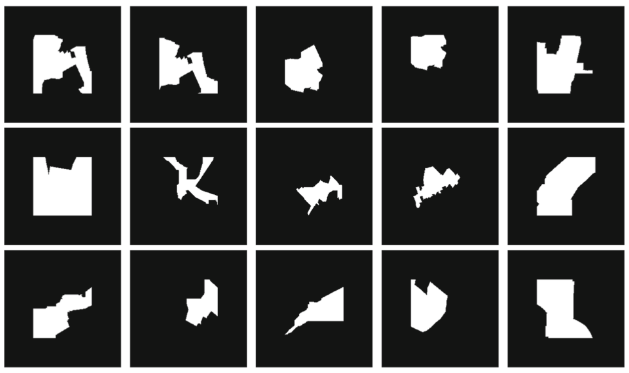
Fig. 6 Example target regions in our dataset

Evaluation Metrics. We evaluate our model based on several popular metrics used in image generation tasks (Brock et al., 2019; Choi et al., 2020; Benny et al., 2021; Sushko et al., 2021) metrics: **FID** (Heusel et al., 2017), **GS** (Khrulkov & Oseledets, 2018), and **MS-SSIM** (Wang et al., 2003). FID utilizes InceptionV3 (Szegedy et al., 2016) to evaluate the quality and diversity of the regenerated urban layout. Like (Chu et al., 2019), we also fine-tune the InceptionV3 network on our dataset, where the input is the urban layout of a region and the output is the function. After that, we use the fine-tuned InceptionV3 network to extract the features of the target urban layout for FID calculation. GS compares the geometrical properties of the ground-truth manifold and the regenerated urban layout data manifold. MS-SSIM evaluates the structural similarity of the regenerated data and the ground-truth data at multiple scales.

We also evaluate our model using city planning based metrics. Inspired by (Alhalawani et al., 2014; Chu et al., 2019; Chen et al., 2021), we introduce connectivity index (CI), road layout similarity (RS), connectivity to surrounding layouts (CSL), building pattern similarity (BS), layout diversity (LD) and layout classification (LC).

- CI measures how well the regenerated road network connects the destinations. We compute it as the ratio of the number of links to the number of road nodes as:

$$CI = \frac{\sum_i^N valence(\hat{v}_i)}{N}, \tag{13}$$

where $\hat{v}$ is the node in the regenerated road network, $valence(\cdot)$ is the valence of the node, N is the number of nodes. Note that the minimum rational CI is 1.4 under urban layout design guidelines (Alhalawani et al., 2014), and a low CI indicates that there are a lot of broken roads.

- RS measures how well the regenerated road layouts match the characteristics of target road layouts. It is computed as the ratio of the total number of regenerated roads minus the total number of ground-truth target roads to the number of testing samples as:

$$RS = \frac{\sum_i^N \mid num(L_r) - num(G_l) \mid_i}{N}, \tag{14}$$

where L_r and G_l are the regenerated and ground-truth target urban road layouts. $num(\cdot)$ is the total number of roads. N is the total number of testing samples.

- CSL measures the continuity between the regenerated roads and surrounding roads. We compute the average distance between the end node of the regenerated roads and its closest end node of surrounding roads on the region boundary as:

$$CSL = \frac{\sum_i^N \parallel v_i - \hat{v}^i \parallel}{N} \tag{15}$$

where $\hat{v}$ is the end node of the regenerated roads. v is its closest end node of the surrounding roads. N is the number of nodes on the boundary.

- BS evaluates the morphological and spatial similarity of buildings between regenerated and ground-truth building layouts. We first model the building layout using the Delaunay triangulation (DT) graph (Lee & Lin, 1986), where nodes are morphological properties (shapes) and edges are spatial relations. We then transform the DT graph from spatial domain to frequency domain based on Graph Fourier Transform. Finally, we compute BS as the deviation between regenerated and GT target building layouts w.r.t. the frequency domain features as:

$$\begin{aligned} \mu &= (\mu_{o1} - \mu_{g1})^2 + \cdots + (\mu_{on} - \mu_{gn})^2 \\ &\quad + (\mu_{om-n+1})^2 + \cdots + (\mu_{gm})^2 \\ BS &= \sqrt{\mu}, \end{aligned} \tag{16}$$

where μ_o and μ_g are frequency signals of regenerated and ground-truth building layouts.

- LD (Chu et al., 2019) measures the diversity of the regenerated urban layouts. We compute the percentage of urban layouts in one map falling outside the 20 m vicinity of the urban layout in the other map, and vice versa. The elements included in the urban layouts we used in LD consist of road-related elements: roads and intersections, and building-related elements: building footprints and distributions.
- LC measures the degree to which the characteristics of the regenerated layouts match the characteristics of the layouts of the specified functions using layout classification. The characteristics include road width, road style, building shape and building distribution. We utilize multi-module layout features including layout image, road density, building density, 4-way crossing proportion and connectivity index to classify the function of the regenerated urban layouts (Alhalawani et al., 2014) and obtain the accuracy as the LC score. The 4-way crossing proportion means the proportion of 4-way crossing w.r.t. all intersections.

Note that we utilize 10-fold cross-validation for evaluation. The metrics are computed on the test dataset with 1500 maps.

Baselines. To the best of our knowledge, ours is the first work aiming to perform the function-aware urban layout regeneration task. By regarding the surrounding urban layout map as a damaged image and the target region as the hole, image inpainting shares some similarities with our problem. We thus treat several state-of-the-art (SOTA) image inpainting methods both unconditionally and conditionally as our baselines. In addition, apart from comparing to our data-driven learning methods, we also compare our method with the traditional rule-based methods.

- Unconditional image inpainting methods: PConv (Liu et al., 2018), FeatEq (Liu et al., 2020), SPN (Zhang et al., 2021) and LaMa (Suvorov et al., 2022). We train all these methods using their official released codes on our training dataset. We concatenate the surrounding urban layout map and the functional map together along the channel dimension for feeding into these methods.
- Conditional image inpainting methods: EdgeConnect (Nazeri et al., 2019) and CTSDG (Guo et al., 2021b). Instead of filling the holes based on only the damaged images, these methods take an edge map as a condition for guiding the inpainting. To train these methods, we adopt the same approach used by the unconditional image inpainting methods, but also take the functional map as a conditional input.
- Rule-based method: CityEngine (Kelly, 2021). CityEngine is a SOTA method based on a rule-based urban synthesis approach (Parish & Müller, 2001). The parameters used in the rules need to be manually adjusted through an interface. For fair comparisons, we assign the

necessary values based on the statistics from our training dataset, such as the angle, the crossing ratio and the range of the road length, and on different target functions. Directly setting the parameters is not sufficient, and we further manually merge the regenerated urban layout with the surrounding for smooth transition along the boundary. Specifically, we combine adjacent points and remove artificial lines one by one to ensure that the joints are smooth. As this is a labor-intensive task, we only compare with this method qualitatively.

5.1 Qualitative Results

Urban Road Layout Regeneration. We show the qualitative results in Fig. 7. We can see that SPN and EdgeConnect fail to regenerate urban road layouts. The target regions either remain empty or become unreasonably dense. FeatEq also regenerates unreasonably dense structures only at the centers of the target regions. Although PConv, CTSDG and LaMa can regenerate urban road layouts, they fail to produce geometric structures that are similar to the ground truths. PConv regenerates repetitive layout patterns within some urban layouts, as pointed by the arrows. CTSDG and LaMa produce lots of noise, causing a lot of broken and dotted roads, which are impractical in real life. The main reason for the failure of these baseline methods is that they are designed for natural images, which have rich texture and structural information. In contrast, urban road layouts have sparse structures without any textures. For example, due to the lack of texture information, SPN cannot obtain effective semantic priors to facilitate layout regeneration. Besides, for the conditional image inpainting methods, our functional map is very different from the original edge map used as the "condition", as the edge map tends to impose local constraints, while our functional map tends to provide global guidance. This shows that all these existing methods developed for relevant tasks are not suitable for our task.

