Volume 34 (2015), Number 2 pp. 1–13 COMPUTER GRAPHICS *forum*

Example-Driven Procedural Urban Roads

G. Nishida, I. Garcia-Dorado, and D. Aliaga

Purdue University, USA

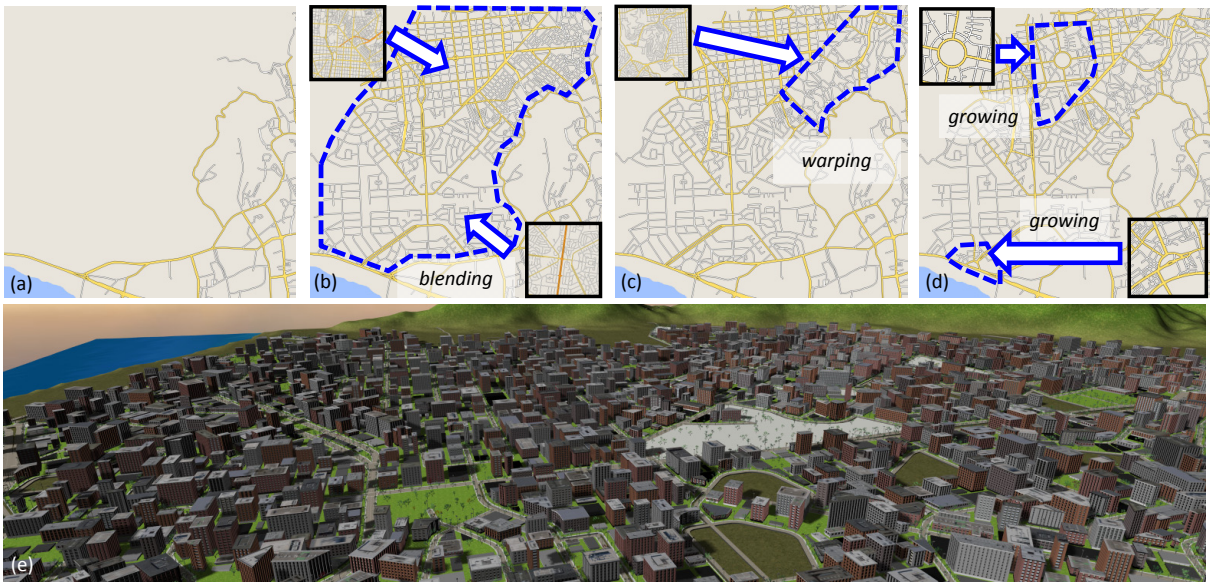

Figure 1: *Example-driven result: our interactive approach enables a user to quickly design a road network for an entire city. In this example, a) the user starts with a virtual city using roads from Jiangmen, China (one of the world's 10 fastest-growing cities). b) The user selects a target space for a new urban area, and a new road network is generated by growing and blending two road styles. c) Roads in the top right corner are replaced by a selected example of curved roads. d) Additionally, other interesting road network configurations are inserted. e) Finally, a 3D city model is created. All road network examples were obtained from OpenStreetMap and corresponded to styles extracted from Madrid, San Francisco, Canberra, Tel-Aviv, and London.*

Abstract

Synthesizing and exploring large-scale realistic urban road networks is beneficial to 3D content creation, traffic animation, and urban planning. In this paper, we present an interactive tool that allows untrained users to design roads with complex realistic details and styles. Roads are generated by growing a geometric graph. During a sketching phase, the user specifies the target area and the examples. During a growing phase, two types of growth are effectively applied to generate roads in the target area; example-based growth uses patches extracted from the source example to generate roads that preserve some interesting structures in the example road networks; procedural-based growth uses the statistical information of the source example while effectively adapting the roads to the underlying terrain and the already generated roads. User-specified warping, blending, and interpolation operations are used at will to produce new road network designs that are inspired by the examples. Finally, our method computes city blocks, individual parcels, and plausible building and tree geometries. We have used our approach to create road networks covering up to 200 km^2 and containing over 3,500 km of roads.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—I.3.6 [Computer Graphics]: Methodology and Techniques—

1. Introduction

We present an interactive tool to design and create realistic detailed roads with complex styles. In recent years, urban modeling has received significant attention in computer games, movies, and urban planning. Being able to synthesize and explore large-scale realistic urban road networks is beneficial, for example, to urban procedural modeling (e.g., [Cit]), traffic animation (e.g., [WSL13, GDAU14]), and urban simulation (e.g., [Wad02]).

