GlobalMapper: Arbitrary-Shaped Urban Layout Generation

Liu He
Purdue University
he425@purdue.edu

Daniel Aliaga
Purdue University
aliaga@purdue.edu

Overview

Below is a summary of the contents in each section of this supplemental material:

- Sec. 1: Details of calculating building shape type and occupancy ratio.
- Sec. 2: Details of graph representation and canonical spatial transformation.
- Sec. 3: Details of synthesis for building footprint generation.
- Sec. 4: Details of dataset collection.
- Sec. 5: Ablation study of our model.
- Sec. 6: Comparisons to Diffusion Models.
- Sec. 7: User study on generation similarity.
- Sec. 8: Semantic manipulations on building row number.
- Sec. 9: Generation using random interpolation.
- Sec. 10: Dead-end roads scenarios.
- Sec. 11: Applications to Urban Weather Forecast Model.
- Sec. 12: Using our 28 cities, we show additional results of random layout generation, controllable generation, and a large-scale map.

1. Building Shape Determination

We select 4 shape types (*Rectanglar, L-shape, U-shape, and X-shape*) that well represent most buildings. As Fig. 1 shows, for each building type, we use Powell's method to find the best parameters set (e_{1-10}) to achieve the highest IoU between the parameterized template and the building contour. Then we choose the shape type with the highest IoU as the building shape s_i, which was described in main paper Sec. 3.1. Following the simplification envelopes notion [7], these shapes provide a close envelope around all buildings in our dataset (average building diameter = 58.16m) with average **Hausdorff distance** (an estimation of contour deviation showed in upper-right of Fig. 1) **of only 1.36m (stddev = 2.53m) and IoU = 95.78%**. The occupancy ratio a_i is calculated by the building area over the area of the oriented bounding box. This metric is also used in the synthesis step described in main paper Sec. 3.3 and Supplemantal Sec. 3.

2. Canonical Spatial Transform

Instead of the original geographic coordinate system, we utilize a main axis based coordinate system for the canonical spatial transformation. In Fig. 2, similar to representation scheme used in [10], we first calculate the 2D skeletonization of the city block polygon and then we modify the two end sections of the longest path in the skeleton to obtain the main axis. The length W of the main axis is considered as the width of the block. We calculate the distance from the centroid of each building, H_i, to the main axis (y_i'), and its distance (x_i') along the main axis from the starting point. The distance of a building centroid from the main axis is also referred to as the minor axis. We label block height H as twice the value of the mean minor axis length; i.e., $H = 2 \cdot mean(H_i)$. Both block width W and height H are utilized to normalize building position (x_i', y_i') and size (w_i, h_i). The values after normalization will be the attributes for G' which is the input to our method.

After the canonical spatial transformation, we heuristically match each building (node) in the city block to our 2D grid graph ($[4, 30], N = 120$). For clarity, a detailed graph showing the connectivity of an example G' is presented in Fig. 3. Meanwhile, the row number of the block is also given by the building (node) matching (1, 2, 3, or 4 rows).

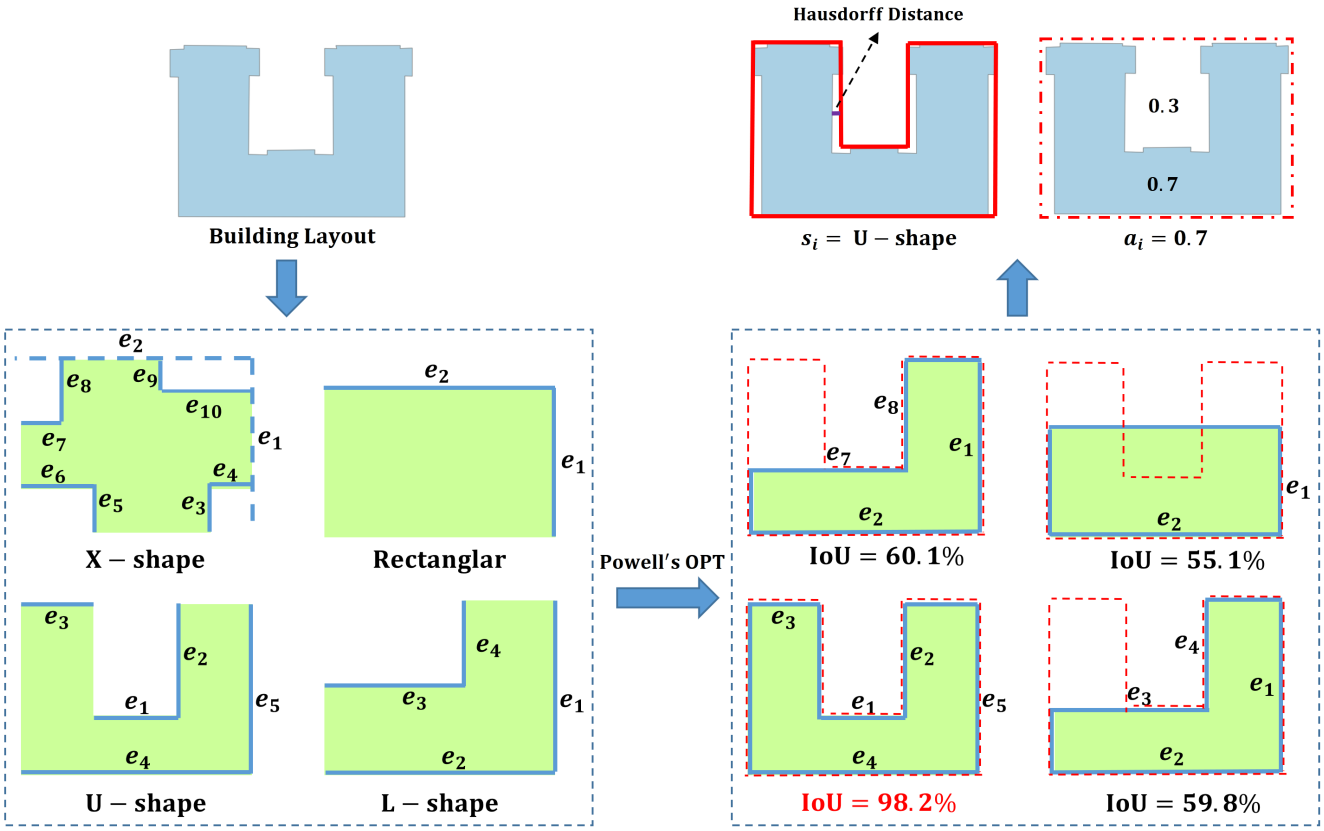

Figure 1. **Building Shape Determination.** Powell's method is utilized for searching the parameter set (e_{1-10}) achieving the highest IoU between the parameterized template and the building contour. The shape type obtaining the highest IoU is determined as the building shape type input feature s_i. (The parameterized templates, and corresponding IoUs, are sketched visualizations and calculations of Powell's optimization process.)

