EUROGRAPHICS 2024 / A. Bermano and E. Kalogerakis
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 43 (2024), Number 2

Hierarchical Co-generation of Parcels and Streets in Urban Modeling

Zebin Chen Peng Song F. Peter Ortner

Singapore University of Technology and Design

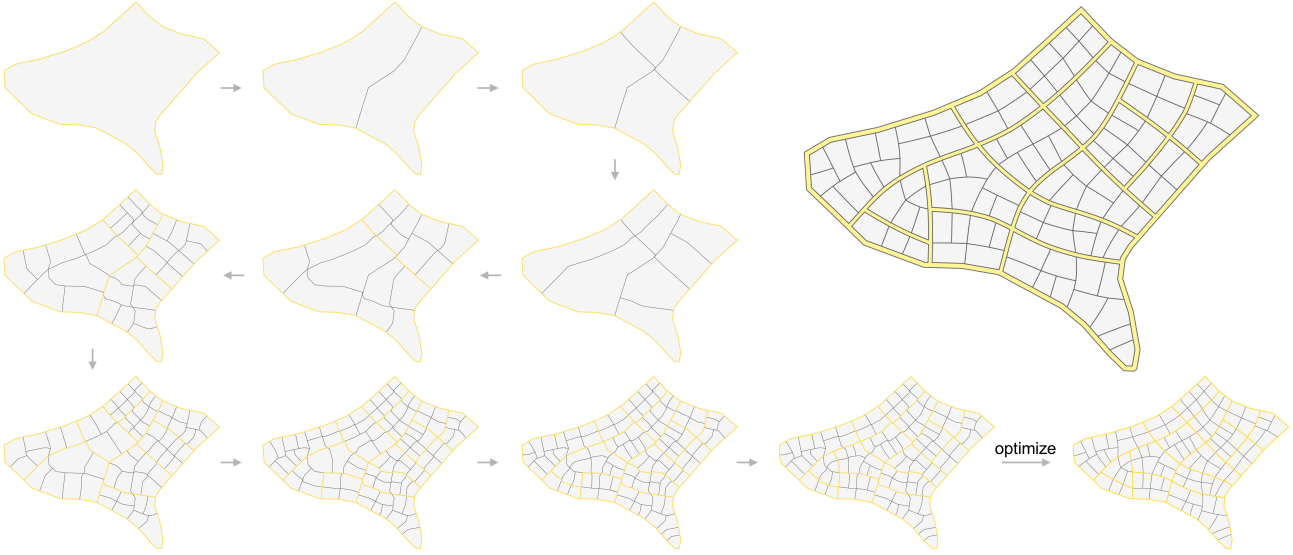

Figure 1: *Taking (top left) a polygonal land shape as an input, our approach hierarchically co-generates parcels (in grey) and streets (in yellow) guided by a set of urban design requirements. We perform a (bottom right) global optimization on the co-generated parcels and streets to improve their geometric quality.*

Abstract

We present a computational framework for modeling land parcels and streets. In the real world, parcels and streets are highly coupled with each other since a street network connects all the parcels in a certain area. However, existing works model parcels and streets separately to simplify the problem, resulting in urban layouts with irregular parcels and/or suboptimal streets. In this paper, we propose a hierarchical approach to co-generate parcels and streets from a user-specified polygonal land shape, guided by a set of fundamental urban design requirements. At each hierarchical level, new parcels are generated based on binary splitting of existing parcels, and new streets are subsequently generated by leveraging efficient graph search tools to ensure that each new parcel has a street access. At the end, we optimize the geometry of the generated parcels and streets to further improve their geometric quality. Our computational framework outputs an urban layout with a desired number of regular parcels that are reachable via a connected street network, for which users are allowed to control the modeling process both locally and globally. Quantitative comparisons with state-of-the-art approaches show that our framework is able to generate parcels and streets that are superior in some aspects.

CCS Concepts

• ***Computing methodologies*** → *Shape modeling;*

1. Introduction

Modeling urban spaces, or simply *urban modeling*, is becoming increasingly popular in computer graphics [VAW*10, STBB14] and urban planning applications. One fundamental problem in urban modeling is to model the geometry of parcels and streets. This problem is challenging since parcels and streets are highly coupled with each other in the real world. All the parcels in a certain area have to be connected by a street network such that each parcel has a street access, which in turn defines a part of the parcel's boundary shape.

Existing works address parcel modeling [VKW*12] and street modeling [CEW*08, GPMG10, BWK14] as two separate problems, to simplify the modeling process. A few research works study the problem of generating both parcels and streets. A typical approach [PM01, KM07, WMWG09, EBP*12, KMK*17] is to first model a street network and then to subdivide the induced partition of land into parcels. However, since the streets and parcels are modeled separately, the generated street network does not ensure that each parcel is reachable and the generated parcels may have irregular shapes.

In this paper, our goal is to model and optimize parcels and streets in a *unified computational framework*, aiming to overcome the above limitations of existing works. To achieve this goal, we discussed with urban planners about how they perform urban planning in the real world. From these discussions, we summarize the urban planning process with the following schema. Starting from an area of open land, an urban planner first divides the land into different regions and plans streets to connect these regions. Next, the planner subdivides each region into blocks and plans new streets to connect the blocks to the existing streets. The planner repeats this hierarchical planning process until the land is partitioned into a desired number of parcels, during which a street network is always maintained to ensure that all the parcels are reachable. In this paper, we will not differentiate among region, block, parcel since all of them can be represented as a polygonal shape that is preferred to be regular. From a geometric perspective, we consider all of them as parcels conceptually.

We have two key observations on the real-world urban planning process: 1) parcels and streets are generated *hierarchically* instead of linearly; and 2) parcels and streets are generated *simultaneously* instead of independently. Inspired by the two observations, we propose a *hierarchical* approach to *co-generate* parcels and streets from a user-specified polygonal region, guided by a family of fundamental urban design requirements; see Figure 1 (left) for an example. At each hierarchical level, new parcels are generated by binary partitioning on existing parcels, among which some new parcels may not be reachable by the current street network. In this case, new streets are subsequently generated by leveraging efficient graph search algorithms to create a street access for each unreachable parcel while ensuring that the extended street network is still geometrically connected and simple.

Our hierarchical co-generation approach outputs an urban layout with a desired number of parcels, each of which is reachable via a connected and simple street network. However, the generated parcel boundaries and streets may not have smooth shape, which is unwanted in practice. Hence, we propose a global optimization on the generated initial urban layout to further improve the geometric quality of the parcels and streets; see Figure 1 (right) for an example. Our framework allows users to control the generated urban layouts globally in many different ways, including input land shape, parcel number, parcel shape, parcel orientation, and street pattern. Thanks to our hierarchical approach, it also allows users to control specific parcels and streets locally in an intermediate urban layout generated at each hierarchical level, mimicking the real-world urban planning process.

Contributions. We make the following contributions:

- We propose a computational scheme of hierarchical co-generation of parcels and streets, ensuring that each parcel is reachable by the street network.
- We propose a global optimization on the co-generated parcels and streets represented by a mesh and a few graphs, to improve their geometric quality.
- We perform quantitative comparisons with state-of-the-art approaches [YWVW13, ABB*17, Esr23] and show that our computational framework is able to generate parcels and streets that are superior in some aspects (e.g., parcel regularity).

2. Related Work

Urban modeling. An urban space is a complex collection of buildings, parcels, blocks and neighbourhoods interconnected by streets. Researchers in computer graphics have studied various problems of modeling urban spaces, including interactive reconfiguration [ABVA08] and editing [LSWW11, ZCCB21], inverse procedural modeling [VGDA*12], geometrical and behavioral modeling [VABW09a], analyzing characteristics [AYWM14], simulating walkability [MKMA19], and placement of vegetation [NPA*22]. We refer readers to [VAW*10, STBB14, KKC18] for surveys on modeling of urban spaces. Modeling streets and parcels is a key problem in urban modeling. Although the geometry of streets and parcels are highly coupled with each other, modeling streets and modeling parcels have been studied as separate problems. Some existing works [PM01, KM07, WMWG09, EBP*12, KMK*17] first model a street network and then subdivide the induced partition of land into parcels while others [VABW09b, YWVW13] hierarchically subdivide a given land into parcels and then choose a subset of the boundary segments of the parcels to form a street network according to predefined rules/templates. Compared with the existing works, our hierarchical co-generation approach not only creates parcels and streets that are superior in some aspects, but also allows users to control the parcel and street generation process in various ways, including parcel number, parcel shape, parcel orientation, and street pattern; see Section 6.

Street modeling. Existing works on street network modeling can be classified into three classes according to the modeling approach, i.e., procedural, example-based, and optimization-based approaches. Procedural approaches generate road networks based on a set of rules, including growing-based [PM01, KM07, BWK14], template-based [SYBG02], tensor-based [CEW*08], shortest path-based [GPMG10, GPGB11], and patch-based [TB17] approaches. Example-based approaches rely on analyzing data such as existing city layouts to generate street networks similar to the

real-world counterpart. Early example-based approaches [AVB08, YS12, EVC*15, NGDA16] extract statistical information from the data for synthesizing street networks. More recently, machine learning algorithms have been developed to generate street networks represented as images [HWWK17, KM19, FJY22] or graphs [CLA*19, OM20]. Optimization-based approaches formulate the street network generation as an optimization problem. Peng et al. [PYB*16] proposed an optimization-based approach to generate street networks in four different levels (i.e., major roads, collector roads, local roads, and cul-de-sacs), aiming to satisfy user-specified high-level functional specifications on the streets. Our street modeling approach falls in the optimization-based class. Different from [PYB*16] that purely models streets but not parcels, our street modeling aims to model a street network that connects all the parcels in a given land. We leverage graph search algorithms to efficiently find such a street network, guided by the goal of minimizing the total street length and the total number of street junctions.