Various modeling approaches have been proposed to generate realistic roads and/or large-scale road networks. *Manual modeling* of road networks is an option but quickly becomes impractical for large road networks or for frequent modification. *Procedural modeling* is a popular approach to create large urban models including roads, but it requires specialized knowledge to design new parameterized rules and/or to determine the parameter values needed to create desired road patterns (e.g., [PM01]). *Inverse procedural modeling* uses parameter estimation to avoid providing explicit parameter values as input (e.g., [VGDA*12, TLL*10]), but they usually require users to define the rules a priori. While for some 2D and 3D content, rules are inferred automatically (e.g., [vBM*10,BWS10,TYK*12]), to our knowledge there is no automatic rule inference method for road networks. *Example-based modeling* has produced realistic results in image-based texture synthesis [EL99], 3D model synthesis [MM08], and stroke stylization [LYFD12], for example. However, for highly structured content, such as a road network, it is challenging to control a pure example-based method. Altogether, the main challenge is to develop an efficient and easy-to-control method to design, adapt, and synthesize realistic roads of complexity similar to those in the real world without requiring explicitly specified parameterized rules and parameter values.

Our methodology builds upon two key observations. First, a road network can be naturally represented as a geometric graph [Pac04], in which each vertex is associated with a unique point in the Euclidean plane and each edge is associated with a simple curve joining the points associated with its end vertices. Thus, we look to a geometric graph growing system as a framework to synthesize road networks. Second, instead of explicitly requiring predefined road geometries and configurations, we automatically extract road geometries from a set of example road networks. Our system uses roads in OpenStreetMap (OSM) format, which is a free online editable map of the world and currently has over 23 million km [OSM]. We let the user choose one or more example road networks to define the road style to use. Then, our approach uses the example to grow a custom road network occupying a target area. Moreover, our method supports high-level synthesis and tiling operations, as well as some interactive modeling, to yield road networks different than any of the input examples. The end result contains both the details and the realism from the examples yet has the flexibility and detail amplification of procedural modeling.

Our approach consists of a three main phases: 1) during a sketching phase, the user specifies the example road networks (i.e., selected regions from OSM) and may specify the target terrain and some manually-modeled roads (e.g., highways); 2) during road-network processing, our system automatically extracts a set of patches (i.e., meaningful road structures) for example-based growth and a set of statistical features for procedural-based growth; 3) during a growing phase, starting from automatically or user-specified seeds, patches or statistical features are used to grow the road network into the target area while maintaining the style of the example. Furthermore, user-specified procedural growth, warping, blending, and interpolation operations can be used at will to produce new road network designs that are inspired by the selected examples. Since our method automatically finds a large number of patches from the example road networks and does not assume predetermined road geometries, novel detailed and realistic structures similar to any chosen example road network are easily synthesized and modified. Finally, our method creates a plausible city model consisting of blocks, individual parcels, buildings, vegetation, and street lamps (Figure 1).

We have used our approach to create road networks covering up to 200 km^2, containing over 3,500 km of roads, and examples fragments from over 15 cities worldwide. Novel road networks are interactively created in minutes. Using road networks from a newly selected city, or area, is done on-the-fly and with no preprocessing.

The main contributions of our work include:

- an interactive system enabling users to design highly detailed and realistic road networks easily and quickly,
- an effective combination of example-based and procedural-based growth to generate roads with fine details and realism that come from the examples, and
- a set of high-level synthesis and tiling operations, including warping, blending, and interpolation, to produce new styles and to provide seamless connections between multiple grown areas of potentially very different road configurations.

2. Related Work

Our approach builds upon procedural modeling and example-based synthesis. Procedural modeling can, in theory, generate any kind of road network as long as the appropriate rules and parameter values are given. Vanegas et al. [VAW*10] and Musialski et al. [MWA*13] provide surveys of urban procedural modeling and reconstruction. Vanegas et al. [VABW09] combines behavioral modeling and geometrical modeling to produce plausible urban models, but defining the behavioral properties requires some expertise. Smelik et al. [STBB14] discusses several issues that hinder the use of procedural methods. In general, the detail amplification inherent in procedural modeling tends to make the definition of a formal grammar ill-conditioned, even for experts. Defining rules requires a trial-and-error methodology to achieve a desired outcome including road networks.

Inverse procedural modeling may provide users with a higher-level of parameters to tune. For example, Talton et al. [TLL*10] discovers the derivation sequence needed, for a provided grammar, to yield a desired target shape. If this work were to be applied to road networks, it would require addressing the formidable task of designing a predetermined set of rules able to succinctly support all desired road styles and features. Vanegas et al. [VGDA*12] enables the user to specify high-level urban indicators such as the average distance of a house from the street, road tortuosity, and amount of sun exposure per building. The users are not aware of the complex relationship between the input parameter values and the output. But, all rules have to be predefined by the user. Bokeloh et al. [BWSK12] uses grids and symmetries to find pattern primitives. These heuristics work very well for shapes composed of regular patterns, but is more difficult to apply to stochastic patterns, such as trees and road networks. Talton et al. [TYK*12] infers a set of rules for a variety of design patterns but requires a set of labelled hierarchical designs as input.