The rule-based method, CityEngine, can regenerate valid urban road layouts, but are limited only to a few styles because of the hand-crafted modeling rules. In addition, as it requires manually merging the regenerated urban road layout with the surrounding, the regeneration process can be tedious and time-consuming. Instead, our method can regenerate valid urban road layouts with rich styles. For example, in the first three columns of Fig. 7, even for the same function (*i.e.*, residential), our model can regenerate different urban road layouts to match the different surrounding contexts in style and orientation near the boundaries.

Complete Urban Layout Regeneration. As regenerating the urban building layout on the invalid road layout is meaningless, we only compare with PConv, CTSDG and LaMa that can regenerate feasible urban road layouts. The regenerated results are shown in Fig. 8. We can see that PConv, CTSDG and LaMa fail to regenerate valid building layouts not only because of the artifacts in the synthesized road layouts, but also due to the lack of an effective mechanism for correlating the functions and the building layouts. In contrast, our method can regenerate valid urban layouts, which are close to the ground truths for both roads and buildings.

To demonstrate the effectiveness of our model and its generality, we show the results of our method in regenerating urban layouts under different user-specified functions in Fig. 9. Our method can regenerate urban layouts with rational geometric structures while maintaining the functional characteristics (*e.g.*, road density and building size). For example, roads in residential regions have a higher density than those in commercial and mixed-use regions, and buildings in commercial regions have larger footprints. Even under the same function, our model is able to regenerate diverse results for the same input design scenario, as shown in Fig. 10. We also find that our method can regenerate urban layout with multi-functions in the target region as a condition. We show a few such examples to demonstrate the generalization of our model in Fig. 11.

5.2 Quantitative Results

We measure the performances of the compared methods quantitatively first on urban road layout regeneration as shown in Table 2, and then on complete urban layout regeneration (including both roads and buildings) as shown in Table 3. We can see that our model outperforms all the baselines on all three metrics for the urban layout regeneration task (*i.e.*, including both roads and buildings).

The quantitative results of city planning based metrics are shown in Table 4. The results show that compared with the baselines, our method regenerates more rational and diverse urban layouts, including road layouts and building layouts. PConv, CTSDG and LaMa regenerate a lot of broken roads, resulting in low CI scores. Besides, as PConv and CTSDG regenerate many empty areas, *i.e.*, they cannot regenerate urban layouts, their LD scores are extremely high. The low RS, low BS and high LC scores indicate that our method can regenerate the desired urban layouts following the implicit representation of the regeneration rules. The low CSL score indicates that our method regenerates continuous roads near the boundary.

5.3 Ablation Study

In this subsection, we examine the effectiveness of our key model components, including the function-layout adaptive (FA) block, the geometry-level losses and the pixel-level losses.

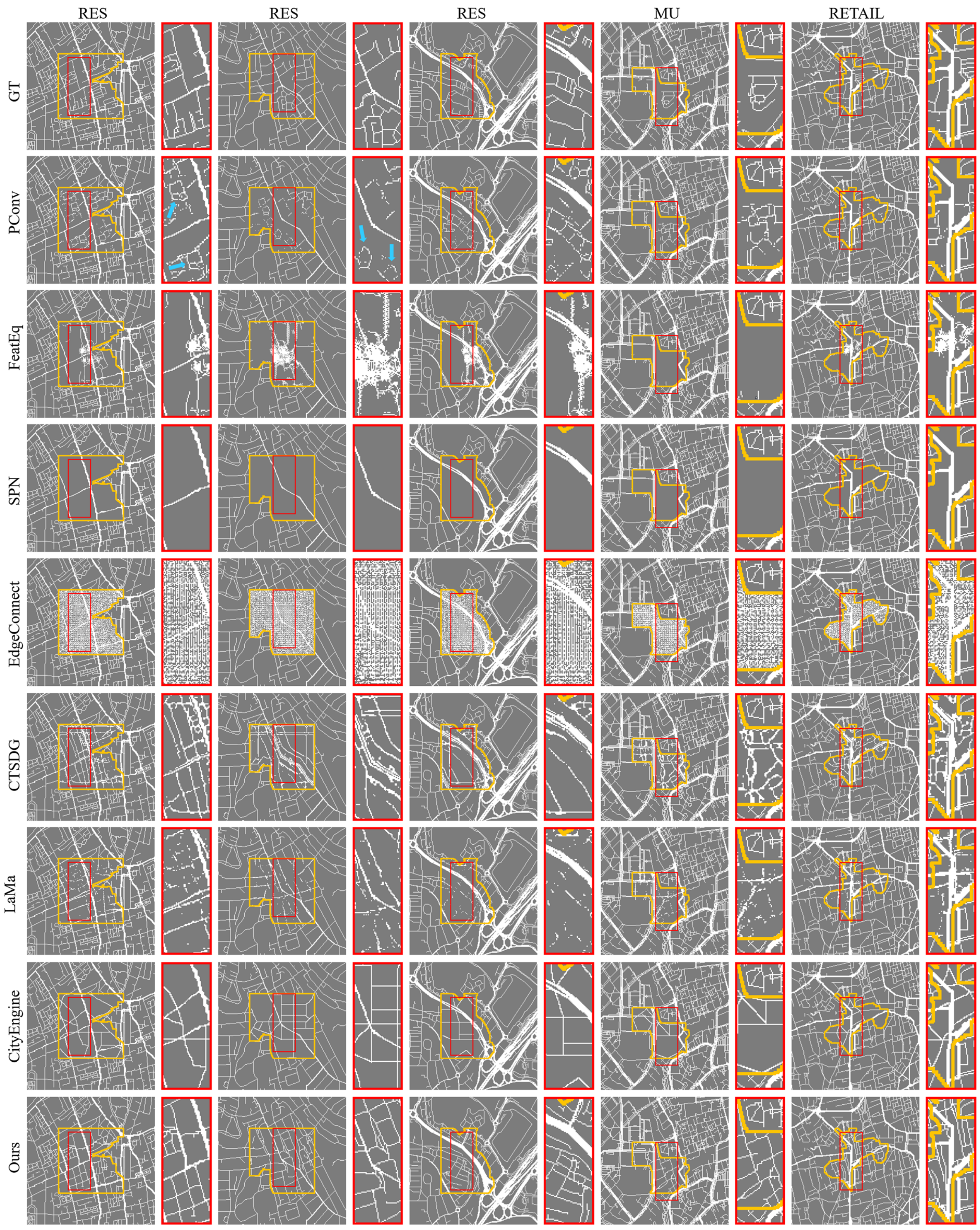

Fig. 7 Comparison with baselines for urban road layout regeneration. The target regions are indicated with yellow boxes. We highlight the regenerated result within the red box on the right of each image. The target function is shown at the top of each column (Color figure online)