Parcel modeling. Procedural approaches have been developed to model parcels. Parish and Müller [PM01] assume that the blocks created by street crossings have convex and regular shape and subdivide each block into multiple parcels using a simple recursive algorithm that divides the longest edges that are approximately parallel. Kelly and McCabe [KM07] generalized the subdivision approach to support both convex and concave city blocks. Vanegas et al. [VKW*12] proposed two subdivision algorithms for parcel modeling; the first algorithm is based on an offset of the straight skeleton of a polygonal block while the second algorithm extends the subdivision algorithm in [PM01] with an adaptive spatial partitioning of a city block using oriented bounding boxes. The recursive subdivision scheme also has been used in [DC14, WMWG09, VABW09b] for modeling parcels. Different from the above works that compute a straight line for splitting a large parcel into two smaller ones, Yang et al. [YWVW13] proposed to compute a smoothly curved hyperstreamline over the cross field of a parcel for binary splitting of the parcel, which allows subdividing a parcel with complex shape. For simplicity, we abbreviate hyperstreamline as streamline in this paper. In our paper, we make use of the streamline-based splitting [YWVW13] to generate new parcels. Different from [YWVW13] that chooses a streamline for binary splitting mainly based on the streamline's geometry, we propose to choose a streamline based on the two resulting parcels' size, shape, and street access; see Equation 2.

3. Problem Formulation

We take as input an area of land represented as a 2D polygon **S**. By default, the whole boundary of the 2D polygon **S** forms a circular street. Users are allowed to mark segments on the polygon boundary as non-street. The other input from users is the minimally allowed size of a parcel. Our goal is to model parcels and streets in the polygonal region **S**. In particular, we formulate a set of fundamental design requirements on parcels and streets after a discussion with several urban planners as well as referring to urban design books [Pop15, BYK19]:

1. *Parcel shape*. The polygonal shape of each parcel should be as regular as possible.
2. *Parcel reachability*. Each parcel should have a street access such that it is reachable via the street network.
3. *Street connectivity*. A street network should be geometrically connected; i.e., isolated streets are not allowed.
4. *Street length*. The total length of a street network should be as small as possible to reduce the land occupied by streets.
5. *Street junction*. The number of junctions in the street network should be as small as possible for efficient traffic flow.
6. *Cul-de-sacs*. Cul-de-sacs is optional to appear in the street network, which can be controlled by users.

The above design requirements about parcels and streets are fundamental and general yet not exhaustive. Additional design requirements are possible to be incorporated in our computational framework such as preference on parcel orientation (e.g., for a preferable building layout arrangement); see Section 6 and Figure 16.

Modeling geometry of parcels. The geometry of a parcel is modeled as a closed polygon, with N corners and N sides; see Figure 2 (left). Each side of a parcel is modeled as a polyline with a few vertices. The shape of each parcel side is either straight or smoothly curved. When the shape is smoothly curved, it is approximated by a straight line that connects the two corresponding parcel corners, which is called an *approximate side*. All the approximate sides form an *approximate polygon* of the parcel; see the dashed polygon in Figure 2 (left). Users are allowed to interactively specify an approximate polygon for the input 2D polygon **S** if it has freeform shape. Figure 2 (right) shows a special case, where each side of the parcel is a straight line. In this case, the approximate polygon of the parcel is exactly the same as the parcel's shape.

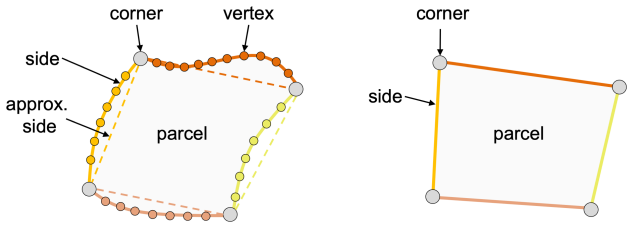

Figure 2: *Modeling the geometry of a parcel. (Left) Each side of the parcel is modeled as a polyline with several vertices, which can be approximated by a straight line that connects the two corresponding parcel corners. (Right) Each side of the parcel is a straight line that connects the two corresponding parcel corners.*

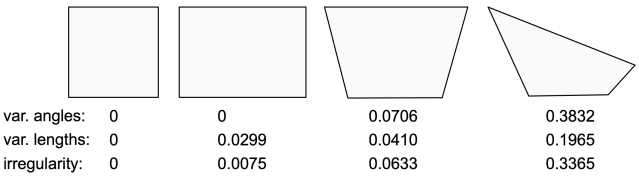

Figure 3: *Measuring shape irregularity of parcels. For each parcel, the normalized variance of interior angles, normalized variance of side lengths, and shape irregularity metric are shown in the top, middle, and bottom, respectively.*

Measuring shape irregularity of parcels. We measure the shape irregularity of each parcel using its approximate polygon due to the polygon's geometric simplicity. In particular, the shape irregularity metric I is defined as:

$$I = \gamma_1 \frac{1}{N}\sum_{i=1}^{N}(\theta_i - \bar{\theta})^2 + \gamma_2 \frac{1}{N}\frac{1}{\bar{l}^2}\sum_{i=1}^{N}(l_i - \bar{l})^2 \quad (1)$$

where θ_i is an interior angle in radian in the approximate polygon, $\bar{\theta}$ is the average interior angle of the polygon, l_i is a side length of the polygon, $\bar{l}$ is the average side length of the polygon, and γ_1 and γ_2 are two weights. By default, we set $\gamma_1 = 0.75$ and $\gamma_2 = 0.25$ in our experiments. Users are allowed to adjust these two weights in the urban modeling process. Figure 3 shows four examples polygons and their irregularity metric values, as well as the two components of the metric (i.e., normalized variance of interior angels and normalized variance of side lengths).

Representing parcels and streets. A city, or fragment of a city, consists of parcels and streets; see Figure 4(a). We assume that the centerline of all the streets is a subset of the boundaries of all the parcels. A parcel is reachable if at least one of the parcel's boundary sides is a part of the street network. We use the following data structures to represent parcels and streets in an urban layout:

- *Parcel mesh* **M**. A parcel mesh represents the geometry of parcels in an urban layout; see Figure 4(a). In a parcel mesh, each vertex is either a parcel corner or a parcel vertex (see again Figure 2) and each face represents the polygonal shape of a parcel. The valence of each parcel vertex is always equal to 2 while the valence of each parcel corner is larger than 2 when it is shared by multiple parcels.
- *Parcel corner graph.* A parcel corner graph is a simplification of the parcel mesh **M**, which keeps all the parcel corners (as well their connections) and ignores all the parcel vertices. In detail, a parcel corner graph is an undirected graph embedded in the 2D space, in which each node is a parcel corner and each edge connects two consecutive parcel corners; see Figure 4(b). Each node in the parcel corner graph is not necessary to be a corner for all the parcels sharing it. For example, node c_6 is a corner of parcels P_3 and P_4 but not parcel P_1. This is because the two incident edges of corner c_6 in parcel P_1 (i.e., edges c_5c_6 and c_6c_7) are close to collinear, forming a single side of the parcel P_1. Due to this reason, P_1's approximate polygon is a quad instead of a pentagon. Note that determining the right number of sides for a parcel's approximate polygon is critical for measuring its irregularity using Equation 1. In our experiments, we identify two consecutive edges as collinear if their included angle is larger than 135°.
- *Parcel graph.* All the parcels in the urban layout as well as their neighboring relationship are represented using a parcel graph; see Figure 4(c). In a parcel graph, each node is a parcel represented by nodes in the parcel corner graph stored in a counter-clockwise direction. Each node also stores the approximate polygon of the parcel. For example, parcel P_1 is represented as $c_5c_6c_7c_2c_1$, and its approximate polygon is stored as $c_5c_7c_2c_1$. Each edge in the parcel graph stores the edge(s) shared by the two neighboring parcels in the parcel corner graph. For example, parcels P_1 and P_4 are neighbors, and their shared edge is c_6c_7.
- *Street network graph.* A street network graph is a connected subgraph of the parcel corner graph; see Figure 4(b&d). A street network consists of a set of streets. To simplify the modeling, we assume that the shape of each street is either straight or smoothly curved. A street network graph can be decomposed into a set of streets based on the included angles between consecutive edges in the graph. In detail, two consecutive edges in the street network graph belong to the same street if their included angle is larger than 135°. For example, the street network in Figure 4(d) consists of 5 streets. Based on the identified streets, nodes in the street network graph can be classified into three classes: 1) street junction that connects two or more streets; 2) street end that forms an endpoint of a cul-de-sac; and 3) street point that connects edges in a street. The street network graph will be used to generate streets in Section 4.
- *Street graph.* In a street graph, each node is a street represented by nodes in the street network graph and each edge represents a junction that connects two streets; see Figure 4(e). For example, the orange street is represented as $c_5c_6c_7c_8c_9$ and the blue street is represented as c_5c_{12}. The junction between the two streets is the parcel corner c_5. The street graph will be used for optimizing the geometry of a street network in Section 5.