Although not an inverse procedural modeling method, Chen et al. [CEW*08] provides a tensor field approach yielding an intuitive way to design a road network without explicitly requiring any rules. Roads in the target area are generated according to major and minor streamlines. A user interface helps the user provide some control over the resulting road structure, but it still requires some expertise to obtain a desired output. Moreover, deviating from 90-degree intersections and creating roundabouts and other detailed road patterns is not possible with a pure tensor-field approach. Yang et al. [YWVW13] extends this approach by combining template-based splitting with streamline-based splitting to generate optimized road networks and parcel layouts. More complex pattern of roads and parcels can be produced by pre-designed templates, but designing such templates is not a trivial task. Our example-driven approach supports the addition of any road network pattern without any pre-designed templates or user-performed rule creation.

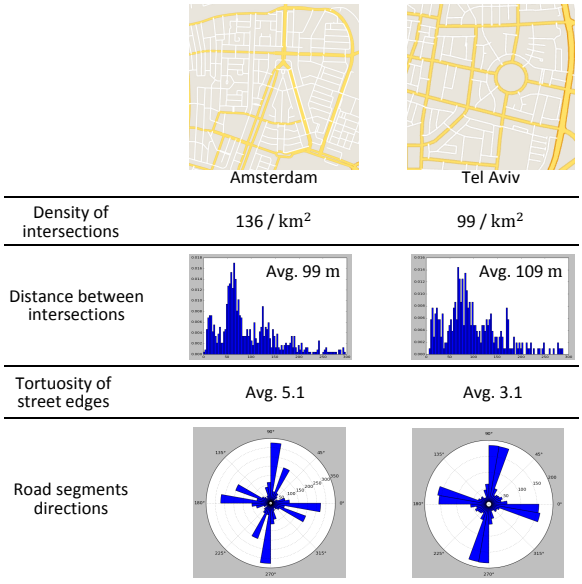

Figure 2: *Visual and statistical comparisons: two road networks look visually different but have similar statistics.*

There are two main variants of example-based methods: parametric methods that extract parameterized features from the examples and non-parametric methods that use examples directly.

- *Parametric methods* are very flexible for well-understood parameters. Aliaga et al. [AVB08] describes a stochastic example-based approach. The generation parameters include the spatial distribution of intersection types, average street length, and average tortuosity. It leads to a flexible system but the parameterization loses many interesting and meaningful road configurations existing in the examples, such as circular plazas, parallel roads, symmetric structures, and explicitly designed road configurations. Figure 2 shows visual and statistical comparisons of two road networks in Amsterdam and Tel Aviv. These two road networks have different styles while their statistics are very similar. This implies that using only statistics cannot fully capture the complex details of the road styles. Based on this observation, we claim that realistic details of road configurations are difficult to represent only by simple statistical features and are lost within the aggregation process.

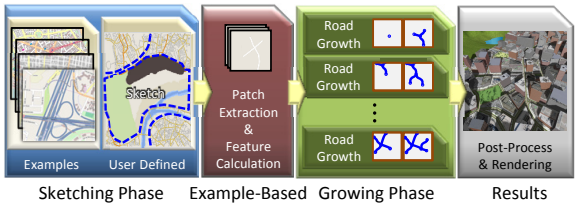

Figure 3: *System pipeline: summary of the entire pipeline of our example-driven road growing system.*

- *Non-parametric methods* are effective for relatively complex but unstructured content such as textures (e.g., [WLKT09, YBY*13]), some 3D models (e.g., [FKS*04, MM08]), and stroke stylization (e.g., [LYFD12]). For roads, however, the problem becomes more difficult. There is no stationarity and locality, as in texture synthesis. Simple blending cannot be used to seamlessly connect and merge adjacent highly varying road styles. Further, road styles lie not only in the global structure, like the overall density and curviness, but also in the local structure such as circular and triangular shapes. Aliaga et al. [ABVA08] decomposes the urban layout into tiles and formulates an urban layout editing operation as the minimization of gap error and tile deformation error. Since the roads are represented by edges of blocks and parcels, the resulting arrangement of tiles naturally exhibits valid road networks. However, it only supports affine transformations of the original urban layout, which makes it difficult to design a complex shape of road networks.

3. Overview

An overview of our method's pipeline is shown in Figure 3. First, a user employs a paintbrush-based tool to draw land use and terrain height within the target area (e.g., an empty area for content generation or a part of an existing city). Our system automatically segments the region into disjoint land use areas labelled as one of water, flat-land (e.g., coast, in-land), mountain, or park. The k-th disjoint land-use area is called the *target domain* Ω_k for road generation. If the user changes the terrain or alters the sketch, then the system automatically updates the sketch-based segmentation. Further, the user associates at least one *example road network* G_k with Ω_k. G_k is decomposed into G_k^A and G_k^S corresponding to its arterial roads and local streets, respectively.