3. Building Footprint Synthesis

We use our parameterized building shape functions (*Rectangular, L-shape, U-shape, X-shape*) to obtain synthetic footprints. Fig. 4 shows two examples of our building footprint synthesis process. The shape parameters of each generated building of given type $\hat{s}_i$ are randomly perturbed within a predefined range (e.g. for U-shape in Fig. 4, $\frac{e_1}{e_2} \in [0.5, 1.5]$) until a configuration best satisfying the generated occupancy value a_i is found within a maximum number of iterations. Then, the generated building footprint is rotated so that its width dimension is again parallel to the block's main axis (as Fig. 2 shows).

4. Dataset Collection

We obtained our datasets by downloading 2D layouts from OpenStreetMap (OSM) [8]. We chose 28 large cities having good OSM coverage: Chicago, Washington D.C., New York City, Atlanta, Dallas, Los Angeles, Miami, Seattle, Boston, Providence, Baltimore, San Diego, San Francisco, Portland, Milwaukee, Minneapolis, Austin, New Orleans, Phoenix, Denver, Toronto, Vancouver, Austin, Houston, as well as Pittsburgh, Tampa, San Jose, and Norfolk. For the first 24 cities, we selected about 75% of the metropolitan area for training and evaluation. In addition, we collected additional testing regions from the remaining parts of each city. The last four cities are not in our training or validation datasets and thus are effectively used for out-of-distribution testing. In the paper and in this supplemental section, we present visual results from the testing regions (including the last four cities). Overall, we observe that the performance of our method is consistent across all cities.

We downloaded both the road networks and building footprints from OSM. We projected all geographic coordinates to World Geodetic System 1984 (WGS 84) coordinate reference systems. Then, we searched for all simple cycles in the graph of road networks. Each simple road cycle represents a city block. We used each simple cycle (city block) to find contained polygonal shapes (i.e., building footprints). Both city blocks and building footprints are the input for further graph rep-

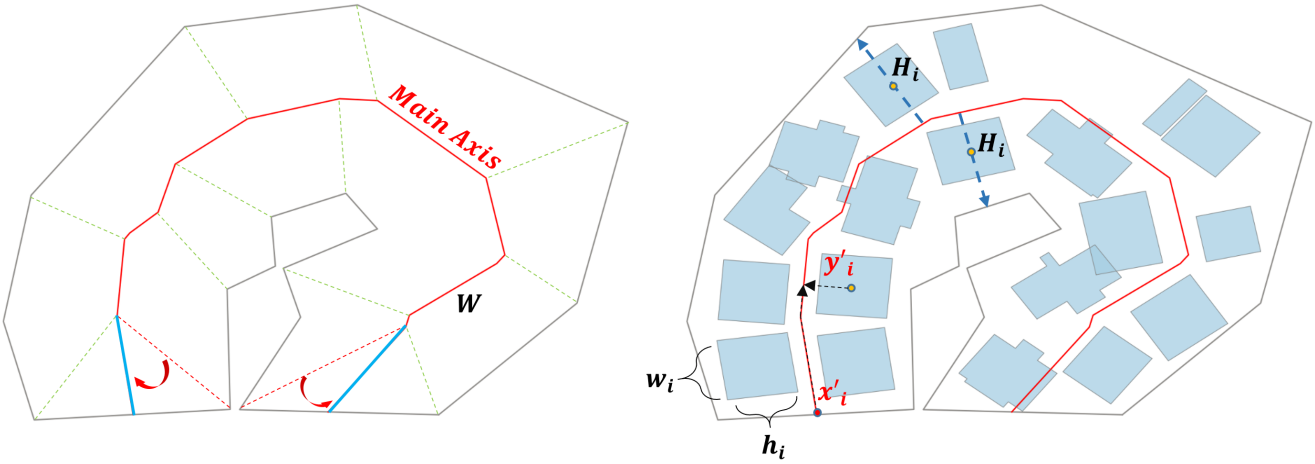

Figure 2. **Canonical Spatial Transform. Left:** To calculate the main axis, we first simplify the city block contour to a polygon (i.e., the shown black contour). Then we find the 2D skeleton of the polygon, which is represented by the show interior dashed lines and the red solid lines. We search for the longest path of the skeleton indicated by red solid line and red dash lines. Finally, we modify two end segments to point to the middle of the opposing block edges (i.e., blue solid line). The length of the main axis is the block width W. **Right:** We calculate the distance from the centroid of each building to the main axis (y'_i), and its distance alongside the main axis from the starting point (x'_i). The distance alongside the vertical line intersecting each building's centroid H_i is also collected. Twice the mean value is the building to main axis distances is the building height $H = 2 \cdot mean(H_i)$. Both block width W and height H are utilized to normalize building position (x'_i, y'_i) and size (w_i, h_i). The values after normalization will be the attributes for G' which is the input to our method.

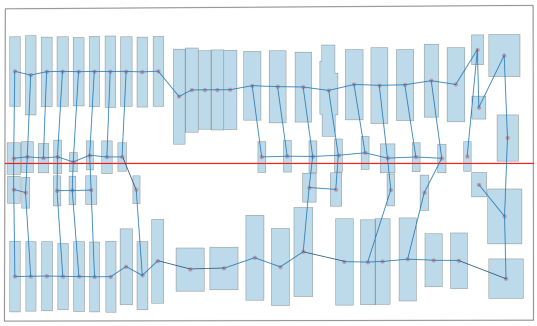

Figure 3. **Graph Representation Visualization.** Visualization of the edge connectivity of an example input graph G'. The nodes marked as non-existing and edges connecting to it are not plotted. After our canonical spatial transformation, we heuristically match each building (i.e., node) in the city block to our 2D grid graph ($[4, 30], N = 120$). The main axis is marked as a horizontal red line.

resentation setup and initial training.

The distribution of total building numbers in each city block is presented in Fig. 5 and the distribution of number of building rows (heuristically calculated as shown in Fig. 3) is presented in Fig. 6. These distributions motivate our selection of a maximum number of buildings per city block (120) and the maximum number of rows (4). Using the 28 cities, we attempt, to the best of our knowledge, to cover all city block types in North America in order to benefit the generalization capability of our method. This enables generating high quality content as well as large-scale maps (e.g., Fig. 13).

5. Additional Ablation Study

We present an additional ablation study on model structures and topology of graph representation, shown in Tab. 2. Overlap index [6] is the percentage of total overlapping area among generated building layouts within a city block. The Out-Block index is the percentage of generated building layout area that is outside of the city block contour. Position error (Pos-E.) is the average percentage of building position error expressed as a percentage of city block's diagonal. Coverage error (Cov-E.) is the error of total building coverage compared to the ground truth. Count error (Ct-E.) is the error of total building count compared to the ground truth.