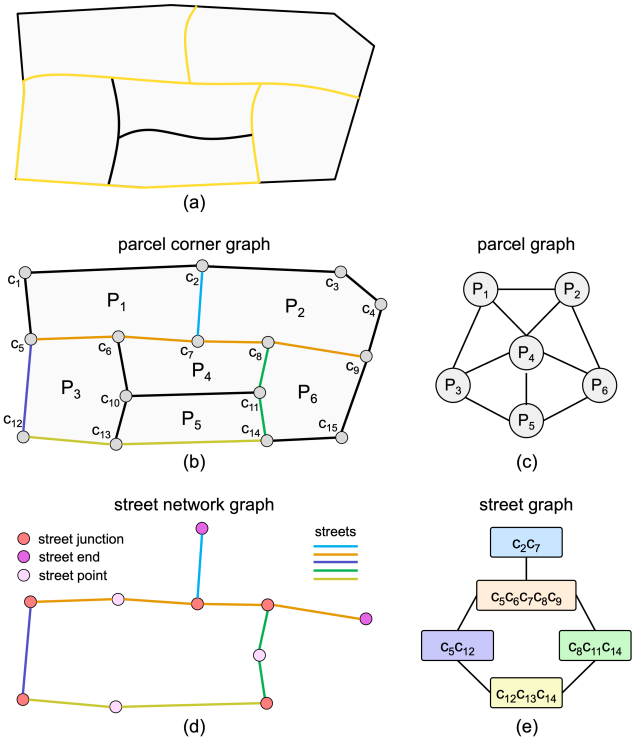

Figure 4: *Representing (a) an urban layout with parcels and streets (in yellow) using (b) a parcel corner graph, (c) a parcel graph, (d) a street network graph, and (e) a street graph. Note that the street network graph is a connected subgraph of the parcel corner graph.*

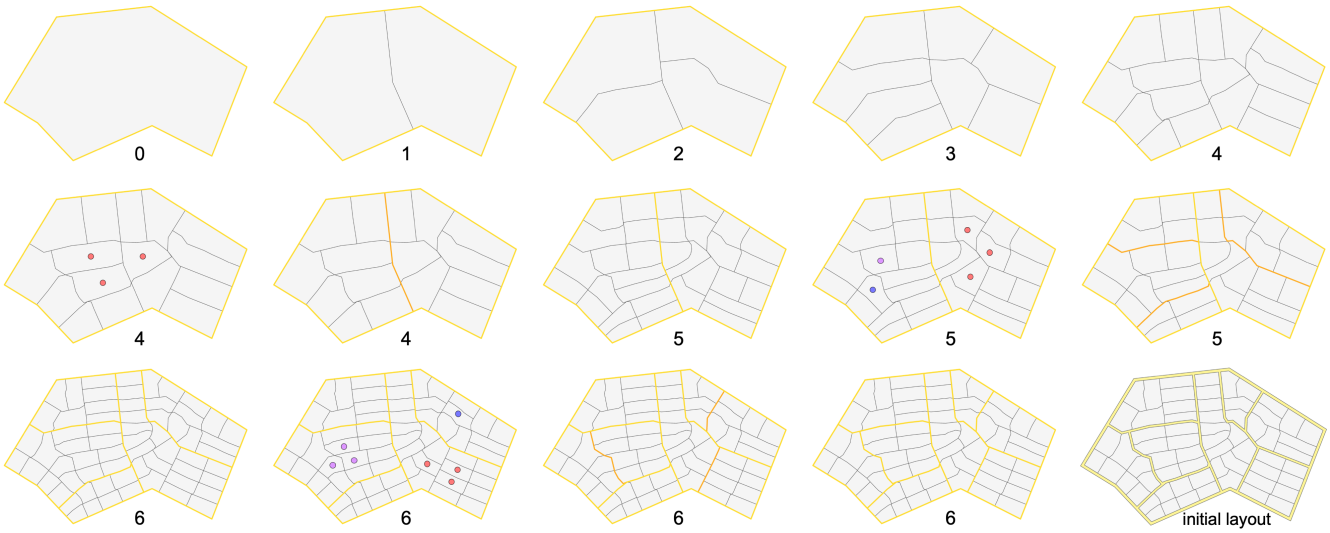

Figure 5: *Our hierarchical approach that co-generates parcels and streets. The number below each subfigure is the corresponding hierarchical level (e.g., the input polygon is at level 0). In levels 1, 2, and 3, all the new parcels are reachable by the existing street network (colored in yellow). Hence, there is no need to generate new streets. In levels 4, 5, and 6, we identify and group unreachable parcels with the existing street network, where parcels in the same group are highlighted with dots of the same color. Then, we generate new streets (colored in orange) to create a street access for each unreachable parcel.*

Overview of our approach. Our approach consists of two stages: 1) co-generate parcels and streets by hierarchically subdividing the input polygonal shape **S** and identifying a subset of parcel boundary sides as streets; see Section 4; and 2) optimize geometry of the parcels and streets via a global optimization on the parcel mesh; see Section 5. In particular, the first stage computes the topology as well as initial geometry of an urban layout, guided by the urban design requirements. The second stage optimizes the geometry of the urban layout to improve its geometric quality. User interactive control on the parcel and street modeling process mainly happens at the first stage.

4. Hierarchical Co-generation

We introduce a hierarchical approach to co-generate parcels and streets from a user-specified polygonal region **S**. At each hierarchical level, we first perform a binary subdivision for each parcel in the intermediate urban layout to generate new parcels, which are preferred to be regular to satisfy the first design requirement in Section 3; see Section 4.1. Next, we check if there is any new parcel that is not reachable via the existing street network. If there is no such parcel (see levels 1-3 in Figure 5), we do not need to generate any new street and thus we can directly move to the next level of parcel generation. Otherwise, we need to extend the existing street network with new streets, aiming to create a street access for each of the unreachable parcels; see parcels highlighted with a dot in levels 4-6 of Figure 5. Section 4.2 describes our street generation algorithm. Our hierarchical co-generation process terminates when any further partitioning of any parcel violates the minimally allowed parcel size constraint.

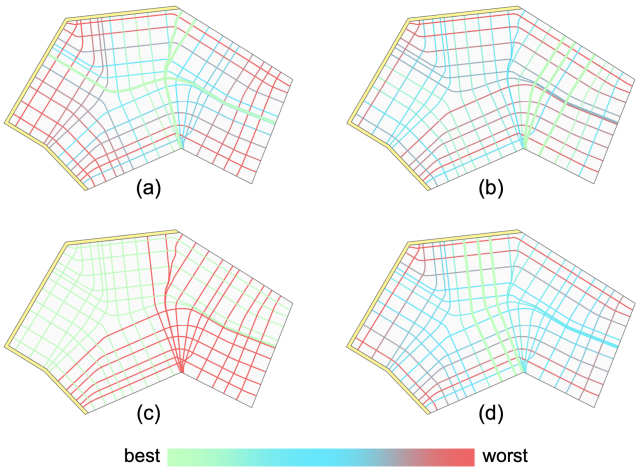

Figure 6: *Visualizing the quality scores of streamlines: (a) balanced parcel size Q_{size}, (b) parcel shape regularity Q_{regu}, (c) parcel street access Q_{acce}, and (d) overall quality score Q. The best three streamlines in (a) (b) and (d) are highlighted by a thicker line width. Note that there are only two kinds of streamlines in (c), corresponding to two possible values (2 or 1) of Q_{acce}.*

4.1. Parcel generation

At each hierarchical level, we subdivide each parcel into two smaller ones by choosing a partitioning line in the input parcel for partitioning it into two halves. The partitioning line can be either straight [PM01, KM07, WMWG09, VABW09b] or smoothly curved [YWVW13]. As demonstrated in [YWVW13], a smoothly

curved partitioning line called *streamline* has at least two advantages in terms of parcel binary partitioning: 1) streamlines (i.e., potential streets) meet at approximately right angles; and 2) streamlines are adaptable to irregular shapes of parcels to be partitioned. Due to this reason, we compute around 20 streamlines for each input parcel following the approach in [YWVW13] as candidates of the partitioning line, where the geometry of each streamline is represented as a polyline; see Figure 6. Note that the input parcel may need to be remeshed for computing the streamlines.

To perform the parcel partitioning, we propose a quality metric for evaluating each streamline computed from the input parcel and then choose the one with the highest metric score as the partitioning line (see again Figure 6):

$$Q = \lambda_1 Q_{\text{size}} + \lambda_2 Q_{\text{regu}} + \lambda_3 Q_{\text{acce}} \tag{2}$$

$$\text{with} \quad Q_{\text{size}} = \frac{S_i}{S_j},$$

$$Q_{\text{regu}} = \frac{1}{1+I_i+I_j},$$

$$\text{and} \quad Q_{\text{acce}} = \begin{cases} 2, & \text{if } s_i \geq \tau \ \& \ s_j \geq \tau \\ 1, & \text{otherwise} \end{cases}$$

where:

- Q_{size} measures how balanced the streamline-based partitioning on the input parcel is, where S_i and S_j are the size of the two resulting parcels P_i and P_j, respectively, assuming $S_i < S_j$.
- Q_{regu} measures regularity of the two resulting parcels P_i and P_j, where I_i and I_j are the irregularity metric of the two parcels computed using Equation 1, respectively.
- Q_{acce} is the number of new parcels that are reachable by the existing street, where s_i (s_j) is the ratio between the length of the new parcel P_i's (P_j's) street access and the average length of the new parcel P_i's (P_j's) boundary sides. τ is a threshold set as 0.5 in our experiments.
- λ_1, λ_2, and λ_3 are weights set as 0.3, 0.5, and 0.2, respectively, by default. Changing the weights from this default setting is one way to explore different partitioning variations for the same input parcel.

Thanks to the smoothly curved shape, the selected streamline for parcel partitioning is generally considered as a single (instead of multiple) boundary side for each of the two generated parcels.

Removing short edges in the parcel mesh. After generating new parcels, short edges may be introduced in the parcel mesh. This is because we partition each parcel independently without considering its neighbors. For example, in Figure 7 (left), each of two input parcels are partitioned into two smaller ones. A short edge (colored in red) is generated since the partitioning line (colored in cyan) of the input parcel at the top does not align well with the partitioning line (colored in blue) of the input parcel at the bottom. These short edges introduce unnecessary complexity to the parcel mesh as well as the parcel corner graph, and thus should be removed. We identify each of these short edges as an edge shared by two parcels whose length is less than $0.2 \cdot \min(\bar{l}_1, \bar{l}_2)$, where $\bar{l}_1$ and $\bar{l}_2$ are the average side length of the approximate polygon of the two parcels, respectively. To remove each short edge, we merge the two vertices of the edge into a single one, which is the middle point of the edge, and then create a short line segment to interpolate the two partitioning lines; see Figure 7 (right). As a result, the shapes of relevant parcels, as well as their approximate polygons, are changed slightly. In addition, two parcels sharing the short edge are not neighbors any more since they only share a vertex after the merge; see parcels A_2 and B_1 in Figure 7. The parcel graph is updated to reflect this change.