Second, our method detects a set of *patches* and computes a set of statistical *features* for each of the example road networks. Each example road network (i.e., G_k^A and G_k^S) is divided into a set of patches $P_k^A = \{p_{k1}^A, p_{k2}^A, \cdots, p_{kN_k^A}^A\}$ and $P_k^S = \{p_{k1}^S, p_{k2}^S, \cdots, p_{kN_k^S}^S\}$, which allows us to retain interesting road network structures. For example, p_{kj}^A represents the j-th patch of the k-th arterial example graph (more details in Section 4.1). For procedural-based growth, the statistical features including road type (e.g., arterial or local street), road length and orientation, and road curvature are computed.

Third, our example-driven method grows a *target graph* T_k in each target domain Ω_k. Our method first grows only arterial roads and then local streets. To initiate growing, at least one seed a_{ku} for $u \in [1, N_s]$ where N_s denotes the number of seeds is needed inside each Ω_k as an isolated vertex in T_k. The user can define the number of seeds and their locations. Otherwise, $N_s = 1$ and the seed is automatically placed at the centroid of Ω_k. The arterial growing begins by placing all the seeds of T_k into a first-in-first-out (FIFO) vertex processing queue Q. Vertices are dequeued and example-based and procedural-based growth are applied to T_k (Algorithm 1). The user can optionally specify the usage probability of procedural-based growth, local and global warping, blending, and interpolation operations that alter the growth process. Finally, a 3D model of the city is generated. For notational brevity, we typically drop the k subindex as well as the A or S super-index (e.g., p_{kj}^A becomes p_j).

Algorithm 1 Road Growth

```
1: procedure GrowRoads(T, initialSeeds, G)
2:     ExtractPatches(G)                          (§ 4.1)
3:     Compute statistical features of G          (§ 4.2)
4:     Q.push(initialSeeds)
5:     while not Q.empty() do
6:         v ← Q.pop()
7:         q ← example-based growth, or           (§ 4.3.1)
8:             procedural-based growth            (§ 4.3.3)
9:         Add new vertices to Q
```

4. Example-Driven Procedural Roads

In this section, we describe our road generation method. Our method starts with extracting the example's geometry from the OSM format creating a geometric graph. OSM provides road network information such as geo-location of the road intersections (i.e.,

Road types	Type in OSM format
Arterial roads	primary, primary_link, secondary, secondary_link, tertiary, tertiary_link
Local streets	residential, living_street

Table 1: *Mapping of OSM format to our road types*

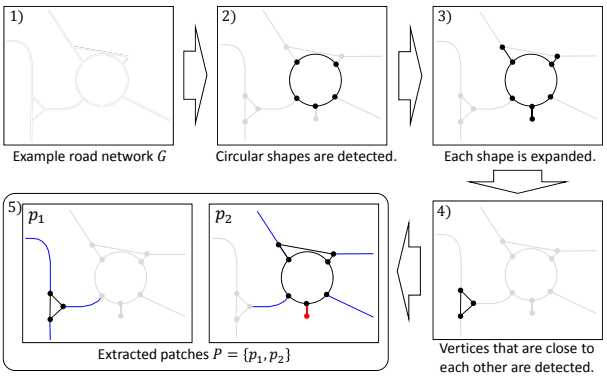

Figure 4: *Subset of the patches extracted from an example road network. 1) Given an example road network G, 2) circular shapes are detected by using a Hough transform. 3) Each detected patch is expanded by Algorithm 2. 4) For the remaining vertices, we select each intersection (i.e., a vertex that has at least three adjoining edges) to create a new patch and expand it by Algorithm 2. Note that vertices that have only one or two edges will not form standalone patches but will become parts of other patches. 5) The remaining edges are added to adjoining patches. In each patch dead-end edges that are not dead end in the original example road graph are marked as connectors (blue lines). Also, dead-end edges in the original example road graph are marked as dead-end (red line).*

graph vertices), road segments (i.e., graph edges and associated polyline), and road segment types. Each road segment is represented by a polyline. Example networks are categorized as arterial roads or local streets by inspecting OSM's road type data (Table 1).

First, we summarize our patch extraction and statistical feature calculation phases in which the examples' geometry from the OSM format is used and then provide details on our multi-seeded road growing phase.

4.1. Example Patches

For each example road network, our method decomposes it into patches $P = \{p_1, p_2, \cdots, p_N\}$ (Figure 4). To keep a meaningful set of road graph edges in a same patch, we detect predefined interesting structures and then expand any remaining intersections into larger units (Algorithm 2). Each road graph edge belongs to one patch except *connector edges*, which are shared between two adjacent patches. Also, dead-end edges are marked as *dead-end*. The user can define interesting structures to be automatically detected using geometric relations (e.g., angle, symmetry, or shape). In our current implementation, our system automatically detects circular shapes (e.g., plazas and roundabouts) using a *Hough transform* and finds tightly clustered vertices (e.g., vertices describing on/off ramps for large arterial roads and major road network interchanges) using a nearest neighbor search.