In all cases, our current method utilized a $4x30$ 2D grid graph ($N = 120$). For node existence, a one-hot

	LayoutVAE [5]	BlockPlanner[13]	VTN [1]
Ours	**98.0%** / 2.0%	**100.0%** / 0.0%	**100.0%** / 0.0%

Table 1. **Additional User Study on Similarity.** We show the results of an additional user study on reconstruction similarity. Values in cell indicate percentage of replies supporting our method over a prior method.

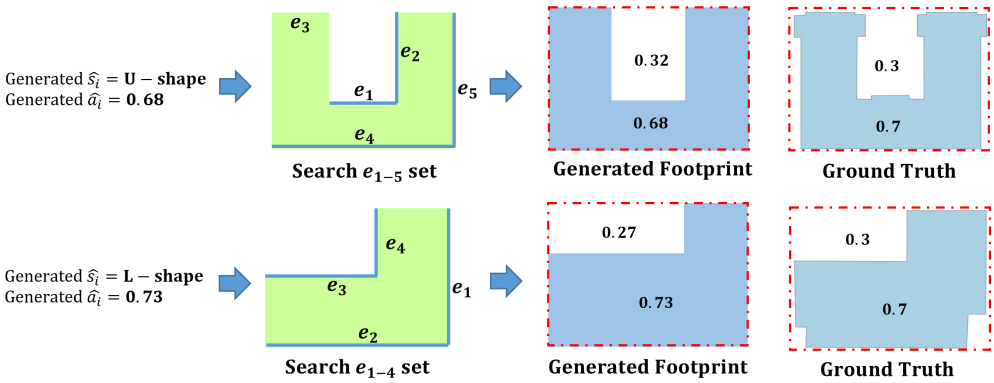

Figure 4. **Building Footprint Synthesis.** Generated features $\{\hat{s}_i, \hat{a}_i\}$ are utilized to synthesize building footprints with corresponding shape type and also occupancy ratio. Our goal is to produce building footprints that are similar (but not necessarily identical) to the original building coverages, positions, and shapes.

Ablations	Overlap↓	Out-Block↓	Pos-E.↓	Cov-E.↓	Ct-E.↓
Ours-Meanpool	1.74	1.42	4.46	1.50	1.71
Ours-No-Onehot	2.54	1.26	5.61	2.50	4.68
Ours-128Latent	1.83	1.17	4.89	1.13	0.46
Ours-256Latent	1.54	1.50	4.63	0.68	0.26
Ours-1024Latent	1.19	**1.17**	3.40	0.29	0.15
Ours-4Head	1.37	1.32	4.19	0.89	0.70
Ours-8Head	1.35	1.44	3.67	**0.074**	0.090
Ours-Ring-Topo	10.39	1.32	14.00	3.67	9.73
Ours-2Row-Topo	1.47	1.42	4.60	0.86	0.37
Ours*	**1.06**	1.25	**3.10**	0.36	**0.072**

Table 2. **Additional Ablation Study.** We report reconstruction metrics (all in %) among various alternative designs. Discussion of these graph representations and model designs are in Sec. 5.

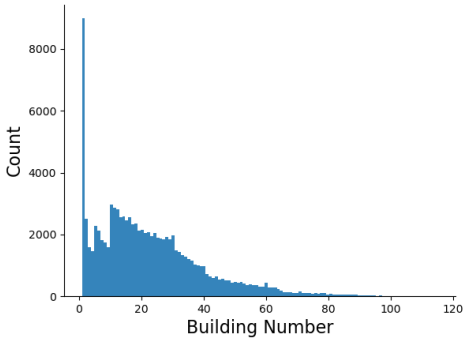

Figure 5. **Building Number Distribution:** The distribution of total building numbers per city block.

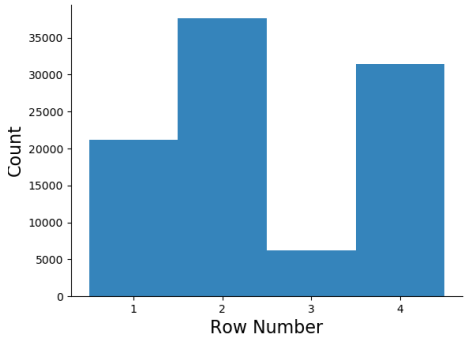

Figure 6. **Row Number Distribution:** The distribution of row numbers per city block.

encoding of node index ($0 \sim 119$) is concatenated. For both encoder and decoder, our model structure stacks 3 graph attention layers [11] with 12 heads. We implemented maximum pooling after each graph attention layer, and we chose the latent vector dimension as 512.

For each ablation setting in Tab. 2, we only change one component of our proposed model in order to evaluate its influence on reconstruction. As shown, max pooling outperforms mean pooling ("Meanpool"). The one-hot encoding helps to obtain an accurate

	L-Sim↑	Ovlp↓ (%)	O-Blk↓ (%)	FID↓	WD↓ (*bbx*)	WD↓ (*ct*)
LayoutDM [4]	16.69	5.54	8.59	48.61	4.94	8.56
Liu *et al.* [3]	12.35	1.69	3.98	56.93	7.64	12.31
Ours (Tab.1)	**22.45**	**1.42**	**0.89**	**14.94**	**1.45**	**0.06**

Table 3. **Quantitative Comparisons to DMs.** We generate 1000 urban layouts and compare to the same amount of real urban layouts. Best values are in bold.

building count prediction ("No-Onehot"). Longer latent length of z does improve performance on OutBound index and Coverage, but decreases other metrics ("128, 256, 1024Latent"). Thus the medium length 512 is chosen as our final model. More attention heads benefits all metrics with decreasing growth rate ("4, 8Head"). We chose 12 heads to keep the balance of model parameter size and performance. We also experimented with a ring graph topology as [13], but it performed poorly on all metrics ("Ring-Topo"). As an alternative, we also experimented with a 2D grid graph of size 2×60 ("2Row-Topo"), which proved to be worse than our final setting of 4×30.

6. Comparisons to Diffusion Models

Under the same comparison scheme described by Sec.4.2 in our paper. We provide quantitative comparisons to diffusion models [4, 3] trained with our datasets. Tab. 3 extends Tab.1 of our paper with DMs, and shows we still outperform.