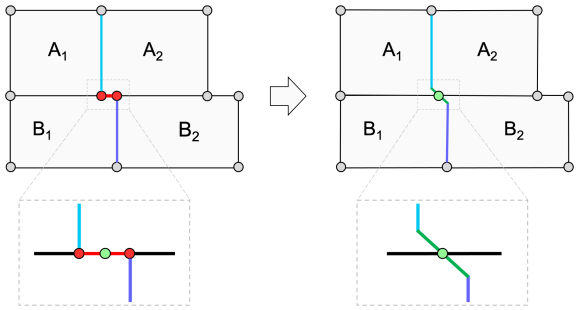

Figure 7: *Removing a short edge (colored in red) in a parcel mesh by merging its two vertices (colored in red) into a single one (colored in green) and introducing a short line segment (in dark green) to interpolate the two partitioning lines (in cyan and blue respectively).*

4.2. Street generation

As aforementioned, the goal of street generation is to create a street access for each parcel that is not reachable by the existing street network. The problem of street generation is to choose a subset of edges in the parcel corner graph as street segments (i.e., put the edges and associated vertices into the street network graph) while satisfying the requirements 2 to 6 in Section 3. Among the five requirements, short total street length and small number of street junctions are two requirements that are difficult to satisfy. When the hierarchical level is low and there are a few parcels, it is possible to enumerate all the possible solutions of streets to be generated and to choose one solution that satisfy the five requirements. However, this exhaustive search approach is not feasible when the hierarchical level becomes higher due to the combinatorial explosion of the search space.

To address the challenge, we propose an approach that leverages graph search algorithms to generate new streets efficiently; see levels 4-6 in Figure 5 for some street generation results. Our approach consists of four steps:

1. *Identify and group unreachable parcels.* We first identify all the unreachable parcels in the parcel corner graph. An unreachable parcel is a face in the parcel corner graph, none of whose edges is in the street network graph. Next, we group all the unreachable parcels, where parcels in the same group form a connected subgraph in the parcel corner graph. Thanks to our hierarchical modeling scheme, each group only has a few (usually less than 10) unreachable parcels. Figure 8(a) shows a simple case with a single group of unreachable parcels.

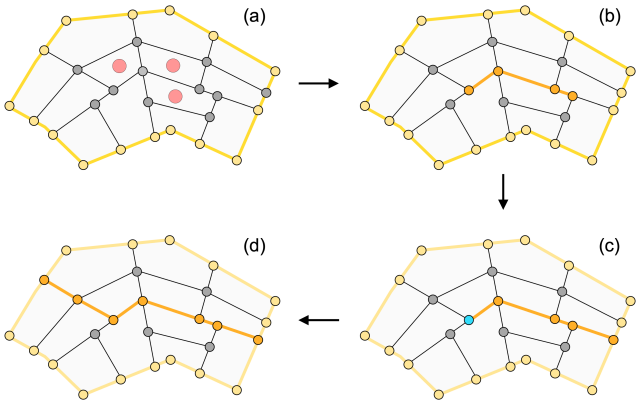

Figure 8: *Generating streets to ensure reachability of newly generated parcels: (a) identify and group unreachable parcels (highlighted with a red dot); (b) compute an L-shaped street access (in orange) for the group of unreachable parcels; (c) connect the L-shaped street access with the existing street network (in yellow) using a weighted shortest path (in orange) in the parcel corner graph, resulting in a street end (in cyan); and (d) connect the street end with the current street network to avoid generating a cul-de-sac.*

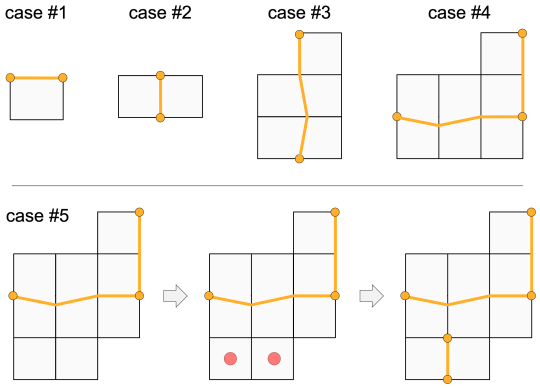

Figure 9: *Choose a subset of edges (colored in orange) as a street access for a group of unreachable parcels. We choose an I-shaped street for case #1, #2, and #3 and an L-shaped street for case #4. In case #5, we choose an L-shaped street to make most parcels reachable and then an I-shaped street to make the remaining two parcels (each with a red dot) reachable.*

2. *Generate street access for each group of unreachable parcels.* For each group of unreachable parcels $\{P_k\}$, we choose a subset of their edges denoted as E in the parcel corner graph as street segments such that $P_k \cap E \neq \emptyset$, $\forall k$ (i.e., ensuring street access). Moreover, the subset of edges E also has to satisfy two additional requirements: 1) short street length; and 2) a small number of street junctions. To solve the problem, we choose a subset of edges that forms an I-shaped (2 endpoints) or L-shaped (2 endpoints + 1 junction) street access; see Figure 9 (top). Note that two consecutive edges (chosen as street access) are considered as colinear if the included angle between the two edges is larger than 135°; otherwise, the vertex shared by the two edges is considered as a street junction. Street accesses with more complex shape (e.g., S-shaped street access) are possible to be included in our street generation algorithm; yet, such street accesses will result in a higher computational cost due to a larger number of combinations of parcel boundary sides.
   In detail, for each corner at the boundary of the group of unreachable parcels $\{P_k\}$, we generate an I-shaped street access for each incident edge of the corner, and then check if this street access makes all the parcels reachable. If such I-shaped street accesses exist, we choose the one with the shortest length as the solution. Otherwise, we enumerate all possible L-shaped street accesses using a similar strategy and choose the one with the shortest length as the solution. In case that we cannot find an I-shaped or L-shaped street access that makes all the parcels in the group reachable, we choose an I-shaped or L-shaped street access that makes a largest number of parcels reachable, and then apply the same approach recursively for the remaining parcels until all the parcels in the group are reachable; see Figure 9 (bottom). In our experiments, we find that a single I-shaped street access is the dominant solution to make each group of parcels reachable; see levels 4-6 in Figure 5 for examples.
3. *Ensure connectivity of the street network.* For each group of unreachable parcels, the generated street access may not be connected with the existing street network; see Figure 8(b). In this case, we need to connect the street access with the existing street network such that the resulting street network is connected; see Figure 8(c). To this end, we find a weighted shortest path to connect the street access (one subgraph) with the existing street network (the other subgraph) in the parcel corner graph using the Dijkstra's algorithm. The weight of each edge consists of two components: 1) edge length for minimizing the length of the shortest path; and 2) an integer 1 (0) if the included angle between the edge and the last edge is smaller (larger) than 135° to minimize the number of junctions on the shortest path. Users are allowed to adjust the weights that combine the two components for generating new streets with desired shape. In our experiments, we give the second component a larger weight.
4. *Avoid cul-de-sacs in the street network (optional).* The above steps may generate cul-de-sacs in the street network; see Figure 8(c). In case that users do not want these cul-de-sacs, our street generation algorithm connects the street end of each cul-de-sac with the current street network using the same approach as in step 3; see Figure 8(d) for an example.

Our hierarchical co-generation process outputs an initial urban layout, guided by the six urban design requirements in Section 3; see again Figure 5. In the initial urban layout, parcels have regular approximate polygons and each parcel is reachable by the street network, which is geometrically connected and simple; see Figure 10 (top). However, some parcel boundary sides in the initial layout do not have smooth geometry, mainly caused by merging vertices to remove short edges in the parcel mesh; see again Figure 7 (right). Moreover, some streets in the street network also do not have smooth shape since it is not considered when generating these streets. To address these issues, we need a global optimization on the initial urban layout's geometry.

5. Geometric Optimization

We propose a global optimization on the parcel mesh **M** to improve the geometric quality of the generated parcels and streets. The search space of the optimization is positions of all the mesh vertices. We makes use of the parcel graph to identify vertices that form each parcel's boundary shape as well as its approximate polygon. We make use of the street graph to identify vertices that form each street as well as vertices that are street junctions.

Objective function. The energy function we want to minimize is defined as:

$$E = \omega_1 E_{\text{regu}} + \omega_2 E_{\text{side}} + \omega_3 E_{\text{stre}} + \omega_4 E_{\text{junc}} + \omega_5 E_{\text{close}} \quad (3)$$

We describe each individual energy term below. ω_1, ω_2, ω_3, ω_4, and ω_5 are weights, and we set their values in a way that the energy terms E_{side} and E_{stre} are more significant than the others in the objective function.

1. *Parcel regularity*. The parcel regularity term is the sum of the irregularity metric computed for each parcel's approximate polygon:

$$E_{\text{regu}} = \sum_{i=1}^{n} I_i. \quad (4)$$

where n is the number of parcels in the urban layout.

2. *Parcel side smoothness*. Removing short edges in Section 4.1 may generate non-smooth parcel boundary sides; see again Figure 7 (right). To address this issue, we smooth each pair of colinear boundary sides, denoted as (B_i, B_j), in the parcel mesh. The smoothness of a point, $\mathbf{v}_k$, on the parcel sides (B_i, B_j) is measured by the squared second-order difference of three consecutive vertices [YWVW13]. The parcel side smoothness term is:

$$E_{\text{side}} = \sum_{(B_i,B_j)} \sum_{\mathbf{v}_{k-1},\mathbf{v}_k,\mathbf{v}_{k+1} \in (B_i,B_j)} (\mathbf{v}_{k-1} - 2\mathbf{v}_k + \mathbf{v}_{k+1})^2. \quad (5)$$

3. *Street smoothness*. For each street, the smoothness of a street point, $\mathbf{v}_k$, is measured using a method same as the 2nd term. The street smoothness term is:

$$E_{\text{stre}} = \sum_{S_j} \sum_{\mathbf{v}_{k-1},\mathbf{v}_k,\mathbf{v}_{k+1} \in S_j} (\mathbf{v}_{k-1} - 2\mathbf{v}_k + \mathbf{v}_{k+1})^2. \quad (6)$$

where S_j is a street in the street network.