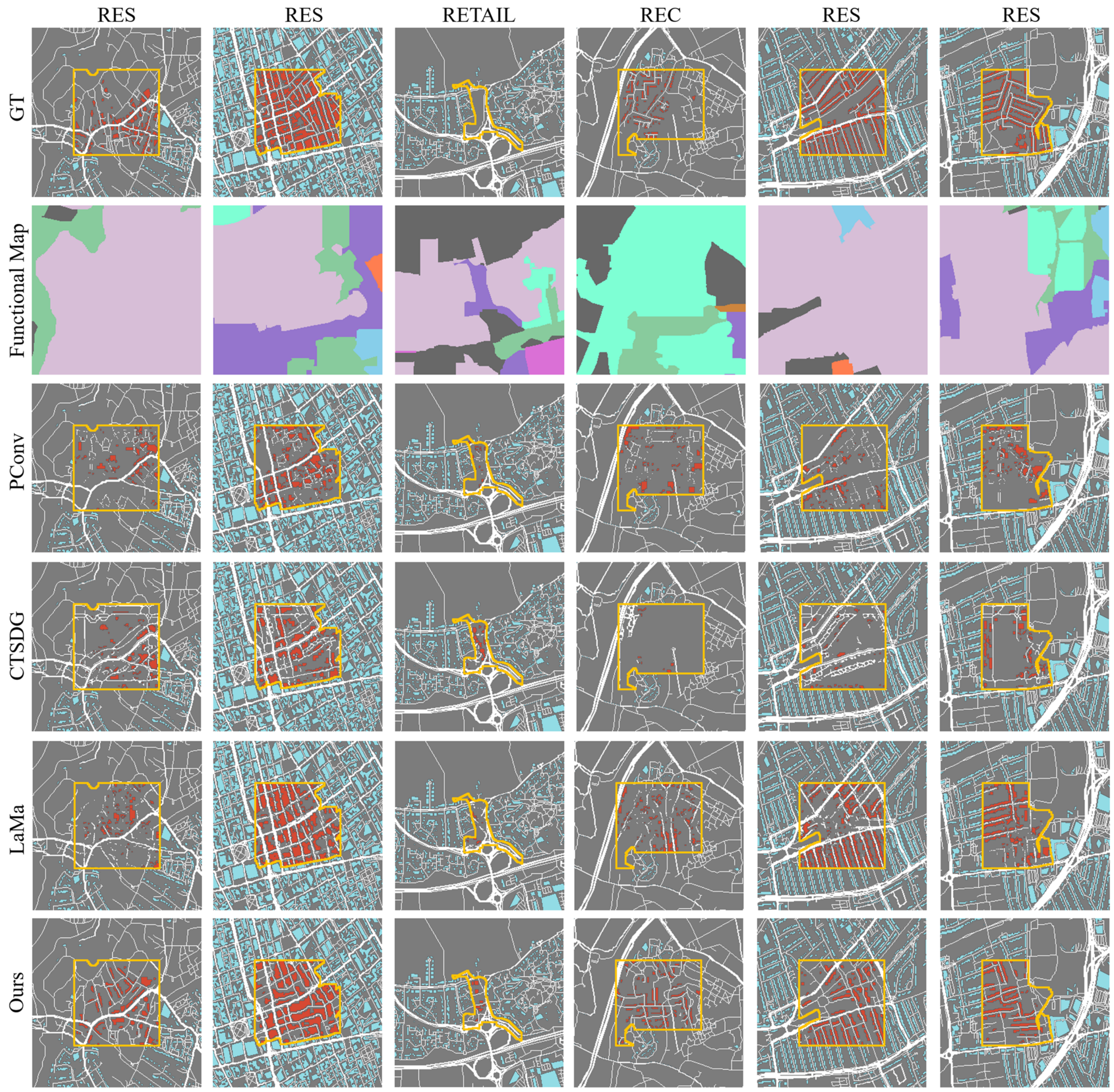

Fig. 8 Comparison with baselines for the complete urban layout regeneration. The complete urban layout includes the urban road layout and the urban building layout. The target function is shown at the top of each column

5.3.1 The Effect of the FA Block

To evaluate the effectiveness of the FA block, we create two variants:

- **w/o FA Blocks**, by replacing the FA blocks with vanilla convolutional layers and directly concatenating the surrounding urban layout features and functional map features during the regeneration;
- **w/o Position Attention**, by removing the position attention module in the FA blocks.

Results of the qualitative comparison are shown in Fig. 12. We can see that by removing the FA blocks, the method regenerates broken roads with artifacts and irregular buildings. It also fails to regenerate distinguishable urban layouts based on different input functions, *e.g.*, the regenerated layouts are nearly the same in the second row of Fig. 12. By only removing the position attention module within each FA block, the

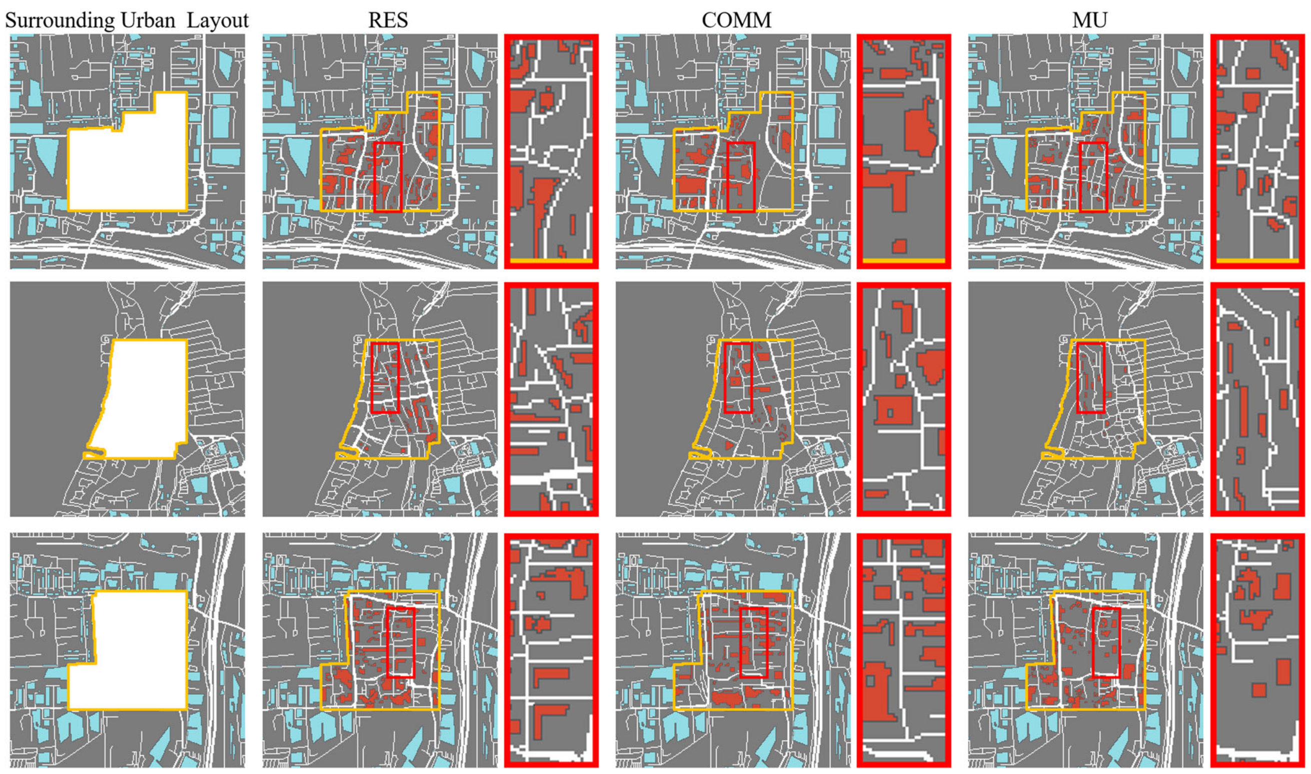

Fig. 9 Urban layout regeneration results by our method under different input target functions. Note how the regenerated urban layouts in the target region change according to the input target function. The target regions are indicated by yellow boxes. We highlight the results within the red box on the right of each image (Color figure online)