7. Additional User Study on Similarity

In addition to the realism user study in the main paper, we conducted a user study for similarity using our method and three existing methods (LayoutVAE [5], BlockPlanner [13], and VTN [1]) all trained with our dataset. The study was performed back-to-back with the same user group as our realism study. The reconstruction task is to generate an urban layout similar to a given real urban layout. The study is performed in a two-alternative forced choice (2AFC) manner. We generated 18 comparisons, each containing a layout generated by our method and the same layout generated by one of the three prior methods (thus 6 layout pairs for each prior method). The urban locations do not overlap across the three sets. In addition to the 18 binary choice questions, we included 2 random duplicate questions for quality checking. Users that answered differently to the same question were discarded as outliers.

During the study, we presented two candidate urban layouts side-by-side from different methods on the left side of the screen, meanwhile the reference real layout was presented on the right side of the screen as a reference. Users were asked to choose which candidate layout looks more similar to the given real world reference. Replies from 50 valid users are collected and synthesized in Tab. 1. Our method clearly outperforms other existing methods. In each test, at least 98.0% of users are in favor of our method over other prior works.

8. Semantic Manipulations

Given our well-trained model, we conducted a study of urban layout style control in latent space. The experiment is similar to the face manipulation application in [9]. We found that a certain extent of disentanglement exists between different styles of urban layouts (e.g., building arrangement in city blocks of a different number of rows). As an example, we utilized our validation dataset and labeled each urban layout by the number of rows (i.e. 1, 2, 3, or 4 in our case). We encoded those layouts to the latent space, and found the cluster centers of each of the four row-number groups. Then, we calculated the direction vectors between pairs of cluster centers. Moving a latent vector in the direction from the 1-row group cluster to the 4-row group cluster corresponds to splitting buildings into smaller ones and adding rows of buildings. In Fig. 7, we show the style changing from 1-row style to 2, 3, or 4-row styles, and vice versa. A similar style manipulation could be performed on building counts, building sizes, and other features as needed by future applications.

9. Interpolations

Our method can generate urban layouts by interpolating latent space between two given urban layouts. In Fig. 8, we show layouts resulting from linearly interpolating both building layout latent vector, and block shape latent vector from one to another. The intermediate layouts correspond to an intuitive style interpolation.

10. Dead-end Roads

As Fig. 9 shows, our method can handle common dead-end roads, such as cul-de-sac's , but not all cases of dead-end roads. The quality of our canonical spatial transformation directly influences the positioning

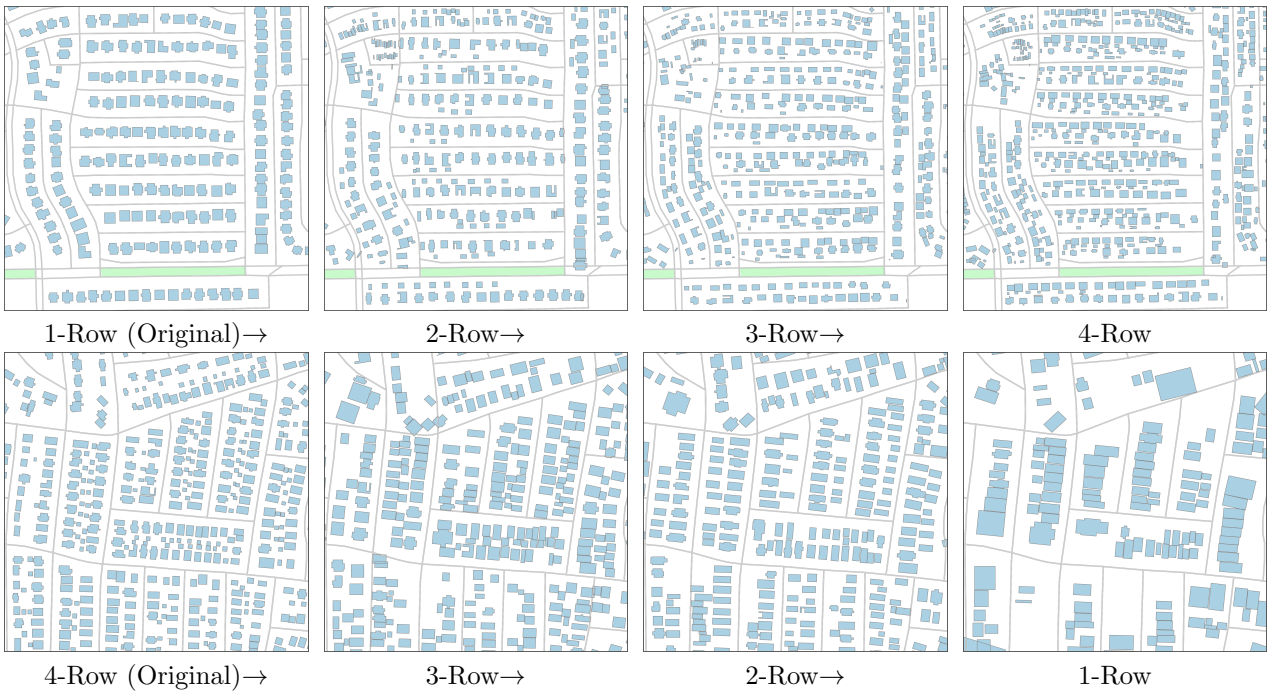

Figure 7. **Semantic Manipulation by Row Number: First Row:** We select a region that mostly contains 1-row style city blocks. By gradually moving its latent vector towards the 4-row cluster center, we manipulate its original style from 1-row to close to 4-rows. **Second Row:** We select a region that mostly contains 4-row style city blocks. Similarly, by gradually moving its latent vector towards the 1-row cluster center, we change the original style from 4-rows to about 1-row.

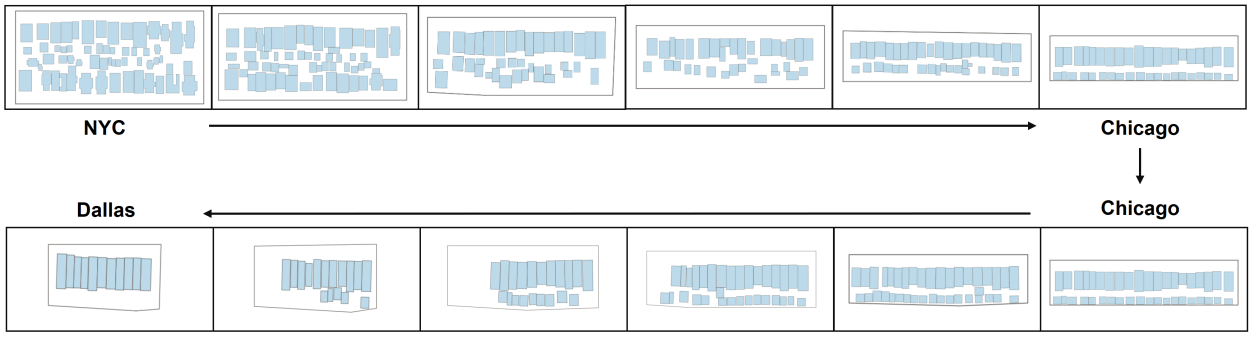

Figure 8. **Interpolation.** Our method is able to simultaneously interpolate between different block shapes (and scale) and different building layouts by linear interpolation. Our method generates smoothly changing interpolations (Rectangular bounding box indicates block shape and scale interpolation as an invariant reference.)

of buildings around dead-end roads. Owing to irregular dead-end roads (as the second row in Fig. 9), concave and complex block contour may result in ugly main axis during 2D skeletonization in Fig. 9, and the imperfect main axis will further influence the quality of the inverse spatial transformation. Thus some building footprints will overlaps the dead-end roads in the final output maps.