4. *Street junction term*. The included angle at each street junction, denoted as $\mathbf{v}_l$, should be close to 90°. The street junction term is defined as:

$$E_{\text{junc}} = \sum_{\mathbf{v}_l} ((\mathbf{v}_k - \mathbf{v}_l) \cdot (\mathbf{v}_m - \mathbf{v}_l))^2. \quad (7)$$

where $\mathbf{v}_l$ is a junction of two streets, $\mathbf{v}_k$ is a vertex on one street, and $\mathbf{v}_m$ is a vertex on the other street.

5. *Layout closeness*. The layout closeness is defined as the sum of the squared distances to the vertices of the initial layout, i.e.,

$$E_{\text{close}} = \sum_i (\mathbf{v}_i - \mathbf{v}_i^0)^2. \quad (8)$$

where $\mathbf{v}_i^0$ is the vertex corresponding to vertex $\mathbf{v}_i$ in the initial layout.

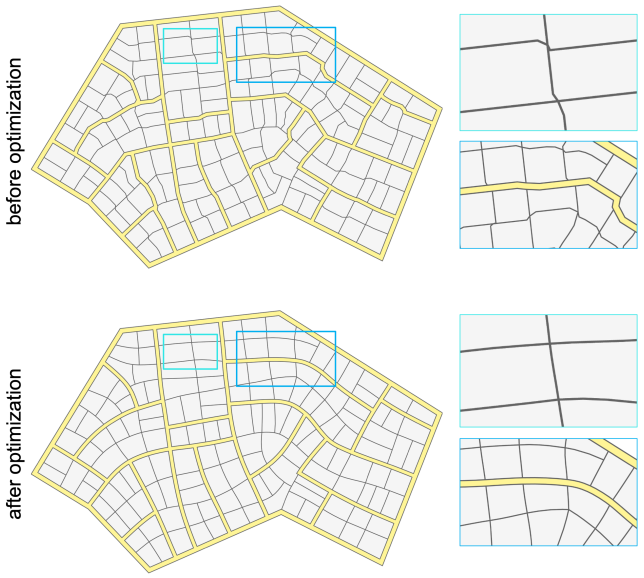

Figure 10: *An urban layout (top) before and (bottom) after our optimization, showing that our optimization results in smooth parcel sides and streets. Zoom-in views of some parcel sides and streets are shown on the right.*

Constraints. The only constraint is to preserve the input land's boundary shape. To this end, we first identify vertices on the boundary polygon that connect two colinear edges. Denote the vertex as $\mathbf{v}_k$, and its two neighboring vertices as $\mathbf{v}_{k-1}$ and $\mathbf{v}_{k+1}$. The constraint is:

$$\mathbf{v}_k = \mathbf{v}_{k-1} + t \frac{(\mathbf{v}_{k+1} - \mathbf{v}_{k-1})}{\|\mathbf{v}_{k+1} - \mathbf{v}_{k-1}\|}, t \in (0,1). \quad (9)$$

For the other vertices, the constraint is:

$$\mathbf{v}_k = \mathbf{v}_k^0. \quad (10)$$

Solver. We solve the optimization problem using the ShapeOp library [BDS*12] implemented in Kangaroo library for Rhino. In the solver, all the energy terms and the constraints explained above are represented using the mesh vertex positions. Self-intersection and degeneration rarely happen in our optimized mesh since parcels in the initial urban layout are already quite regular and our optimization only makes slight change to the urban layout. Figure 10 shows an urban layout before and after the geometric optimization, from which we can see that both parcel sides and streets become much smoother after the optimization while the parcels remains regular.

6. Result

We implemented our computational framework in Rhino and Grasshopper with Python. We conducted all experiments on a laptop computer with an Intel i9 Processor and 32GB of RAM.

Global control on the urban layout. Our computational framework allows users to globally control the generated urban layout in several aspects:

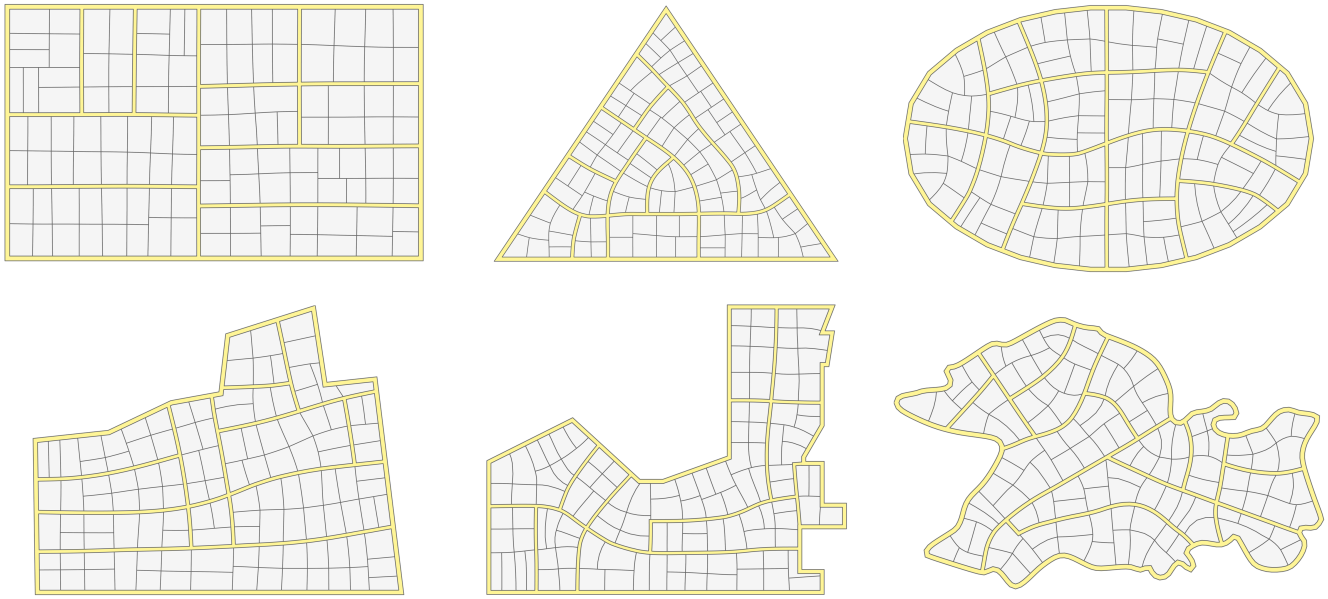

Figure 11: *Our framework allows modeling parcels and streets for input lands with (top) regular and (bottom) irregular shapes. (First row from left to right): Rect, Triangle ,Ellipse; (Second row from left to right): Wyoming, Sant Adria, Wandsworth.*

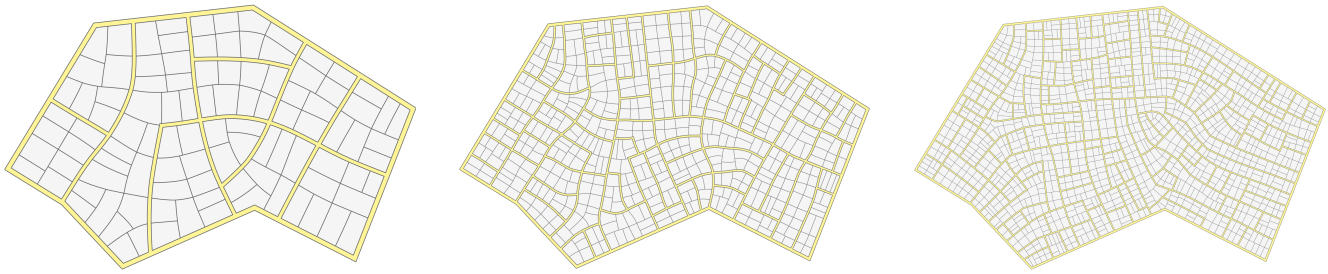

Figure 12: *Our framework allows modeling urban layouts for the same input land with different numbers of parcels, i.e., 102, 539, and 1273 parcels.*

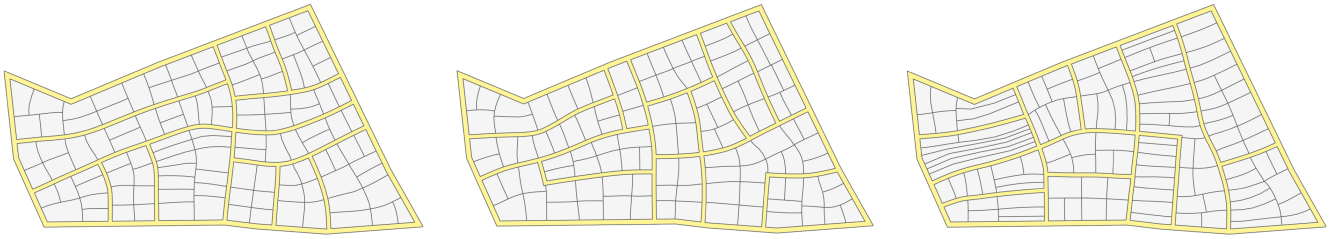

Figure 13: *Our framework allows controlling parcel shapes by adjusting the three weights in the streamline quality metric (see Equation 2); e.g., the middle layout has more regular parcels, and the right layout has some elongated parcels for sharing the same street access.*

- *Input land shape.* Users are allowed to specify the input land shape. Our framework supports modeling parcels and streets for lands with a variety of shapes, including both regular shapes (see Figure 11 (top)) and irregular shapes (see Figure 11 (bottom)). For irregular land shapes, most of our generated parcels are regular, except for some of those close to the land boundary.
- *Parcel number.* Users are allowed to control the number of parcels to be generated from an input land shape by specifying the minimally allowed parcel size; see Figure 12 for examples.
- *Parcel shape.* Users are allowed to specify preference on the parcel shapes by adjusting the three weights in Equation 2 that defines the quality score of a streamline for parcel partitioning; see