Algorithm 2 Detect Patches

```
1: procedure ExtractPatches(G)
2:     {patches} ← Detect(G) (e.g. circular shape)
3:     for each patch ∈ {patches} do
4:         Expand(patch, Q)
5:     for each vertex ∈ G\{patches} do
6:         patch ← Make vertex as a patch
7:         Expand(patch, Q)
8: procedure Expand(patch, Q)
9:     Put all the vertices of patch into Q
10:    while not Q.empty() do
11:        v ← Q.pop()
12:        for each u ∈ adj(v) do
13:            e ← edge(u, v)
14:            if e.length() < threshold then
15:                Add u and e to patch
16:                Q.push(u)
```

4.2. Statistical Features

The system computes the average and the variance of the length ($\bar{l}$ and $V(l)$) and the curvature ($\overline{\kappa}$ and $V(\kappa)$) of the edges of each graph. For the curvature, we use the following approximation [ABS02] to compute the average curvature of an edge's polyline:

$$\kappa = \frac{\sum_m \left\| \frac{z_{m+1}}{\|z_{m+1}\|} - \frac{z_m}{\|z_m\|} \right\|}{\sum_m \|z_m\|}, \qquad (1)$$

where z_m is m-th line segment of the polyline. The statistical features are used for procedural-based growth.

4.3. Arterial Road Generation

After the patch extraction and feature calculation, growing commences from each seed placed in the target domain. Our approach uses a breadth-first graph growing method for each domain Ω_k. At each step, graph vertex v_i is dequeued from Q, and either one of two types of growth is performed: example-based (Section 4.3.1) or procedural-based (Section 4.3.3). Our system, by default, uses example-based growth. Procedural-based growth is used if 1) all the detected patches fail to pass the *patch legality checks* (Section 4.3.2), i.e., it is not possible to find a proper patch or 2) the user chooses to use the interpolation operation (Section 5.3). For procedural-based growth, *local constraints* are also taken into account in order to adapt to the underlying terrain and the already generated roads (Section 4.3.4). After example-based or procedural-based growth is applied, the new exterior