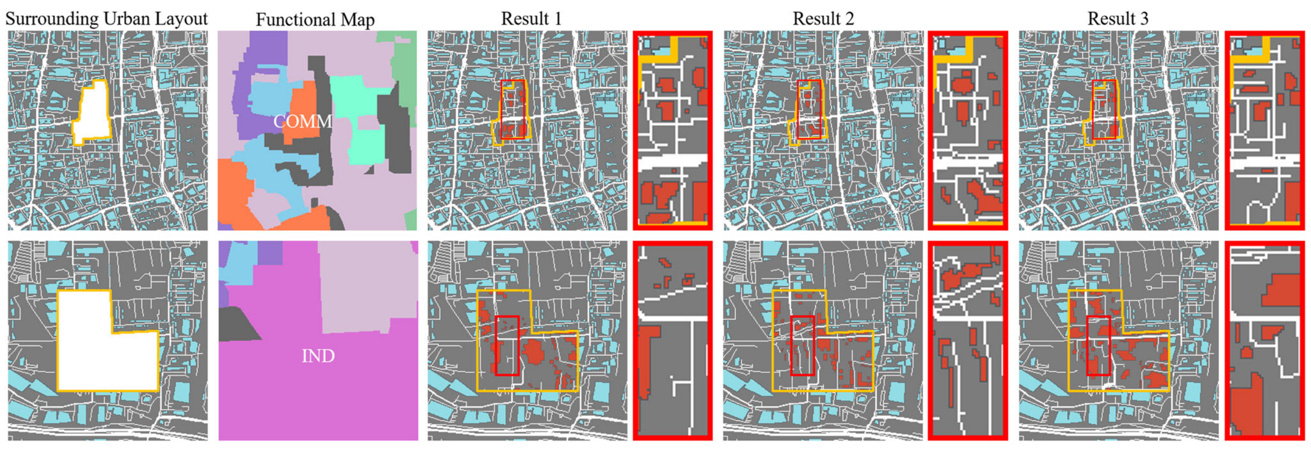

Fig. 10 Diverse results regenerated for each design scenario (with the same target region and function). The target regions are indicated by yellow boxes. We highlight the results within the red box on the right of each image (Color figure online)

results become better but still contain obvious artifacts, such as broken roads.

We further show the feature maps computed by the last second layer of the function-aware regenerator, with and without the attention module, in Fig. 13. We keep the spatial dimensions and visualize the mean values across the channel for each feature map. (Warmer colors indicate higher values.) As can be seen, after adding the position attention module, our model can focus more on the target regions to ensure the quality of the regenerated results. In summary, our FA blocks can effectively fuse the global features from the functions, and local features from the surrounding urban layout. We further show the quantitative evaluation in Table 5. The performance drops of "w/o FA Blocks" and "w/o Position Attention" indicate the necessity of our FA blocks in the function-aware urban layout regeneration task.

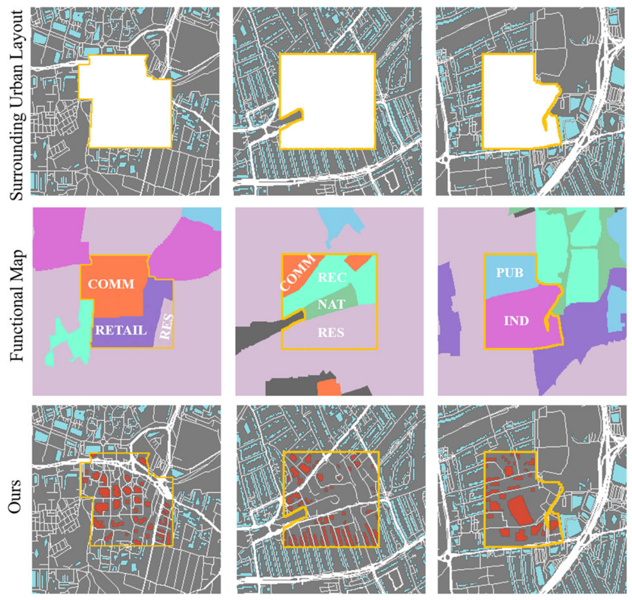

Fig. 11 Urban layout regeneration with multi-function target regions. Yellow boxes indicate the target regions (Color figure online)

Table 2 Comparison with baselines for urban road layout regeneration

Metrics	FID ↓	GS ↓	MS-SSIM ↑
PConv	23.63	25.08	0.73
FeatEq	30.49	34.88	0.66
SPN	35.44	46.95	0.65
EdgeConnect	151.08	90.15	0.45
CTSDG	33.05	41.80	0.66
LaMa	22.12	20.70	0.75
Ours	**17.58**	**10.32**	**0.83**

Best results are highlighted in bold

Table 3 Comparison with baselines for complete urban layout regeneration

Metrics	FID ↓	GS ↓	MS-SSIM ↑
PConv	34.76	45.89	0.65
CTSDG	38.55	52.81	0.63
LaMa	32.11	40.43	0.66
Ours	**30.57**	**35.70**	**0.68**

Best results are highlighted in bold

5.3.2 The Effects of Losses

The effect of geometry-level losses. We compare the performances of our model without each of the geometry-level losses, *i.e.*, **w/o** $\mathcal{L}_j$ and **w/o** $\mathcal{L}_l$. We show the qualitative results in Fig. 14 and quantitative results in Table 5. Without training on the junction loss $\mathcal{L}_j$, a structured road network and buildings can hardly be formed. Without training on the line loss $\mathcal{L}_l$, the model may regenerate broken roads with noise,

Table 4 Quantitative results of our methods and the baselines

Metrics	CI↑	CSL ↓	RS ↓	BS↓	LC↑	LD ↑
PConv	1.53	5.64	103.46	0.65	0.17	0.95
CTSDG	1.23	9.37	90.50	0.78	0.13	0.93
LaMa	1.92	3.64	50.23	0.55	0.32	0.30
Ours	**3.22**	**1.32**	**11.63**	**0.28**	**0.75**	0.62

Best results are highlighted in bold
We evaluate CI (the minimum rational CI is 1.4 under urban layout design guidelines), RS, CSL, BS, LD and LC. Extremely high LD scores indicate failure regeneration

causing invalid building layouts. For example, a building is generated on a road crossing as pointed to be a blue arrow in Fig. 14c. These results indicate that the junction detector helps the model learn road structural details, while the line detector helps the model generate continuous roads.

The effect of pixel-level losses. Similarly, we remove each of the pixel-level losses in this experiment, which include: reconstruction loss **w/o** $\mathcal{L}_r$, perceptual loss **w/o** $\mathcal{L}_p$, and adversarial loss **w/o** $\mathcal{L}_a$. We show the quantitative results in Table 5 and qualitative results in Fig. 14. By removing $\mathcal{L}_r$, the regenerated roads and buildings are often incomplete with artifacts. For example, in the 2^{nd} row of Fig. 14d, the road width is not consistent; it may suddenly become wider near a junction. Besides, with the help of the perceptual loss $\mathcal{L}_p$ and the adversarial loss $\mathcal{L}_a$, the regenerated results have closer distribution with the ground truth, further improving the regeneration performances.

With all the suggested geometry-level and pixel-level losses, our model can achieve the best performance on all three metrics in the urban layout regeneration task.

6 Other Applications

In this section, we demonstrate more applications enabled by our model.

6.1 Other Cities

We regenerate urban layouts in other cities, such as Paris, Madrid and Shanghai. For this experiment, we have collected the urban layout data of Paris, Madrid and Shanghai from OSM. After fine-tuning our model on these data, we regenerate urban layouts as shown in Fig. 15.