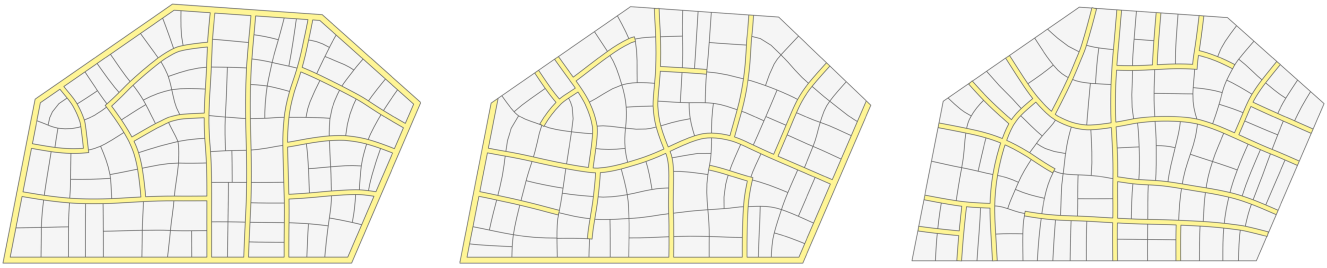

Figure 14: *Our framework allows modeling urban layouts for the same input land with different street patterns, from left to right, loop, cul-de-sac, and tree-like patterns. Note that only part of the land boundary is street in the middle layout and none of the land boundary is street in the right layout.*

Figure 15: *Users are allowed to locally control the generation process of parcels and streets at each hierarchical level, including (b) specifying two new streets (in orange), (d) choosing another streamline for generating new parcels, (g) undoing partitioning on a parcel (for using it as a park), and (j) editing the generated street network by adding a new street. The resulting urban layout is shown in (l), where the green parcel is the park.*

Figure 13. In general, a larger weight λ_2 on the regularity component Q_{regu} results in more regular parcels and a larger weight λ_3 on the street access component Q_{acce} leads to some elongated parcels which try to share the same existing street access.

- *Parcel orientation.* Users are allowed to specify preference on the orientation of parcels, e.g., elongated side of a parcel faces the east-west direction. To this end, we add one more component to the streamline quality metric in Equation 2, which is the angle between the streamline direction and the east-west direction. Figure 16 shows urban layout results with two different preferences on the parcel orientation.
- *Street pattern.* Users are also allowed to control the pattern of generated streets. By avoiding a street end for every generated street, the modeled street network has a loop-like pattern; see Figure 14 (left). By allowing street ends only for the last few hierarchical levels, the generated street network has a cul-de-sac pattern; see Figure 14 (middle). By disabling step 4 (i.e., avoid street ends for the street network) in Section 4.2, the generated street network has a tree-like pattern; see Figure 14 (right).

Local control on the urban layout. Our computational framework also allows users to locally control the parcel and street generation process, enabling users to model specific parcels and streets according to their needs and desires. Thanks to our hierarchical approach, users are able to locally edit parcels and streets generated at each hierarchical level, as long as they still satisfy the urban design requirements in Section 3. After that, our framework continues running from the edited intermediate urban layout to generate new parcels and streets in the next hierarchical level.

Table 1: *Statistics of results shown in the paper, where the column ID is shown on top of the table. The 2nd to 5th columns show the user input, including the land name, perimeter, area, and minimally allowed parcel area. The 6th to 11th columns show the number of parcels with different types of approximate polygons as well as the total number of parcels. The 12th to 14th columns show the minimum, maximum, and average irregularity metric of all the parcels. The 15th to 17th columns show the number of streets, street junctions, and street ends in the street network graph, excluding the user-specified street at the input land boundary. The 18th to 20th columns show the total street length, average street length, and average deviation from 90° for street junction angles. The 21th to 23nd columns show the maximum hierarchical level to generate an urban layout, as well as the computation time to generate and optimize an urban layout, respectively.*

c1	c2 - c5					c6 - c11						c12 - c14			c15 - c17			c18 - c20			c21 - c23		
Fig	user input					parcel (approx. polygon)						parcel (irregularity)			street network graph			street (geometry)			computation		
	land name		land perim. (km)	land area (ha)	min parcel area (ha)	#tri	#quad	#pen	#hex	#hep	#total	I_{min}	I_{max}	I_{avg}	#street	#junction	#end	total length (km)	avg length (km)	junction angle dev. (°)	max hier. level	generation (min)	optimization (min)
1	Tengah		5.685	147.913	1	2	104	2	0	0	108	0.0012	0.1775	0.0327	9	24	0	7.953	0.884	8.64	9	1.10	0.03
11	Rect		6.085	217.603	1	0	116	0	0	0	116	0.0000	0.0754	0.0097	9	19	0	7.574	0.842	0.42	7	0.67	0.03
	Triangle		5.042	121.194	1	1	102	0	0	0	103	0.0001	0.1378	0.0274	8	18	0	5.421	0.678	2.94	8	1.20	0.03
	Ellipse		5.146	195.3784	1	4	124	0	0	0	128	0.0004	0.1591	0.0370	9	25	0	9.291	1.032	7.04	8	1.30	0.05
	Wyoming		5.551	155.5931	1	0	115	0	2	0	117	0.0001	0.1158	0.0175	9	20	0	7.186	0.798	5.10	8	0.91	0.03
	Sant Adria		6.216	167.6226	1	0	103	4	7	0	114	0.0002	0.5539	0.0401	10	20	0	6.216	0.622	8.95	9	1.10	0.04
	Wandsworth		7.195	161.2786	1	4	95	12	5	1	117	0.0027	0.3512	0.0648	12	24	0	7.093	0.591	10.32	8	1.80	0.06
12	Heptagon	lef	5.175	157.0138	1	1	98	2	1	0	102	0.0005	0.1390	0.0306	8	18	0	6.111	0.764	4.64	8	0.78	0.03
		mid	5.175	157.0138	0.2	0	536	3	0	0	539	0.0001	0.2267	0.0237	61	127	0	21.360	0.350	1.79	10	44.30	0.58
		rig	5.175	157.0138	0.1	1	1269	3	0	0	1273	0.0000	0.1667	0.0197	129	320	0	45.662	0.354	3.90	13	408.00	2.20
13	Pathum Wan	lef	5.702	160.1366	1	0	133	1	0	0	134	0.0003	0.1765	0.0309	10	21	0	7.094	0.709	1.32	7	0.85	0.03
		mid	5.702	160.1366	1	0	107	1	0	0	108	0.0007	0.2027	0.0246	13	25	0	7.543	0.580	2.53	8	0.95	0.03
		rig	5.702	160.1366	1	0	111	1	0	0	112	0.0006	0.2489	0.0680	10	21	0	6.920	0.692	1.98	7	1.20	0.03
14	Hexagon	lef	4.486	130.7029	1	1	93	3	0	0	97	0.0007	0.1965	0.0340	12	22	0	6.884	0.574	4.29	7	0.73	0.08
		mid	4.486	130.7029	1	1	88	3	0	0	92	0.0013	0.1406	0.0325	13	15	13	5.940	0.457	1.40	7	0.75	0.06
		rig	4.486	130.7029	1	0	97	2	0	0	99	0.0005	0.2366	0.0412	20	17	21	8.673	0.434	3.36	7	0.77	0.03
16	Bang Khae	lef	4.596	123.4238	1	0	81	5	0	0	86	0.0012	0.5508	0.0734	11	19	0	5.183	0.471	9.62	8	0.73	0.03
		rig	4.596	123.4238	1	1	72	6	0	0	79	0.0011	0.2608	0.0630	9	18	0	5.800	0.644	5.39	7	0.65	0.03

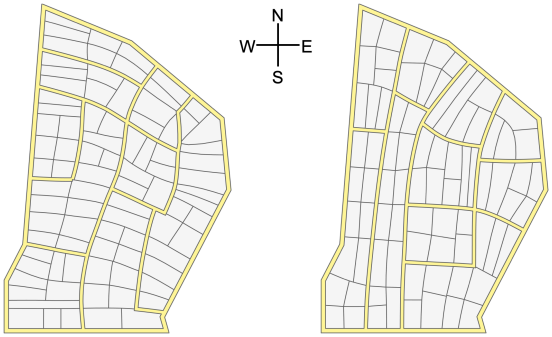

Figure 16: *Our framework allows modeling parcels with two different orientations: (left) east-west orientation and (right) north-south orientation.*

Table 2: *Statistics of results in the three comparison experiments.*

		parcel (approx. polygon)						parcel (irregularity)		street network graph			street (geometry)
		#tri	#quad	#pen	#hex	#hep	#total	I_{max}	I_{avg}	#street	#junction	#end	junction angle dev. (°)
Fig 17	CityEngine	7	88	2	0	0	97	0.2250	0.0382	/	/	/	/
	Our	2	104	2	0	0	108	0.1775	0.0327	/	/	/	/
Fig 18	Decoding Spaces	8	84	1	1	0	94	0.3652	0.0541	14	27	0	24.48
	Our	1	92	2	0	0	95	0.2133	0.0385	10	20	0	3.76
Fig 19	[YWVW 13]	23	966	5	1	0	995	0.3071	0.0273	35	68	12	2.07
	Our	1	961	11	0	0	973	0.3377	0.0199	110	223	0	2.34

Figure 15 shows a running example that demonstrates this feature. At the beginning, the user specifies two new streets besides the input land shape; see Figure 15(b). The user does not like the partitioning on the top left parcel in Figure 15(c) and thus chooses another streamline as the partitioning line for generating another two new parcels; see Figure 15(d). The user thinks that the partitioning on one parcel in Figure 15(f) is not necessary since he/she wants to use that parcel as a park. Hence, he/she undoes the partitioning on the parcel and marks the parcel such that it will not be partitioned any more; see Figure 15(g). The user thinks that the street network generated in Figure 15(i) is not satisfactory since he/she wants more street in the bottom left part of the layout. Hence, he/she adds a new street in that area by choosing an additional subset of graph edges to form the street; see Figure 15(j). The user can repeat these local editing operations in any hierarchical level until generating desired parcels and streets. In practice, we found that the local editing is more effective at low hierarchical levels since there are a small number of parcels and streets. Although not implemented, interactive editing of a co-generated urban layout is possible to be supported by our computational framework such as combining parcels to create parks and changing the street location at some places.