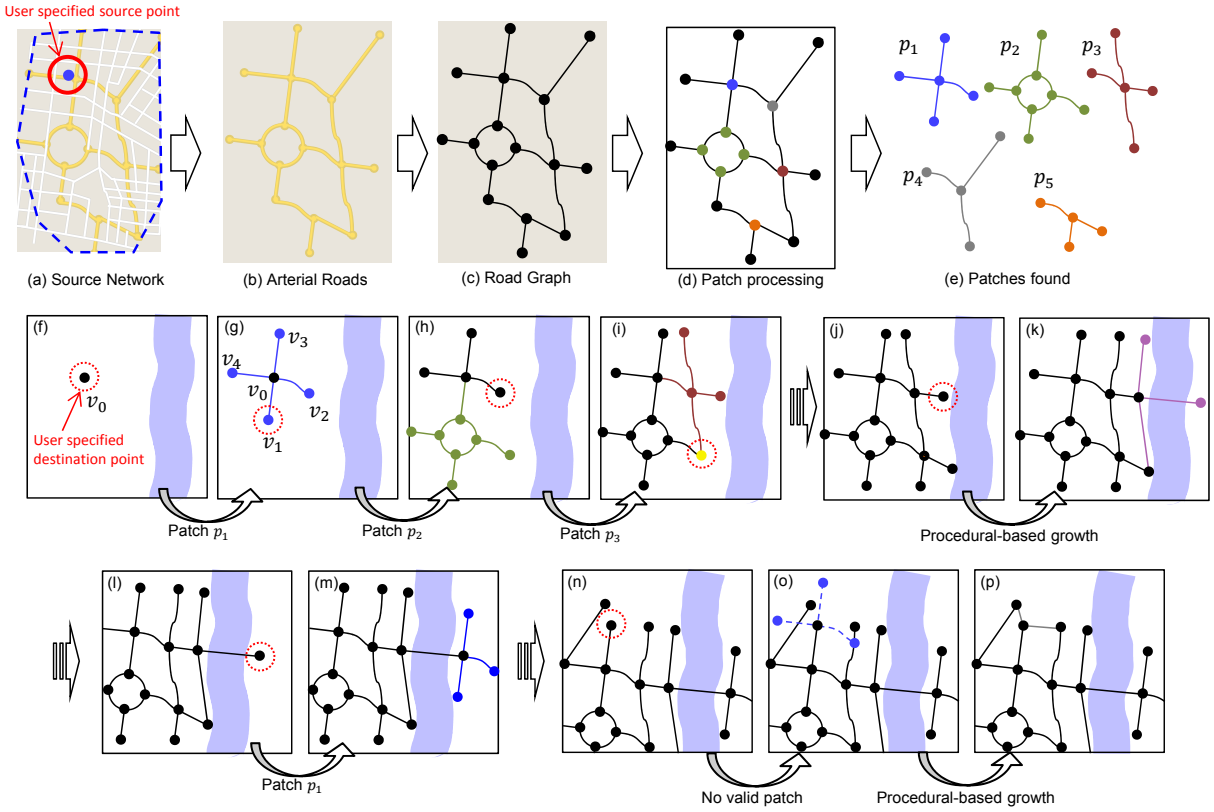

Figure 5: *This figure depicts the road growing process. a)-e) The patches are extracted from the source example. f) For the user-specified destination point (seed), patch p_1 that is closest to the user-specified source point in the example is used to grow the target graph. g) The new exterior vertices v_1, v_2, v_3, and v_4 are enqueued to Q and the next vertex v_1 is dequeued from Q. h) Patch p_2 is selected according to the usage probabilities. i) This probabilistic random selection may choose patch p_3 even though patch p_4 has higher similarity and higher usage probability. Once a patch is selected, it is copied and translated to the position of v_i. If a new vertex of the patch is close to an existing vertex, they are merged into one vertex (yellow dot). j) If no example patch has a non-zero usage probability because of the surrounding environment such as the river in this case, k) procedural-based growth will be performed. l) For a vertex that is generated procedurally, m) an example patch is selected according to the usage probabilities in a similar manner. n) Another case of using procedural-based growth is o) when no example patch has a non-zero usage probability because it overlaps with the already generated roads. p) In this case, procedural-based growth is used so that the generated roads will be connected to the existing roads.*

vertices are enqueued into Q. Once road generation completes (i.e., Q is empty, or all road segments intersect the boundary of the target domain) the user can optionally apply some automatic post-process (Section 4.5) to filter small undesired road segments (e.g., short dangling edges not from an example patch).

Figure 5 visually presents steps from the OSM file loading to road generation. Starting from one or more user-specified destination points, which are used as seeds, example-based or procedural-based growth is performed at each step. Notice that example-based growth preserves some interesting structures but may not be used because of the underlying terrain and the already generated roads while procedural-based growth can complement this issue.

4.3.1. Example-Based Growth

For example-based growth, a patch $p_j \in P$ is randomly selected according to the usage probability $Pr(p_j)$, which is defined by

$$Pr(p_j) = \frac{e^{-\min_{k,l} dist_H(e_{p_j,l}, e_{v_i,k})} \times I(p_j)}{Z}, \qquad (2)$$

where $dist_H(e_1, e_2)$ denotes *Hausdorff distance* between edges e_1 and e_2, $e_{p_j,l}$ denotes the l-th connector edge of patch p_j, $e_{v_i,k}$ denotes the k-th edge connected to the current vertex v_i, $I(p_j)$ is an indicator function that is 1 if the patch passes the legality checks (Section 4.3.2) and 0 otherwise, and Z is a partition function to normalize the probability. In this manner, the selected patch p_j is likely to well fit to v_i.

Once a patch p_j is selected, the patch is copied and translated to the position of v_i. Then, the edges of p_j that are redundant to the existing edges of T are removed and p_j is glued to T. Note that p_j may have a vertex whose coordinates are very close to one of the existing vertices (e.g., yellow dot in Figure 5i). In this case, those two vertices will be merged into one vertex, so the resulting graph is well connected. If there is no valid patch, i.e., no patch has a non-zero usage probability, procedural-based growth will be used (Section 4.3.3).

4.3.2. Patch Legality Checks

To compute the usage probability for each patch, the following legality checks are performed:

- check whether all the edge segments of the patch are above water level,
- check whether the underlying terrain slope of the patch is within a threshold, and
- check whether the patch does not intersect with any road segments of T.

If the patch fails to pass the legality checks, its usage probability becomes 0 according to Equation (2).

4.3.3. Procedural-Based Growth

When all the patches fail to pass the legality checks or the user chooses to use the interpolation operation, procedural-based growth is applied. Unlike example-based growth, this type of growth uses the statistical feature information of the source example, $\bar{l}$, $V(l)$, $\bar{\kappa}$, and $V(\kappa)$, which are used to define the length and the curvature of the new road segments. Given the current vertex v_i and already generated edges connected to v_i, additional (polyline) edges are synthesized so that the angle between each adjacent edges is close to 90 degrees.