6.2 Interactive Regeneration

UrbanEvolver supports different types of interactive urban layout regeneration, not only limited by changing the input

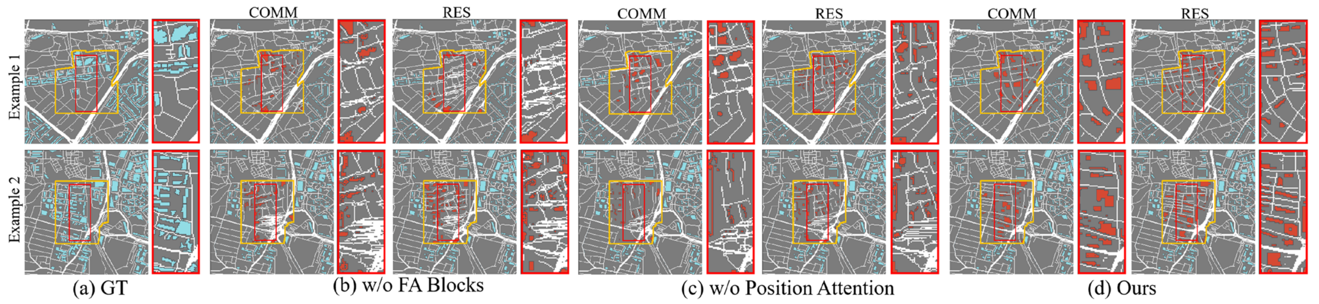

Fig. 12 The effects of the FA block and the position attention module. RES and COMM are functions, indicating residential and commercial, respectively

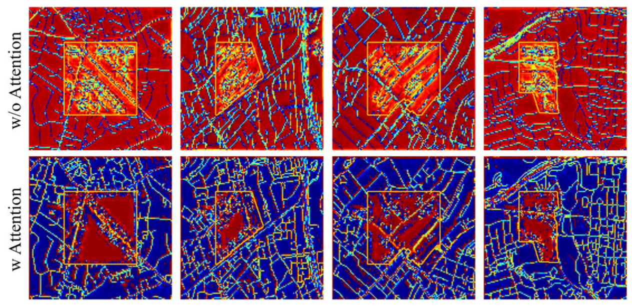

Fig. 13 Visualization of the feature maps derived from the second last layer of the function-aware regenerator, with and without the position attention module. A warmer color indicates a higher value

function as shown in Fig. 9. Here, we show another type of control, by conditioning on the user-specified main roads.

Main roads determine the structure and characteristics of an urban layout (Aliaga et al., 2008), making such an application practical. Our model can allow users to customize the regenerated results through specifying the main roads, without re-training. This can be achieved by removing regions occupied by the specified main roads from the input target region mask. As shown in Fig. 16, our model can regenerate the urban layout constrained on the specified main roads. For example, in the 1st row of Fig. 16, Result 1 produced by the user-specified Main Road 1 can preserve the ring-style main road well. Result 2 has a very different regenerated road layout due to the change in the user-specified roads in Main Road 2. As a result, the building layout is also changed to match with the regenerated urban road layout.

Table 5 Quantitative results of urban road layout / complete urban layout regeneration by ablating key components of our model

Metrics	FID ↓	GS ↓	MS-SSIM ↑
w/o FA	25.90/36.67	30.28/49.55	0.68/0.64
w/o Position Attention	18.58/35.46	14.59/46.70	0.79/0.64
w/o $\mathcal{L}_j$	19.97/36.51	18.40/48.08	0.81/0.63
w/o $\mathcal{L}_l$	19.16/33.97	16.24/42.51	**0.84**/0.65
w/o $\mathcal{L}_r$	18.31/31.89	13.07/39.32	0.81/0.67
w/o $\mathcal{L}_p$	17.66/31.67	10.70/38.44	0.83/0.66
w/o $\mathcal{L}_a$	17.70/31.07	11.42/37.06	0.82/0.67
Ours	**17.58/30.57**	**10.32/35.70**	0.83/**0.68**

Best results are highlighted in bold

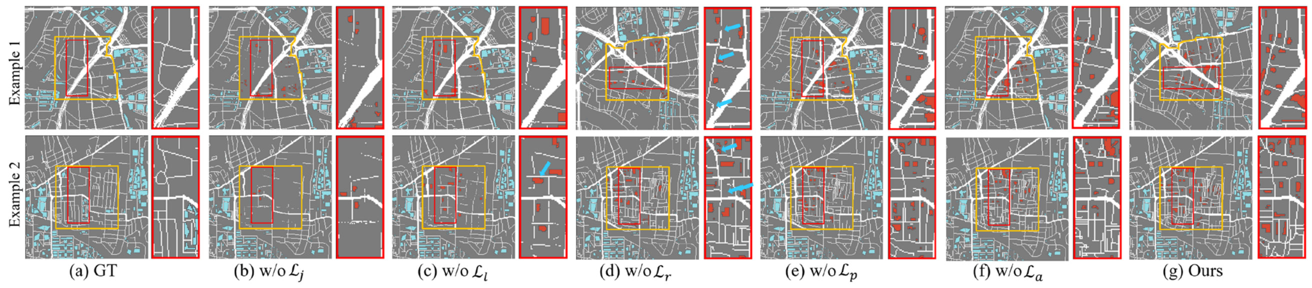

Fig. 14 Ablation study on different losses. The blue arrow in **c** highlights an incorrectly generated red building, which is located at a crossroad. The blue arrows in **d** highlight places where the road width is inconsistent, *e.g.*, it may suddenly become wider near a junction (Color figure online)

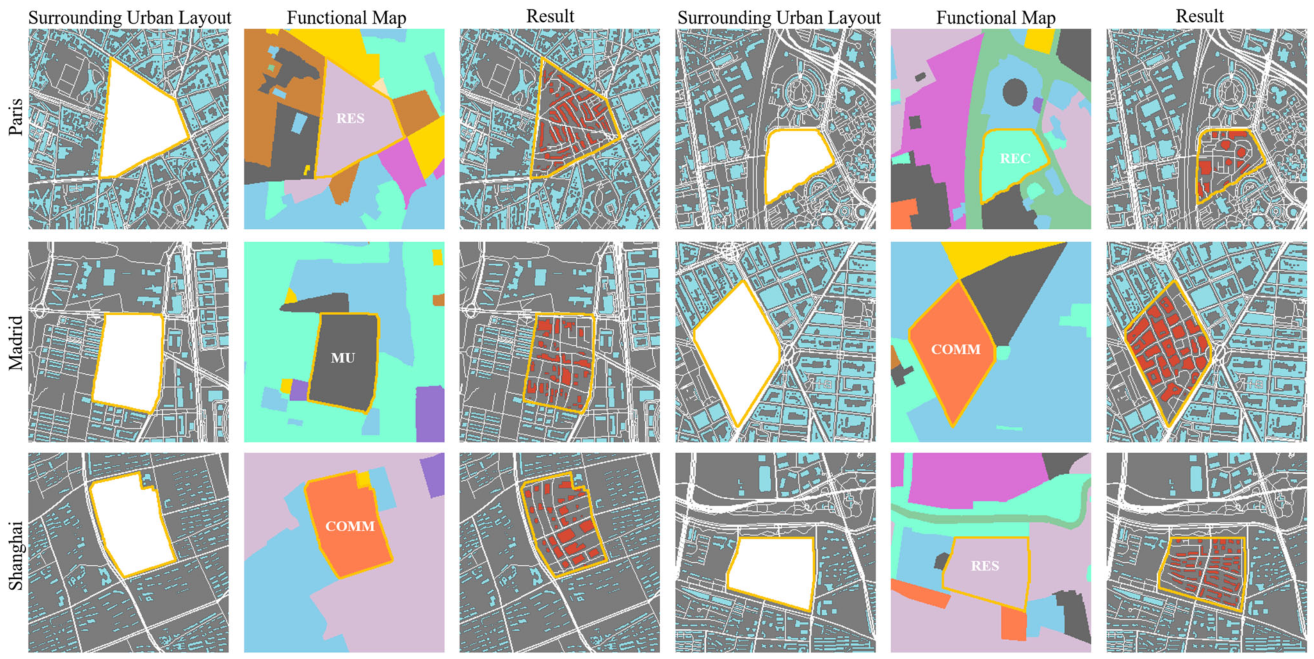

Fig. 15 Regenerated urban layouts of other cities, Paris (top), Madrid (middle), and Shanghai (bottom)