11. Urban Weather Forecast Applications

We implement sparse prior generation as described in main paper Sec. 4.4. Based on the generated 3D building mass models of the entire Chicago, we calculate a selected set of urban canopy parameters as recommended by [12]. They are plan area ratio, area-weighted building height, building surface to plan area ratio, and building height distribution (detailed definitions in [12]). Those parameters are the input of a

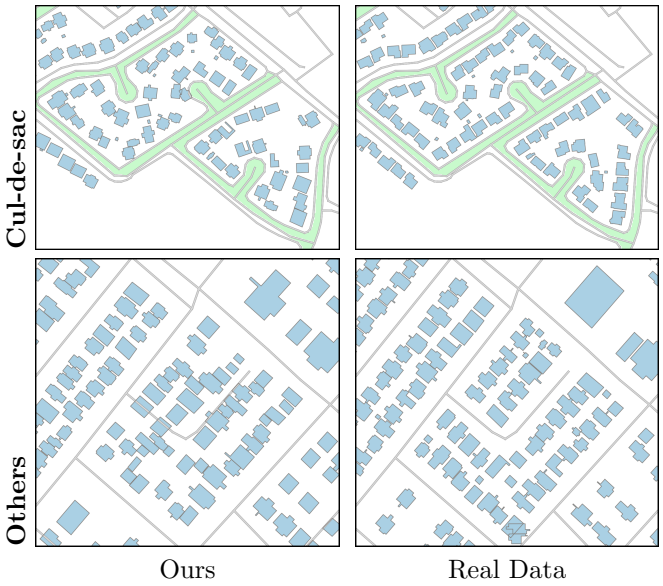

Figure 9. **Dead-end Roads.** Our method can handle common cul-de-sac (the first row), but not other irregular dead-end road scenarios (the second row). The main impact factor is the quality of our canonical spatial transformation.

given urban weather forecast system (WRF [2]) to simulate wind speed and surface temperature.

The visualizations of our generated urban canopy parameters are showed in Fig. 10, and accuracy evaluations of our results compared to the ground truth are showed in Fig. 11. Our method generates vector-based 3D building mass models which can be rasterized into any given resolution. This is a contribution to urban meteorology academia where common spatial resolution is coarser than 500m. Moreover, our method can produce plausible urban canopy parameters given only small portions of input prior (e.g., 5%). This benefits the data equity problem in medium or small cities where complete datasets are not available.

The final WRF simulation results (during a winter date, 12/10/2011) are showed in Fig. 12. The simulation model is based on our 3D building mass model generated from only 5% prior data yielding a similar simulation result to that produced using ground truth. Our average per-pixel wind-speed prediction error is 0.23m/s, and average per-pixel surface temperature prediction error is 0.11 C°.

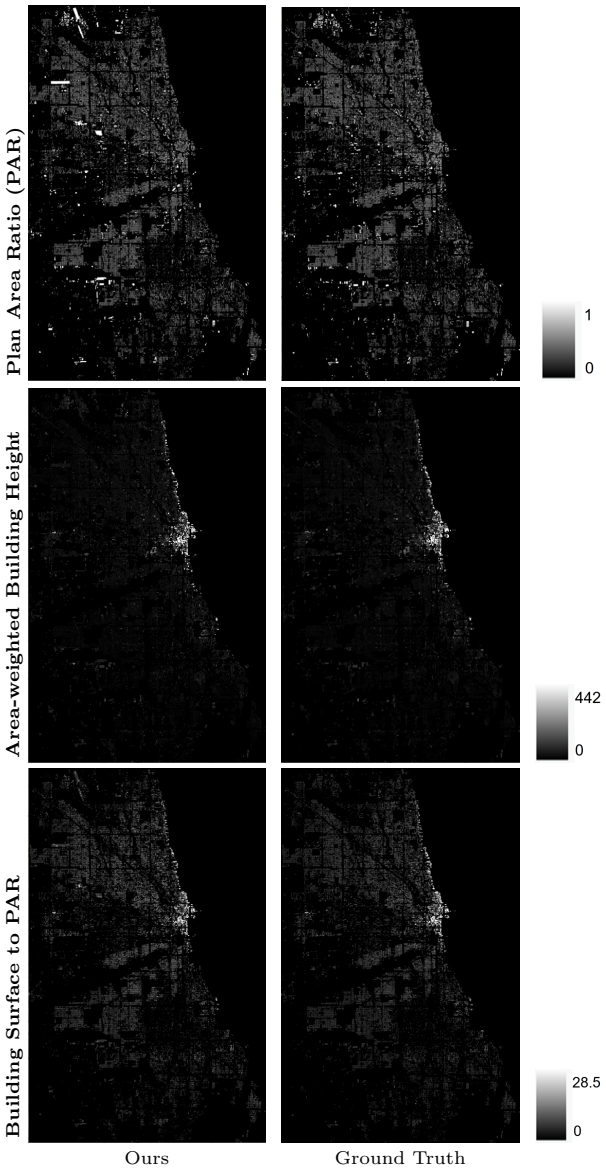

Figure 10. **Urban Canopy Parameter Results.** We show the qualitative comparisons between our generated urban canopy parameters to the ground truth (pixel resolution = 50m). Our method produces plausibly similar results compared to the ground truth.