4.3.4. Local Constraints

For each potential road segment created by procedural-based growth, some local constraints are taken into account in order to adapt to the underlying terrain as well as the already generated roads (e.g., similar to the *localConstraints* in [PM01]). In particular, our method executes the following:

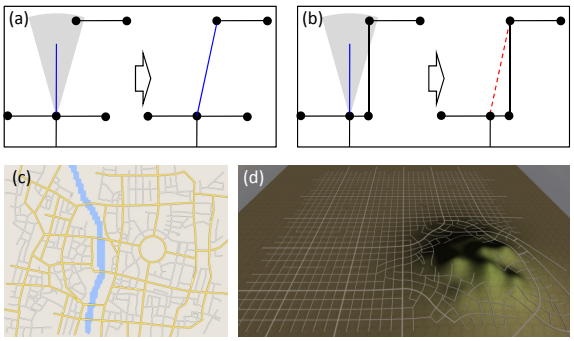

Figure 6: *The road segments generated by procedural-based growth might be a) connected to the already generated roads if they are close, b) discarded because of the acute angle, or adjusted to the underlying terrain such as c) the river and d) the mountains.*

- If there is an existing vertex or edge that is very close to v_i in the direction within a certain range from the road segment, extend the road segment to the closest one to connect (Figure 6a). However, if there is already an existing edge such that the angle between the existing edge and the new edge is too acute, the new edge will be discarded (Figure 6b).
- If the road type is local street and a part of the road segment is below water level, the road segment will be discarded.
- If the road type is an arterial road and a part of the road segment is below water level, extend the road segment until it reaches the other side of the river. A change in the orientation of the road segment is allowed up to a certain range so that the road segment can cross the river with a shorter distance (Figure 6c). If it does not reach the land (i.e., it is going to the ocean), the road segment is discarded.
- If the underlying terrain slope is steep, the road segment will be bended to be perpendicular to the local terrain gradient. However, if the slope is too steep, the road segment will be removed (Figure 6d).

4.4. Local Streets Generation

Once arterial roads are grown, local streets are seeded and grown. The method by which local-street seeds are generated depends on whether the edge was grown by example-based or procedural-based growth. For edges created by example, all the intersections with the local streets along the edge in the example will be used as the initial seeds for local streets generation (Figure 7). These initial seeds also contain the corresponding example patch of the local streets; thus, example-based growth is used. After that, the local streets are generated in the same manner as the arterial roads gener-

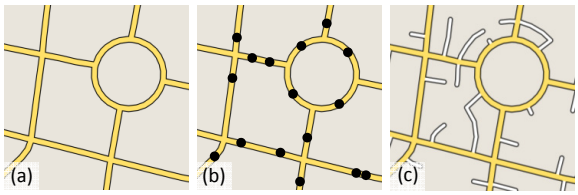

Figure 7: *Local street generation: a) for the edges generated by an example-patch-based rule, b) all the intersections along the edge with local streets in the source example are used to populate the initial seeds for local street generation (black dots). c) Then, the local streets start growing from these initial seeds (white lines).*

ation. In this way, the generated local streets have a similar pattern to the example.

For the edges created by procedural-based growth, we use the statistical features to divide those edges into multiple small fragments and put initial seeds at the boundary between fragments. Then, starting from the initial seeds, local streets generation is processed in the same way as the previously discussed arterial road generation.

4.5. Post-process

After the road generation completes, the user can optionally apply some automatic post-process to remove dangling edges. Starting from a vertex that has only one adjoining edge, the connected edge is removed. However, if the vertex is generated by an example patch and it is marked as dead-end, it is kept without being removed. We iterate this process until there is no such vertex (Figure 8).

5. Synthesis and Tiling Operations

In this section, we describe our high-level synthesis and tiling operations. In addition to using the example roads, these operations allow users to control growth by tuning style in an intuitive manner.

5.1. Warping

Our approach uses piecewise linear transformations to support warping an example road network to its target domain (Figure 9a). Image-based warping is a well-studied approach that defines a point-to-point mapping from a source to a target and distorts the image according to the mapping. The distortion may be acceptable or even make creative results for images. But for road networks, the distortion is undesired in most cases. Our warping operation ensures that example

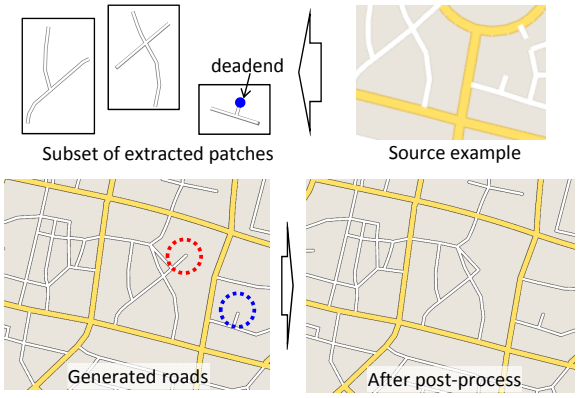

Figure 8: *The post-process filtering removes the dangling edges (red dotted circle) while those that is marked dead-end in the source example (blue dot in the patch) is kept without being removed (blue dotted circle).*

patches grown from the same initial seed use a similar transformation. Hence, the relative position of adjacent intersections is preserved and consequently the local structure does not change much.