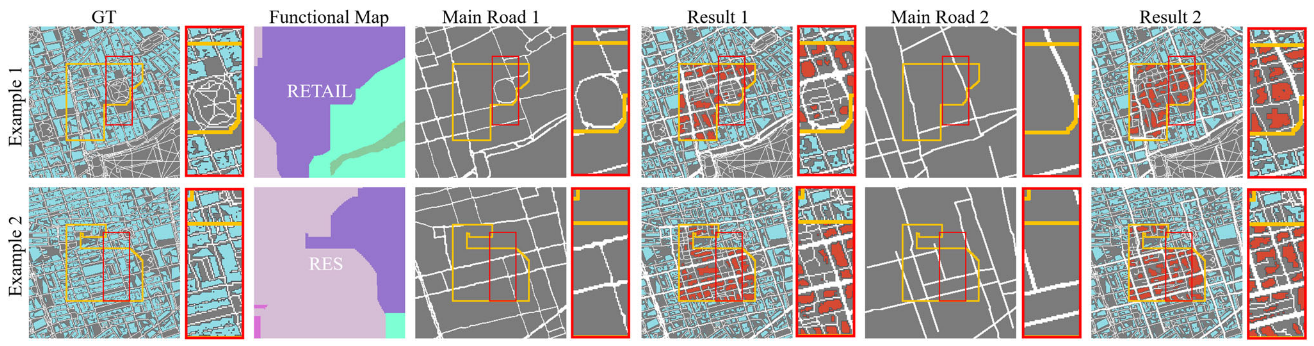

Fig. 16 Interactive urban layout regeneration, controlled by user-specified main roads. Main Road 1 and Main Road 2 are two scenarios of user-specified roads, while Result 1 and Result 2 are the corresponding regenerated urban layouts

6.3 Conditional Sketch Inpainting

We consider the application of our model beyond urban regeneration to conditional sketch inpainting. In this experiment, we train our model on the Sketch dataset (Eitz et al., 2012) and irregular mask dataset (Kargly, 2017). We use the object class as the functional map and the unfinished sketch as the surrounding context. The results are shown in Fig. 17. In general, our method can complete the unfinished parts of the sketches reasonably well.

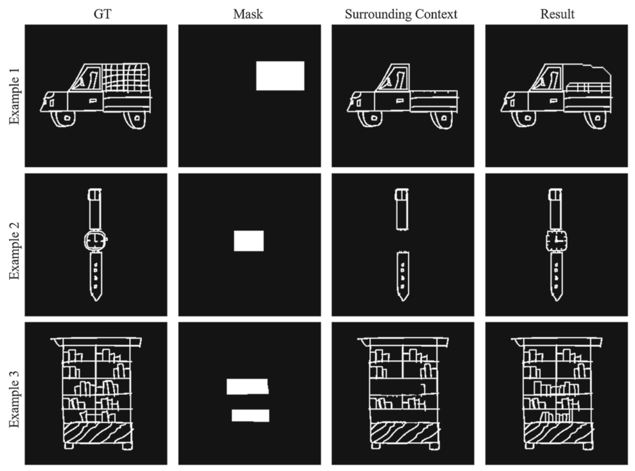

Fig. 17 Sketch inpainting. In general, our model can complete the unfinished sketches well

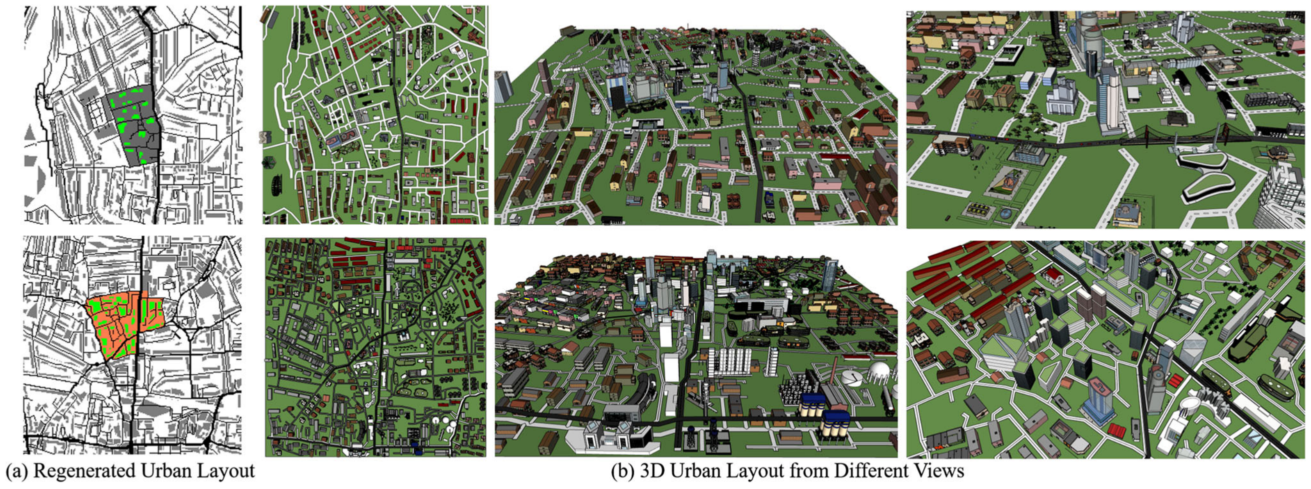
(a) Regenerated Urban Layout
(b) 3D Urban Layout from Different Views

Fig. 18 Modeling 3D urban layout from the regenerated urban layout. **b** shows different views of the assembled 3D city models, given the regenerated urban layout from our model **a**

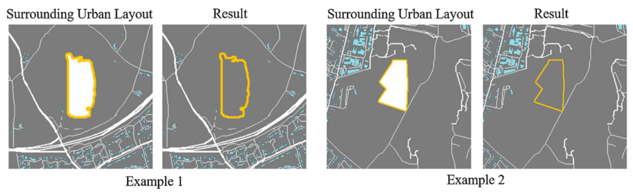

Fig. 19 Failure case. When there is insufficient context, our method may fail to regenerate urban layouts

6.4 Modeling 3D City

Compared to 2D urban layout, 3D urban layout can provide more realistic scenes with richer details, which may allow city planners, government officials and other participants to view and manipulate in a more intuitive manner (Chen, 2011). Based on our regenerated urban layout (*i.e.*, including urban road and urban building layouts), we can easily place 3D models through SketchUp[1] to build a 3D city model, as shown in Fig. 18. We collect a lot of 3D models as a 3D model dataset, and then use SketchUp to construct a 3D city from these models, according to our regenerated urban layout. For the 3D buildings in the 3D model dataset, we collect many prefab 3D building models and categorize them based on their functions. As a result, the building types and heights are indirectly determined by the function. The 3D road models are generated by Street Generator (str, 2009) based on the regenerated road layouts.