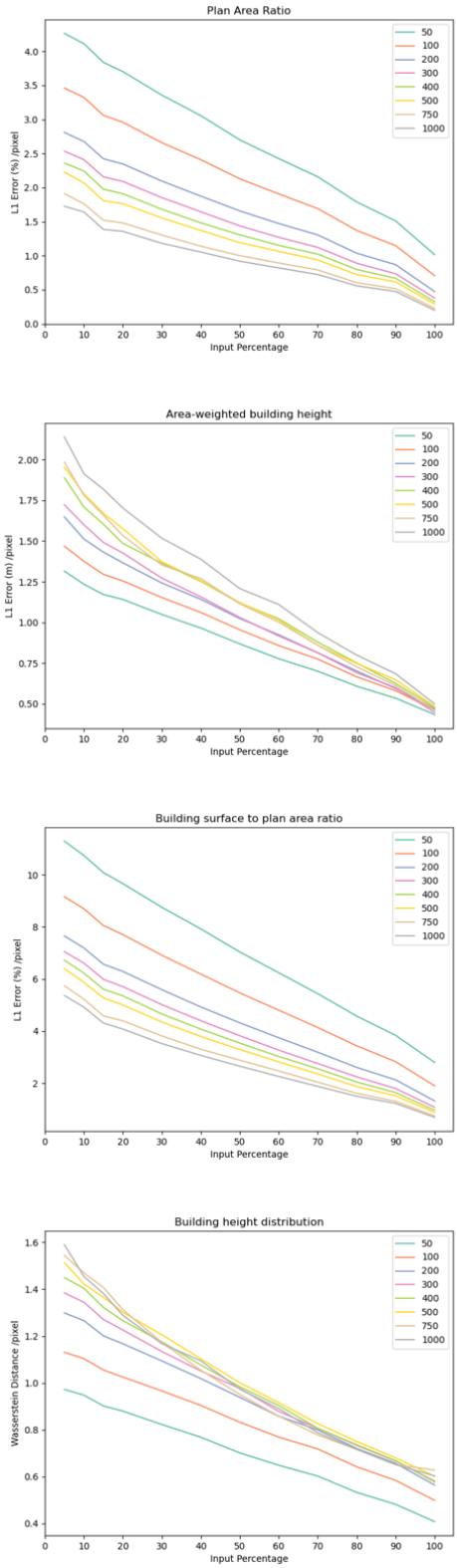

Figure 11. **Accuracy over resolution and given percent of prior.** We show the quantitative comparisons between our generated urban canopy parameters to the ground truth under a range of spatial resolution (50m 1km).

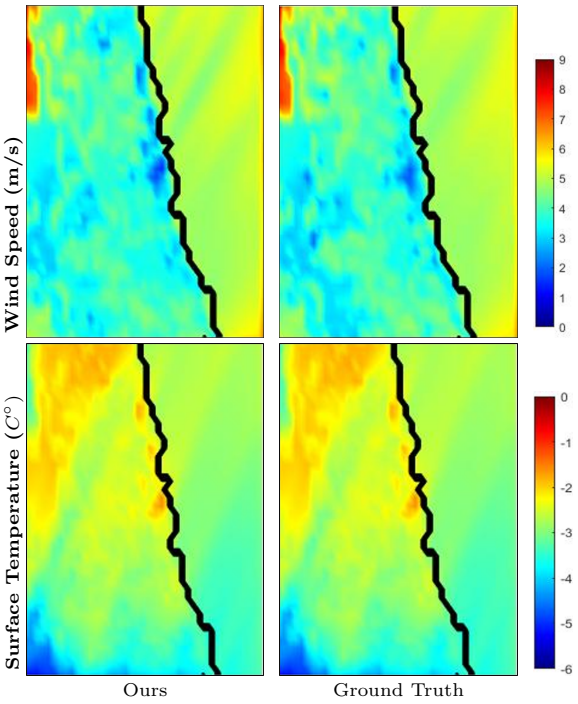

Figure 12. **Urban Weather Forecast Simulation**. We use our generated 3D building mass model (given only 5% of prior) to drive an urban weather forecast model (WRF [2]). The simulation results for wind speed and surface temperature show nearly identical results compared to the ground truth. Our average per-pixel wind-speed prediction error is 0.23m/s, and average per-pixel surface temperature prediction error is 0.11 C°.

12. Controllable Generation Results

We show many *generated* urban layouts based on the test regions of our multiple cities. Fig. 13 contains a large-scale generated map based on the test region of Vancouver. Fig. 14 shows a zoom-in of the red box in the large-scale map. These figures show the potential of our method for city-scale generation. To generate this large-scale map, we randomly select a few percent of the city blocks in the test region and use those to obtain a narrow latent distribution of the latent space (i.e., in Fig. 13 we select 1%). Then, we sample from that distribution in order to populate the entire map. Our generated building layouts are all realistic and similar to the real OSM layouts bounded by green bounding boxes (as in Fig 13 and Fig. 14).

Fig. 15 contains close-ups of a fragment of the test region of 8 different cities. For each city, we generate a big-map in a similar way as described in the prior paragraph. Fragments are cropped as about $\frac{1}{16}$ size of a zoom-in view similar to Fig. 14. Building layouts are diverse in building counts, size, and arrangement across the many city blocks. But all generated layouts are realistic and feasible. These examples aim to demonstrate our realistic and diverse generation ability of urban layouts.

For applications that require *similarity* to a given urban layout (e.g., data augmentation, urban design), we provide three experiments to demonstrate controllable generation (shown for 5 cities, including 3 test-only cities – they were not used in training) in Fig. 16. The controllable generation aims to generate layouts that are similar to a given pattern, but also keep realism.

For the first experiment ("Prior"), we provide the urban layout (i.e. prior) to the encoder for each city block and then generate a corresponding layout, effectively performing a reconstruction task. In this case, high similarity between "Prior" and "Real Data" implies that our method is able to capture and reproduce the styles and details of a large variety of urban layouts.

For the second experiment ("Noised Prior"), we add Gaussian noise to each learned prior. This enables introducing a controllable amount of randomness in building size, count, and arrangement within each city block. The similarity of the ("Noised Prior") column to "Real Data" column depends on the amount of noise added.

For the third experiment ("Random Interpolation"), we generate new latent codes for each city block by interpolating between the latent space priors of two random blocks. This process will force novel content to be created but ensuring a level of similarity to the "Real Data" column.