Statistics. Table 1 shows statistics of all the results shown in the paper. Some input land shapes are chosen as a simplified administrative boundary of a city or a district. For each result, the generated parcels are dominated by quad-shaped parcels, which are preferred in urban modeling; see 7th column (#quad) in Table 1. Very few parcels with other shapes like triangles and hexagons are generated, most of which are close to the boundary of the input land; see the bottom right layout in Figure 11 for example. The 14th column (I_{avg}) in Table 1 shows that our generated parcels have quite regular shapes since the average irregularity metric is generally low. Comparing the 11th (# total) and 15th (# street) columns, we find that a street can make around 10 parcels reachable in average. The 20th column (junction angle dev.) shows that the average deviation of street junction angles from 90° ranges from 0.43° to 13.08°, indicating that most streets meet at approximately right angles. The 21th column (max hier. level) shows the maximum hierarchical

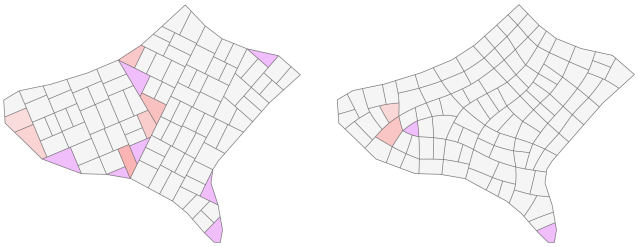

Figure 17: *Comparing (left) CityEngine [Esr23] with (right) our approach. CityEngine generates an urban layout with a larger number of triangles (in purple) and a larger number of irregular non-triangle polygons (in red) whose irregularity is larger than 0.1. Saturation of the highlighted non-triangle polygons indicates their irregularity value.*

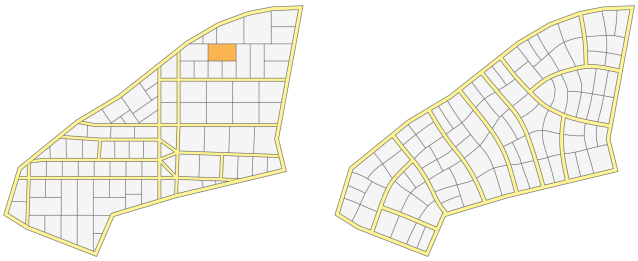

Figure 18: *Comparing (left) DeCodingSpaces [ABB*17] with (right) our approach. DeCodingSpaces generates streets that are not evenly distributed and parcels that have a large variation in size. In addition, the urban layout consists of one parcel (colored in orange) that is not reachable with the street network.*

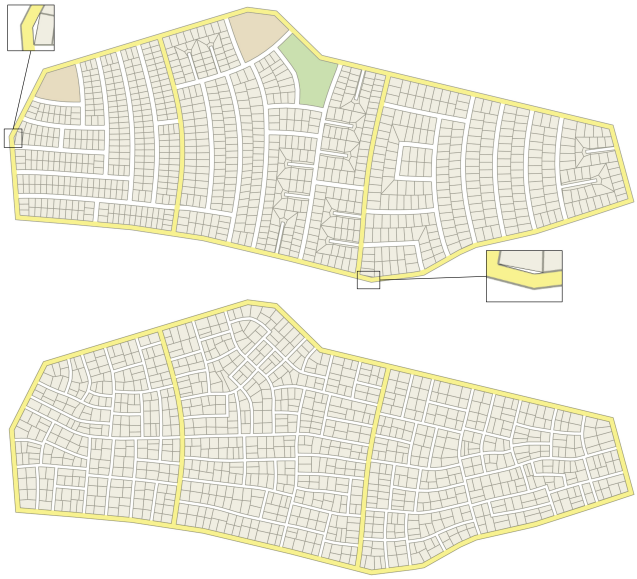

Figure 19: *Comparing (top) a template-based approach [YWVW13] with (bottom) our approach, for the same input land with two prescribed streets (in yellow color). The urban layout generated by [YWVW13] consists of three land regions unpartitioned (two in orange and one in green) and two land regions with a triangular gap (see the two zoom-in views). In contrast, all the land regions can be properly partitioned by our approach without any gap. Note that we render our streets in white (except the prescribed ones) to follow the rendering style of [YWVW13].*

level to generate each result (e.g., level-8 results have around 100 parcels). We found that our binary partitioning tree becomes unbalanced when the hierarchical level gets higher in our experiments. The computation time for each result is dominated by the hierarchical co-generation process and the optimization stage only takes a few seconds; see the rightmost two columns in Table 1.

Comparison with existing approaches. We compare our approach with three existing approaches [Esr23, ABB*17, YWVW13] in terms of generating parcels and streets from a given land shape. Table 2 compares the statistics of the generated results, showing that our approach allows generating parcels and streets with higher geometric quality in general.

Comparison with CityEngine [Esr23]. CityEngine is a rule-based urban modeling software package [Kel21]. It makes use of the oriented bounding box-based subdivision approach [VKW*12] to generate parcels, where partitioning line is always straight. Figure 17 shows parcels generated by CityEngine and our approach for the same input land. The result by CityEngine consists of a larger number of triangles (7 vs 2) and a larger number of irregular non-triangle polygons (6 vs 2). Moreover, the maximum irregularity and average irregularity of CityEngine's parcels are also larger than ours; see again Table 2. This experiment shows that our approach is able to generate parcels that are more regular than CityEngine, for an input land with irregular shape.

*Comparison with DeCodingSpaces [ABB*17].* DeCodingSpaces Toolbox for Grasshopper is a collection of analytical and generative components for algorithmic urban planning. To generate an urban layout, this toolbox first utilizes an L-system-based approach to generate streets and then subdivide the induced partition of land into parcels using an approach similar to CityEngine. Figure 18 shows streets and parcels generated by DeCodingSpaces and our approach for the same input land. Comparing with our generated streets, streets generated by the L-system of DeCodingSpaces are not evenly distributed in the input land; the generated streets have a larger number of junctions (27 vs 20); and some street junction angles deviate a lot from 90°; see also Table 2. Although DeCodingSpaces considers reachability of parcels, it cannot ensure that every parcel is reachable; e.g., there is an unreachable parcel in Figure 18 (left). DeCodingSpaces stops parcel subdivision when it predicts that further subdivision will generate some parcels that are not reachable with the street network. Due to this reason, the resulting parcels have a large variation in size, which is not preferred in urban modeling. Compared with DeCodingSpaces, our co-generation approach allows to generate evenly distributed streets and regular parcels with similar sizes while ensuring reachability of each parcel with the street network.

Comparison with [YWVW13]. We compare our approach with a template-based approach [YWVW13] for generating parcels and

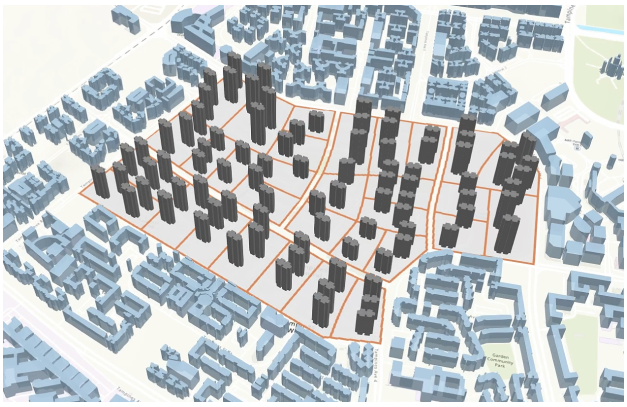

Figure 20: *Integrating our co-generation approach into an urban planning tool to generate an urban layout on a city map consisting of not only streets and parcels but also 3D buildings.*

streets. Due to the use of predefined templates, the urban layout generated by [YWVW13] is well structured, where most parcels have regular quad shape and streets are well utilized to make each parcel reachable; see Figure 19 (top). However, the predefined templates also restrict the parcel generation process, especially for the land regions with irregular shape. For example, there are three land regions (two in orange and one in green color) that cannot be further partitioned since none of the templates can fit these regions. There are also two land regions that can be fit by the templates but a small triangular gap is left in each land region; see the two zoom-in views in Figure 19 (top). Our framework is also able to generate an urban layout where parcels have regular shapes and are reachable by a street network; see Figure 19 (bottom). Since our approach does not rely on any template, our generated urban layout looks less uniform than the one by [YWVW13] and requires more streets to make the parcels reachable. Yet, a benefit of not using any template is that our approach is able to generate parcels from lands with very complex shapes, potentially increasing land utilization.

User scenario. We show that our hierarchical co-generation approach can be integrated into an urban planning tool to generate urban layouts with not only streets and parcels but also buildings; see Figure 20. In the urban planning tool, users first specify the land for urban development or redevelopment on the map of a city. Then, streets and parcels are automatically generated by using our hierarchical co-generation approach, during which users can control the generation process in various ways (e.g., parcel orientation and street pattern). Thanks to the regularity of our generated parcels, 3D buildings can be easily modeled in each parcel; see the accompanying video for a demo.

7. Conclusion

This paper presents a computational framework to model parcels and streets, guided by a set of fundamental design requirements in urban modeling. Our framework consists of a hierarchical approach to co-generate parcels and streets and a global optimization to improve geometric quality of the parcels and streets. Our framework is able to model urban layouts with high geometric quality, where parcels have regular shapes and are all reachable by a connected and simple street network. Thanks to the hierarchical co-generation approach, users are able to control the generated parcels and streets in many different ways, both locally and globally.

Limitations and future work. Our work has several limitations that open up interesting directions for future research. First, we model parcels without considering land use (e.g., residential, commercial, or industrial). Studying land-use simulation and applying it to guide the parcel modeling would be an important future work. Second, we abstract the street generation as a graph search problem, without considering many practical factors such as hierarchy of streets, traffic flow, and walking/driving distance (e.g., average walking distances of residential parcels to parks). Third, we formulate a set of design requirements and metrics for urban modeling in this paper. In the future, we may study whether the proposed metrics are applicable to different kinds of urban layouts. Lastly, our modeling approach assumes the input land is planar, ignoring its height variation. Extending our approach to model parcels and streets in an uneven land would be an interesting future work.