7 Conclusion

In this paper, we target to address a new task called function-aware urban layout regeneration. The core design of our proposed model UrbanEvolver is to generate a function-aware valid urban layout with the function-layout adaptive block and a set of geometry-level and pixel-level losses. A large-scale urban layout dataset is collected for training and supporting further research. Experiments have demonstrated the effectiveness of our model. Our automatic urban layout regeneration method significantly reduces the costs of urban layout regeneration.

Our method also has limitations. When there is insufficient context, our approach may fail to regenerate urban layouts, as shown in Fig. 19. Besides, since our dataset covers mostly of flat terrains, our method may not produce urban layouts that best represent landscapes of mountains and hills. In addition, our method is also not able to regenerate detailed urban road layout information, such as traffic signs, traffic lights and bus stops. As a future work, we are looking into ways to regenerate urban layouts with more details.

Acknowledgements This work was supported by the National Key Research and Development Program of China under grant number 2022YFC2407000, the Interdisciplinary Program of Shanghai Jiao Tong University under grant number YG2023LC11 and YG2022ZD007, National Natural Science Foundation of China under grant number 62272298 and 62077037.

Data Availability The datasets that support the current study are available from the corresponding author on reasonable request. We will release the dataset and code in the following link, https://github.com/LittleQBerry/UrbanEvolver.

[1] www.sketchup.com.

References

Aliaga, D. G., Vanegas, C. A., & Benes, B. (2008). Interactive example-based urban layout synthesis. *ACM Transactions on Graphics, 27*(5), 1–10.

Amazon Web Services. (2016). SpaceNet dump retrieved from https://registry.opendata.aws/spacenet

Amirtahmasebi, R., Orloff, M., Wahba, S., et al. (2016). *Regenerating urban land: A practitioner's guide to leveraging private investment*. Washington, D.C: Urban Development, World Bank Publications.

Atkins, C. B. (2008). Blocked recursive image composition. In *ACM multimedia* (pp. 821–824).

Belli, D., Kipf, T. (2019). Image-conditioned graph generation for road network extraction. In *NeurIPS Workshops*.

Benny, Y., Galanti, T., Benaim, S., et al. (2021). Evaluation metrics for conditional image generation. *International Journal of Computer Vision, 129*(5), 1712–1731.

Brock, A., Donahue, J., Simonyan, K. (2019). Large scale GAN training for high fidelity natural image synthesis. In *ICLR*.

Campen, M., Bommes, D., & Kobbelt, L. (2012). Dual loops meshing: Quality quad layouts on manifolds. *ACM Transactions on Graphics, 31*(4), 1–11.

Chang, K. H., Cheng, C. Y., Luo, J., et al. (2021). Building-GAN: Graph-conditioned architectural volumetric design generation. In *ICCV* (pp. 11,956–11,965).

Chen, G., Esch, G., Wonka, P., et al. (2008). Interactive procedural street modeling. *ACM Transactions on Graphics*. https://doi.org/10.1145/1399504.1360702

Chen, R. (2011). The development of 3d city model and its applications in urban planning. In *ICG* (pp. 1–5).

Chen, Y., Li, H., He, B., et al. (2015). Multi-objective genetic algorithm based innovative wind farm layout optimization method. *Energy Conversion and Management, 105*, 1318–1327.

Chen, Z., Ma, X., Yu, W., et al. (2021). Measuring the similarity of building patterns using graph Fourier transform. *Earth Science Informatics, 14*, 1953–1971.

Choi, Y., Uh, Y., Yoo, J., et al. (2020). Stargan v2: Diverse image synthesis for multiple domains. In *CVPR* (pp. 8185–8194).

Chu, H., Li, D., Acuna, D., et al. (2019). Neural turtle graphics for modeling city road layouts. In *ICCV* (pp. 4522–4530).

Eitz, M., Hays, J., & Alexa, M. (2012). How do humans sketch objects? *ACM Transactions on Graphics, 31*(4), 1–10.

Fisher, M., Savva, M., & Hanrahan, P. (2011). Characterizing structural relationships in scenes using graph kernels. *ACM Transactions on Graphics, 30*(4), 34. https://doi.org/10.1145/2010324.1964929

Fisher, M., Ritchie, D., Savva, M., et al. (2012). Example-based synthesis of 3D object arrangements. *ACM Transactions on Graphics, 31*(6), 1–11.

Fisher, M., Savva, M., Li, Y., et al. (2015). Activity-centric scene synthesis for functional 3D scene modeling. *ACM Transactions on Graphics, 34*(6), 1–13.

Fu, J., Liu, J., Tian, H., et al. (2019). Dual attention network for scene segmentation. In *CVPR* (pp. 3146–3154).

González-Morcillo, C., Martin, V., Fernandez, D. V., et al. (2010). Gaudii: An automated graphic design expert system. In *IAAI* (pp. 1775–1780).

Groenewegen, S. A., Smelik, R. M., de Kraker, K. J., et al. (2009). Procedural city layout generation based on urban land use models. In *Eurographics (Short Paper)* (pp. 45–48).

Guo, S., Jin, Z., Sun, F., et al. (2021a). Vinci: An intelligent graphic design system for generating advertising posters. In *ACM CHI* (pp 1–17).

Guo, X., Yang, H., & Huang, D. (2021b). Image inpainting via conditional texture and structure dual generation. In *ICCV* (pp. 14,134–14,143).

Gupta, K., Lazarow, J., Achille, A., et al. (2021). Layouttransformer: Layout generation and completion with self-attention. In *ICCV* (pp. 984–994).

Hartmann, S., Weinmann, M., Wessel, R., et al. (2017). Streetgan: Towards road network synthesis with generative adversarial networks. In *ICCECGVCV*.

Henderson, P., Subr, K., & Ferrari, V. (2017). Automatic generation of constrained furniture layouts. arXiv:1711.10939

Heusel, M., Ramsauer, H., Unterthiner, T., et al. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS* (pp. 6626–6637).

Hu, R., Huang, Z., Tang, Y., et al. (2020). Graph2plan: Learning floorplan generation from layout graphs. *ACM Transactions on Graphics, 39*(4), 118.

Hudson, D., & Zitnick, L. (2021). Compositional transformers for scene generation. In *NeurIPS* (pp. 9506–9520).

Johnson, J., Alahi, A., & Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. In *ECCV* (pp. 694–711).

Jyothi, A. A., Durand, T., He, J., et al. (2019). Layoutvae: Stochastic scene layout generation from a label set. In *ICCV* (pp. 9894–9903).

Kargly. (2017). Qd-imd: Quick draw irregular mask dataset. https://github.com/karfly/qd-imd/

Kelly, T. (2021). Cityengine: An introduction to rule-based modeling. *Urban Informatics*. https://doi.org/10.1007/978-981-15-8983-6_35

Khrulkov, V., & Oseledets, I. (2018). Geometry score: A method for comparing generative adversarial networks. In *ICML* (pp. 2626–2634).

Lechner, T., Ren, P., Watson, B., et al. (2006). Procedural modeling of urban land use. In *ACM SIGGRAPH (Poster)*.

Lee, D., Liu, S., Gu, J., et al. (2018). Context-aware synthesis and placement of object instances. In *NeurIPS* (pp. 10,414–10,424).

Lee, D. T., & Lin, A. K. (1986). Generalized Delaunay triangulation for planar graphs. *Discrete Computational Geometry, 1*(3), 201–217.

Lenormand, M., Picornell, M., Cantú-Ros, O. G., et al. (2015). Comparing and modelling land use organization in cities. *Royal Society Open Science, 2*(12), 150,449.

Li, J., Yang, J., Hertzmann, A., et al. (2019). Layoutgan: Generating graphic layouts with wireframe discriminators. In *ICLR (Poster)*.