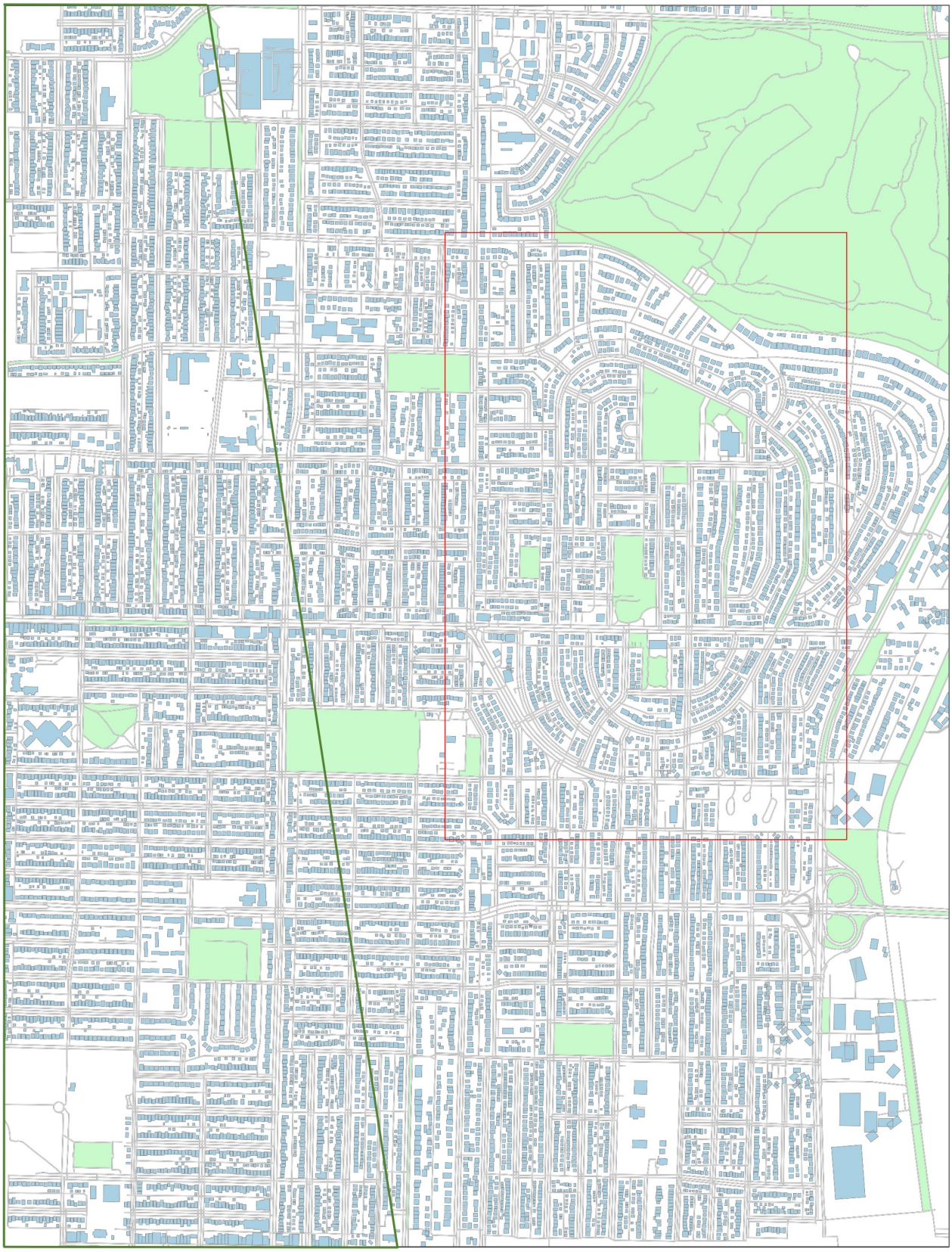

Figure 13. **Large-scale Map (mosaic with real OSM).** We show a large-scale map generated by our method. A zoom-in view of the layout in the red box is in Fig. 14. Green bounding box: Real OSM layouts. The rest: Our generated layouts.

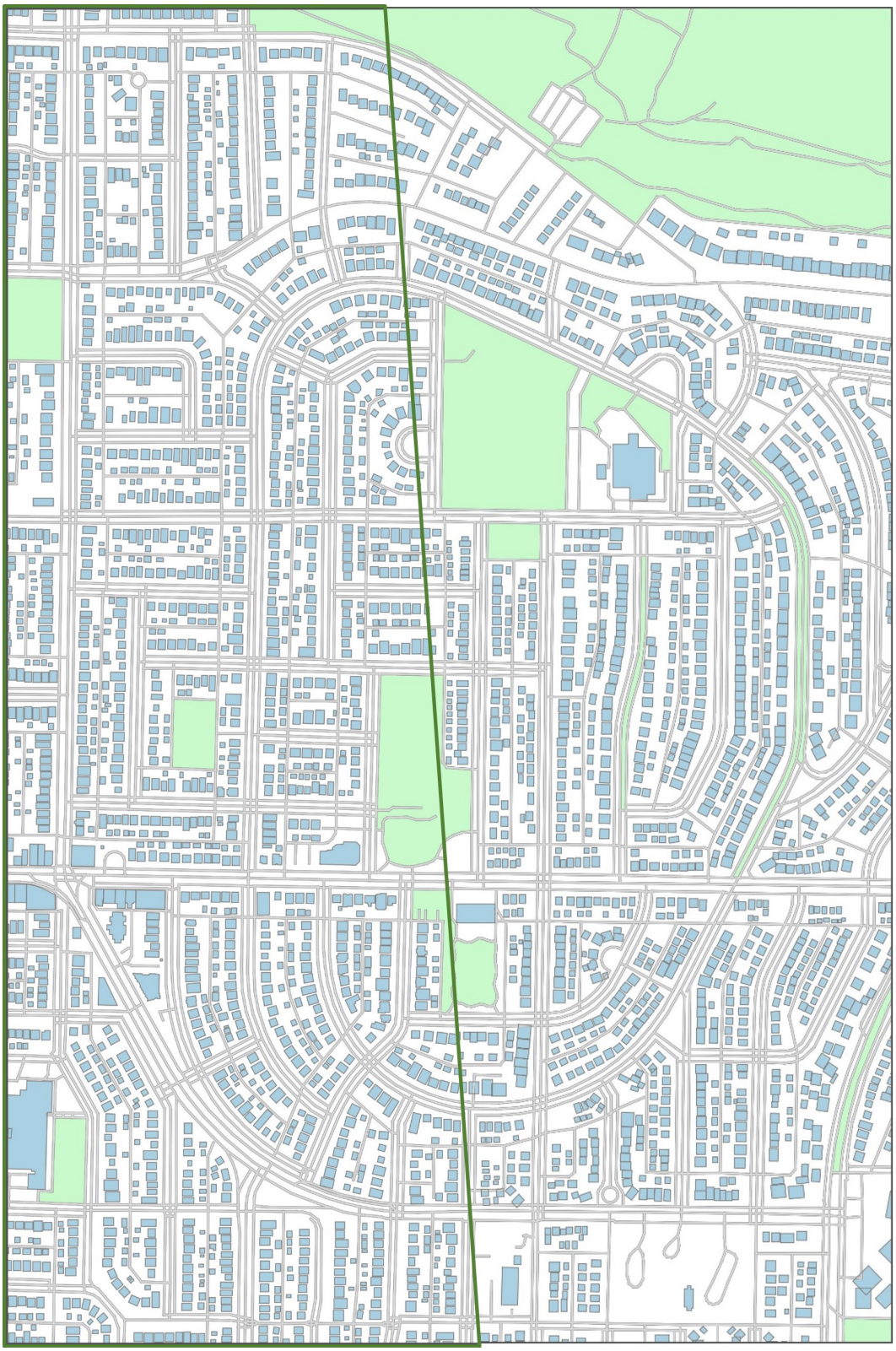

Figure 14. **Zoom-in of Large-Scale Map (mosaic with real OSM).** The zoom-in view of the red box in Fig. 13. Green bounding box: Real OSM layouts. The rest: Our generated layouts.