8. Acknowledgement

We thank the reviewers for their valuable comments and Anna Claudia Yenardi for helping on previewing our result within an interactive urban planning tool. This work was supported by the Singapore URA grant (RS-INDUS-00095: Computational modelling for optimisation of planning and urban design parameters).

References

[ABB*17] ABDULMAWLA A., BIELIK M., BUŠ P., DENNEMARK M., FUCHKINA E., MIAO Y., KNECHT K., KÖNIG R., SCHNEIDER S.: DecodingSpaces Toolbox for Grasshopper, 2017. https://toolbox.decodingspaces.net. 2, 12

[ABVA08] ALIAGA D. G., BENEŠ B., VANEGAS C. A., ANDRYSCO N.: Interactive reconfiguration of urban layouts. *IEEE Comp. Graph. and App. 28*, 3 (2008), 38–47. 2

[AVB08] ALIAGA D. G., VANEGAS C. A., BENEŠ B.: Interactive example-based urban layout synthesis. *ACM Trans. on Graph. (SIGGRAPH Asia) 27*, 5 (2008), 160:1–160:10. 3

[AYWM14] ALHALAWANI S., YANG Y.-L., WONKA P., MITRA N. J.: What makes london work like london? *Comp. Graph. Forum (SGP) 33*, 5 (2014), 157–165. 2

[BDS*12] BOUAZIZ S., DEUSS M., SCHWARTZBURG Y., WEISE T., PAULY M.: Shape-Up: Shaping discrete geometry with projections. *Comp. Graph. Forum (SGP) 31*, 5 (2012), 1657–1667. 8

[BWK14] BENEŠ J., WILKIE A., KŘIVÁNEK J.: Procedural modelling of urban road networks. *Comp. Graph. Forum 33*, 6 (2014), 132–142. 2

[BYK19] BUSQUETS J., YANG D., KELLER M.: *Urban Grids: Handbook for Regular City Design*. ORO Editions, 2019. 3

[CEW*08] CHEN G., ESCH G., WONKA P., MÜLLER P., ZHANG E.: Interactive procedural street modeling. *ACM Trans. on Graph. (SIGGRAPH) 27*, 3 (2008), 103:1–103:10. 2

[CLA*19] CHU H., LI D., ACUNA D., KAR A., SHUGRINA M., WEI X., LIU M.-Y., TORRALBA A., FIDLER S.: Neural turtle graphics for modeling city road layouts. In *Proc. Int. Conf. on Comp. Vis.* (2019), pp. 4522–4530. 3

[DC14] DAHAL K. R., CHOW T. E.: A gis toolset for automated partitioning of urban lands. *Environmental Modelling & Software 55* (2014), 222–234. 3

[EBP*12] EMILIEN A., BERNHARDT A., PEYTAVIE A., CANI M.-P., GALIN E.: Procedural generation of villages on arbitrary terrains. *Visual Computer 28* (2012), 809–818. 2

[Esr23] ESRI: ArcGIS CityEngine - Advanced 3D City Design Software, 2023. https://www.esri.com/en-us/arcgis/products/arcgis-cityengine/overview. 2, 12

[EVC*15] EMILIEN A., VIMONT U., CANI M.-P., POULIN P., BENEŠ B.: Worldbrush: Interactive example-based synthesis of procedural virtual worlds. *ACM Trans. on Graph. (SIGGRAPH) 34*, 4 (2015), 106:1–106:11. 3

[FJY22] FANG Z., JIN Y., YANG T.: Incorporating planning intelligence into deep learning: A planning support tool for street network design. *Journal of Urban Technology 29*, 2 (2022), 99–114. 3

[GPGB11] GALIN E., PEYTAVIE A., GUÉRIN E., BENEŠ B.: Authoring hierarchical road networks. *Comp. Graph. Forum (Pacific Graphics) 30*, 7 (2011), 2021–2030. 2

[GPMG10] GALIN E., PEYTAVIE A., MARÉCHAL N., GUÉRIN E.: Procedural generation of roads. *Comp. Graph. Forum 29*, 2 (2010), 429–438. 2

[HWWK17] HARTMANN S., WEINMANN M., WESSEL R., KLEIN R.: StreetGAN: Towards road network synthesis with generative adversarial networks. In *25th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision* (2017), pp. 133–142. 3

[Kel21] KELLY T.: CityEngine: An introduction to rule-based modeling. In *Urban informatics*, Shi W., Goodchild M. F., Batty M., Kwan M.-P., Zhang A., (Eds.). Springer, 2021, ch. 35, pp. 637–662. 12

[KKC18] KIM J.-S., KAVAK H., CROOKS A.: Procedural city generation beyond game development. *SIGSPATIAL Special 10*, 2 (2018), 34–41. 2

[KM07] KELLY G., MCCABE H.: Citygen: An interactive system for procedural city generation. In *Fifth International Conference on Game Design and Technology* (2007), pp. 8–16. 2, 3, 5

[KM19] KEMPINSKA K., MURCIO R.: Modelling urban networks using variational autoencoders. *Applied Network Science 4* (2019), 114:1–114:11. 3

[KMK*17] KOENIG R., MIAO Y., KNECHT K., BUŠ P., MEI-CHIH C.: Interactive urban synthesis: Computational methods for fast prototyping of urban design proposals. In *International Conference on Computer-Aided Architectural Design Futures* (2017), pp. 23–41. 2

[LSWW11] LIPP M., SCHERZER D., WONKA P., WIMMER M.: Interactive modeling of city layouts using layers of procedural content. *Comp. Graph. Forum (Eurographics) 30*, 2 (2011), 345–354. 2

[MKMA19] MATHEW C. D. T., KNOB P. R., MUSSE S. R., ALIAGA D. G.: Urban walkability design using virtual population simulation. *Comp. Graph. Forum 38*, 1 (2019), 455–469. 2

[NGDA16] NISHIDA G., GARCIA-DORADO I., ALIAGA D. G.: Example-driven procedural urban roads. *Comp. Graph. Forum 35*, 6 (2016), 5–17. 3

[NPA*22] NIESE T., PIRK S., ALBRECHT M., BENES B., DEUSSEN O.: Procedural urban forestry. *ACM Trans. on Graph. 41*, 2 (2022), 20:1–20:18. 2

[OM20] OWAKI T., MACHIDA T.: RoadNetGAN: Generating road networks in planar graph representation. In *International Conference on Neural Information Processing* (2020), pp. 535–543. 3

[PM01] PARISH Y. I. H., MÜLLER P.: Procedural modeling of cities. In *Proc. of SIGGRAPH* (2001), pp. 301–308. 2, 3, 5

[Pop15] POPE A.: *Ladders*. Princeton Architectural Press, 2015. 3

[PYB*16] PENG C.-H., YANG Y.-L., BAO F., FINK D., YAN D.-M., WONKA P., MITRA N. J.: Computational network design from functional specifications. *ACM Trans. on Graph. (SIGGRAPH) 35*, 4 (2016), 131:1–131:12. 3

[STBB14] SMELIK R. M., TUTENEL T., BIDARRA R., BENEŠ B.: A survey on procedural modelling for virtual worlds. *Comp. Graph. Forum 33*, 6 (2014), 31–50. 2

[SYBG02] SUN J., YU X., BACIU G., GREEN M.: Template-based generation of road networks for virtual city modeling. In *Proc. ACM Symposium on Virtual Reality Software and Technology* (2002), pp. 33–40. 2

[TB17] TENG E., BIDARRA R.: A semantic approach to patch-based procedural generation of urban road networks. In *12th International Conference on the Foundations of Digital Games* (2017), pp. 71:1–71:10. 2

[VABW09a] VANEGAS C. A., ALIAGA D. G., BENEŠ B., WADDELL P. A.: Interactive design of urban spaces using geometrical and behavioral modeling. *ACM Trans. on Graph. (SIGGRAPH Asia) 28*, 5 (2009), 111:1–111:10. 2

[VABW09b] VANEGAS C. A., ALIAGA D. G., BENEŠ B., WADDELL P.: Visualization of simulated urban spaces: Inferring parameterized generation of streets, parcels, and aerial imagery. *IEEE Trans. Vis. & Comp. Graphics 15*, 3 (2009), 424–435. 2, 3, 5

[VAW*10] VANEGAS C. A., ALIAGA D. G., WONKA P., MÜLLER P., WADDELL P., WATSON B.: Modeling the appearance and behavior of urban spaces. *Comp. Graph. Forum (Eurographics) 29*, 1 (2010), 25–42. 2

[VGDA*12] VANEGAS C. A., GARCIA-DORADO I., ALIAGA D. G., BENEŠ B., WADDELL P.: Inverse design of urban procedural models. *ACM Trans. on Graph. (SIGGRAPH Asia) 31*, 6 (2012), 168:1–168:11. 2

[VKW*12] VANEGAS C. A., KELLY T., WEBER B., HALATSCH J., ALIAGA D. G., MÜLLER P.: Procedural generation of parcels in urban modeling. *Comp. Graph. Forum (Eurographics) 31*, 2 (2012), 681–690. 2, 3, 12

[WMWG09] WEBER B., MÜLLER P., WONKA P., GROSS M.: Interactive geometric simulation of 4d cities. *Comp. Graph. Forum (Eurographics) 28*, 2 (2009), 481–492. 2, 3, 5

[YS12] YU Q., STEED A.: Example-based road network synthesis. In *Eurographics - Short Papers* (2012), pp. 53–56. 3

[YWVW13] YANG Y.-L., WANG J., VOUGA E., WONKA P.: Urban pattern: Layout design by hierarchical domain splitting. *ACM Trans. on Graph. (SIGGRAPH Asia) 32*, 6 (2013), 181:1–181:12. 2, 3, 5, 6, 8, 12, 13

[ZCCB21] ZHOU X., CHANG P., CANI M.-P., BENEŠ B.: Urban brush: Intuitive and controllable urban layout editing. In *Proc. ACM UIST* (2021), pp. 796–814. 2