Lipp, M., Scherzer, D., Wonka, P., et al. (2011). Interactive modeling of city layouts using layers of procedural content. *Computer Graphics Forum, 30*(2), 345–354.

Liu, G., Reda, F., Shih, K., et al. (2018). Image inpainting for irregular holes using partial convolutions. In *ECCV* (pp. 89–105).

Liu, H., Jiang, B., Song, Y., et al. (2020). Rethinking image inpainting via a mutual encoder-decoder with feature equalizations. In *ECCV* (pp. 725–741).

Merrell, P., Schkufza, E., & Koltun, V. (2010). Computer-generated residential building layouts. *ACM Transactions on Graphics, 29*(6), 181. https://doi.org/10.1145/1882261.1866203

Merrell, P., Schkufza, E., Li, Z., et al. (2011). Interactive furniture layout using interior design guidelines. *ACM Transactions on Graphics, 30*(4), 1–10.

Mi, L., Zhao, H., Nash, C., et al. (2021). Hdmapgen: A hierarchical graph generative model of high definition maps. In *CVPR* (pp. 4227–4236).

Nauata, N., Chang, K. H., Cheng, C. Y., et al. (2020). House-GAN: Relational generative adversarial networks for graph-constrained house layout generation. In *ECCV* (pp. 162–177).

Nauata, N., Hosseini, S., Chang, K. H., et al. (2021). House-GAN++: Generative adversarial layout refinement network towards intelligent computational agent for professional architects. In *CVPR* (pp. 13,632–13,641).

Nazeri, K., Ng, E., Joseph, T., et al. (2019). Edgeconnect: Generative image inpainting with adversarial edge learning. In *ICCV Workshops*.

Nishida, G., Garcia-Dorado, I., & Aliaga, D. G. (2016). Example-driven procedural urban roads. *Computer Graphics Forum, 35*(6), 5–17.

OpenStreetMap Contributors. (2017). Map features. https://wiki.openstreetmap.org/wiki/Map_features

OpenStreetMap Contributors. (2017). Planet dump. Retrieved from https://planet.osm.org

Ovsjanikov, M., Ben-Chen, M., Solomon, J., et al. (2012). Functional maps: A flexible representation of maps between shapes. *ACM Transactions on Graphics, 31*(4), 1–11.

Owaki, T., & Machida, T. (2020). Roadnetgan: Generating road networks in planar graph representation. In *ICONIP* (pp. 535–543).

O'Donovan, P., Agarwala, A., & Hertzmann, A. (2014). Learning layouts for single-pagegraphic designs. *IEEE Transactions on Visualization and Computer Graphics, 20*(8), 1200–1213.

Pang, X., Cao, Y., Lau, R. W., et al. (2016). Directing user attention via visual flow on web designs. *ACM Transactions on Graphics, 35*(6), 1–11.

Panozzo, D., Block, P., & Sorkine-Hornung, O. (2013). Designing unreinforced masonry models. *ACM Transactions on Graphics, 32*(4), 1–12.

Parish, Y., & Müller, P. (2001). Procedural modeling of cities. In *ACM SIGGRAPH* (pp. 301–308).

Park, T., Liu, M. Y., Wang, T. C., et al. (2019). Semantic image synthesis with spatially-adaptive normalization. In *CVPR* (pp. 2337–2346).

Pautrat, R., Lin, J. T., Larsson, V., et al. (2021). Sold^2: Self-supervised occlusion-aware line description and detection. In *CVPR* (pp. 11,368–11,378).

Peng, C. H., Yang, Y. L., & Wonka, P. (2014). Computing layouts with deformable templates. *ACM Transactions on Graphics, 33*(4), 1–11.

Qiao, X., Zheng, Q., Cao, Y., et al. (2019). Tell me where i am: Object-level scene context prediction. In *CVPR* (pp. 2633–2641).

Ripon, K. S. N., Glette, K., Khan, K. N., et al. (2013). Adaptive variable neighborhood search for solving multi-objective facility layout problems with unequal area facilities. *Swarm and Evolutionary Computation, 8*, 1–12.

Ryu, D. S., Chung, W. K., & Cho, H. G. (2010). Photoland: A new image layout system using Spatio-temporal information in digital photos. In *ACM SAC* (pp. 1884–1891).

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *ICLR*.

(2009). Street generator. https://sketchucation.com/forums/viewtopic.php?f=180&t=19410

Sushko, V., Gall, J., & Khoreva, A. (2021). One-shot GAN: Learning to generate samples from single images and videos. In *CVPR Workshops* (pp. 2596–2600).

Suvorov, R., Logacheva, E., Mashikhin, A., et al. (2022). Resolution-robust large mask inpainting with Fourier convolutions. In *WACV* (pp. 3172–3182).

Suzuki, S., et al. (1985). Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing, 30*(1), 32–46.

Szegedy, C., Vanhoucke, V., Ioffe, S., et al. (2016). Rethinking the inception architecture for computer vision. In *CVPR* (pp. 2818–2826).

Ueno, M., & Satoh, S. (2021). Continuous and gradual style changes of graphic designs with generative model. In *IUI* (pp. 280–289).

Umetani, N., Igarashi, T., & Mitra, N. J. (2012). Guided exploration of physically valid shapes for furniture design. *ACM Transactions on Graphics, 31*(4), 86.

Wang, Z., Simoncelli, E., & Bovik, A. (2003). Multiscale structural similarity for image quality assessment. In *ACSSC* (pp. 1398–1402).

Weber, B., Müller, P., Wonka, P., et al. (2009). Interactive geometric simulation of 4d cities. *Computer Graphics Forum, 28*(2), 481–492.

Yang, X., Mei, T., Xu, Y. Q., et al. (2016). Automatic generation of visual-textual presentation layout. *ACM Transactions on Multimedia Computing, Communications, and Applications, 12*(2), 1–22.

Yang, Alhalawani S., Yl Wonka, P., et al. (2014). What makes London work like London? *Computer Graphics Forum, 33*(5), 157–165.

Yang, Y. L., Wang, J., Vouga, E., et al. (2013). Urban pattern: Layout design by hierarchical domain splitting. *ACM Transactions on Graphics, 32*(6), 1–12.

Yu, L. F., Yeung, S. K., Tang, C. K., et al. (2011). Make it home: Automatic optimization of furniture arrangement. *ACM Transactions on Graphics, 30*(4), 86.

Žalik, B. (2001). Merging a set of polygons. *Computers and Graphics, 25*(1), 77–88.

Zhang, H., Goodfellow, I., Metaxas, D., et al. (2019). Self-attention generative adversarial networks. In *ICML* (pp. 7354–7363).

Zhang, P., Li, C., & Wang, C. (2020). Smarttext: Learning to generate harmonious textual layout over natural image. In *ICME* (pp. 1–6).

Zhang, W., Zhu, J., Tai, Y., et al. (2021). Context-aware image inpainting with learned semantic priors. In *IJCAI* (pp. 1323–1329).

Zhang, Y., Li, X., Wang, A., et al. (2015). Density and diversity of OpenStreetMap road networks in China. *Journal of Urban Management, 4*(2), 135–146.

Zhao, K., Han, Q., Zhang, C. B., et al. (2021). Deep hough transform for semantic line detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 44*, 4793–4806.

Zheng, X., Qiao, X., Cao, Y., et al. (2019). Content-aware generative modeling of graphic design layouts. *ACM Transactions on Graphics, 38*(4), 1–15.

Zheng, Y. (2015). Methodologies for cross-domain data fusion: An overview. *IEEE Transactions on Big Data, 1*(1), 16–34.