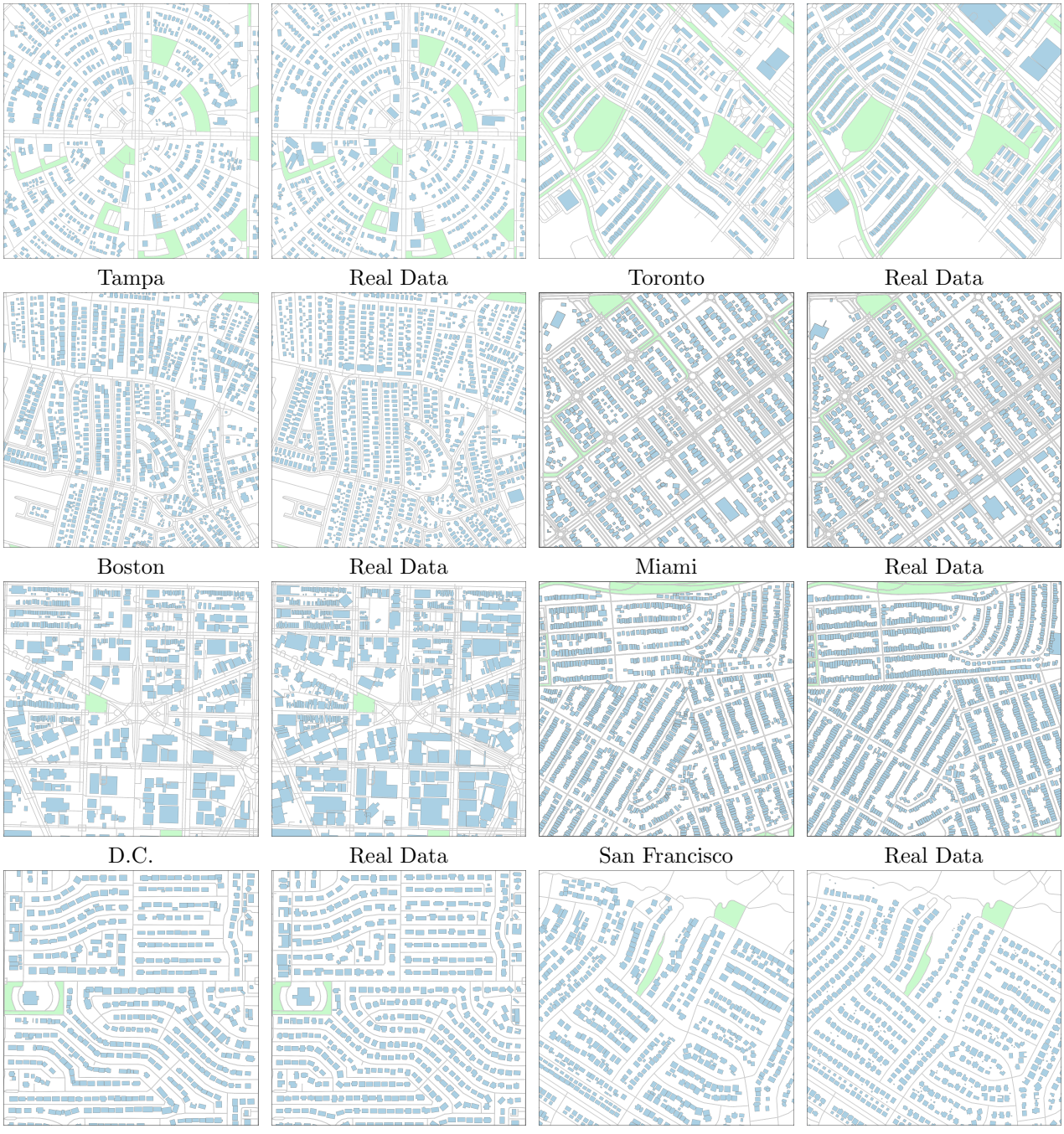

Figure 15. **Generated Layouts.** Given arbitrary road networks from our testing regions, we generate urban layouts for 8 cites. The random generation aims to keep realism and diversity. Building layouts are diverse in building counts, size, and arrangement across different urban blocks but all generated layouts are plausible for the given road network.

Figure 16. **Controlled Generation of Layouts:** Given arbitrary road networks, we generate urban layouts by reconstruction from the learned priors ("Prior"), by adding Gaussian noise to learned priors ("Noised Prior"), and by random interpolation between all possible pairs of learned priors ("Random Interpolation"). The controllable generation aims to generate layouts that are similar to a given pattern, but also keep realism. The real data are presented for reference. This figure includes Pittsburgh, Norfolk, San Jose, Dallas, and New York City.

References

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

[2] Fei Chen, Hiroyuki Kusaka, Robert D. Bornstein, Jason Ching, C. Sue B. Grimmond, Susanne Grossman-Clarke, Thomas Loridan, Kevin W. Manning, Alberto Martilli, Shiguang Miao, David J. Sailor, Francisco Salamanca, Haider Taha, Mukul Tewari, Xuemei Wang, Andrzej A. Wyszogrodzki, and Chao-Lin Zhang. The integrated wrf/urban modelling system: development, evaluation, and applications to urban environmental problems. *International Journal of Climatology*, 31, 2011. 7, 8

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

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

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

[6] Jianan Li, Jimei Yang, Aaron Hertzmann, Jianming Zhang, and Tingfa Xu. Layoutgan: Generating graphic layouts with wireframe discriminators. *arXiv preprint arXiv:1901.06767*, 2019. 3

[7] David Luebke, Benjamin Watson, Jonathan D. Cohen, Martin Reddy, and Amitabh Varshney. *Level of Detail for 3D Graphics*. Elsevier Science Inc., USA, 2002. 1

[8] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2017. 2

[9] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9243–9252, 2020. 5

[10] 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

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

[12] Michael Mau Fung Wong, Jimmy Chi Hung Fung, Jason Ching, Peter Pak Shing Yeung, Jason Wai Po Tse, Chao Ren, Ran Wang, and Meng Cai. Evaluation of uwrf performance and modeling guidance based on wudapt and nudapt ucp datasets for hong kong. *Urban Climate*, 28:100460, 2019. 6

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