To perform a warp, a user provides a guideline with corresponding control points in the example and in the target domain. The control points in the target domain are also used as multiple initial seeds a_u for road generation (i.e., $N_s > 1$). Given guideline segments $l_u^E (u = 1, 2, \cdots, N_{s-1})$ in the example and $l_u (u = 1, 2, \cdots, N_{s-1})$ in the target domain, the rotation angle for seed a_u is

$$\phi(a_u) = \frac{\rho(l_u) + \rho(l_{u+1})}{2} - \frac{\rho(l_u^E) + \rho(l_{u+1}^E)}{2}, \quad (3)$$

where $\rho(l_u)$ represents the orientation of the line segment l_u.

Then, starting from the initial seeds, the road generation algorithm described in Section 4.3 is applied. When a patch p_j is selected for the current vertex v_i, the world-space rotation angle defined for the original seed is applied so as to rotate the patch. In this way, all the road graph fragments grown from the same initial seed are in the same orientation. Thus, the relative locations of the adjacent intersections are preserved except when two neighboring intersections are grown from different seeds. In this case, it is possible that two edges outgoing from two adjacent intersections cross each other, but our legality checks can handle such a situation as described in Section 4.3.

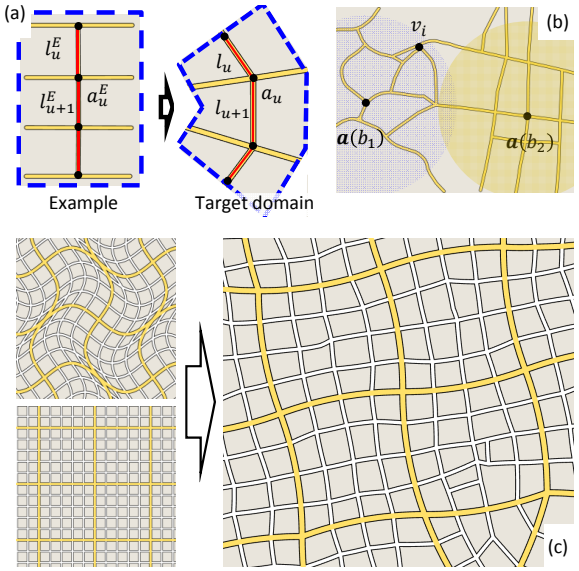

Figure 9: *Synthesis operations: a) warping: a piecewise linear transformation is computed based on user-specified guidelines. b) Blending: given a vertex v_i, the probability $Pr(b_s)$ of using example b_s ($s = 1, 2$ in this example) is a random variable with a Gaussian distribution. Its standard deviation is controlled by the user. c) Interpolation: a mixture of two different styles is synthesized by computing a weighted average of feature information (i.e., the length and the curvature of the edges in the shown two examples).*

5.2. Blending

We also provide a blending operation by which the roads grown using more than one example can gradually change from the style of one example to the style of another (Figure 9b). This ability mitigates sudden changes on the boundary between different styles which might cause unexpected road interruptions and/or oddly-shaped connecting roads. Since the sudden change can be desired in some areas, a user-specified parameter determines the width of the blending area.

Given a vertex v_i and a set of blending example networks $B = \{b_1, \cdots, b_{N_B}\}$, where typically $N_B = 2$, the probability $\Pr(b_s)$ of using example b_s to construct patches is defined as

$$\Pr(b_s) = \frac{1}{\sum_t \Pr(b_t)} \exp\left(\frac{\|\boldsymbol{x}(v_i) - \boldsymbol{x}(a(b_s))\|^2}{2\sigma_{b_s}^2 D^2}\right), \quad (4)$$

where $\boldsymbol{x}(v_i)$ are the coordinates of the vertex v_i, $a(b_s)$ is the initial seed for example b_s, σ_{b_s} is the parameter to control the probability distribution, and D is to normalize the distance between v_i and $a(b_s)$. This probabilistic selection generates a smooth transition from the style of one example to that of another.

5.3. Interpolation

Interpolating example road networks enables the creation of novel networks appearing to be *inspired* by the examples but are different. There is no general way to interpolate two graphs. Even if the two graphs have the same structure, finding corresponding edges is a NP problem and is known as the graph isomorphism problem [KST93]. To avoid this expensive computation, we use procedural-based growth for interpolation. The length and curvature feature information of the edges to interpolate are used to grow the target graph. Let A and B be the two different example styles and let $\bar{l}_X$, $V(l_X)$, $\bar{\kappa}_X$, and $V(\kappa_X)$ be the average and the variance of the length and the curvature of the edges in style X (i.e., $X = A$ or $X = B$). Then, the interpolated length l^* and curvature κ^* are computed as the weighted average of two styles:

$$\begin{cases} l^* = l_A^* t + l_B^*(1-t) \\ \kappa^* = \kappa_A^* t + \kappa_B^*(1-t) \end{cases} \text{for } 0 \le t \le 1, \quad (5)$$

where $l_X^* \sim \mathcal{N}(\bar{l}_X, V(l_X))$ and $\kappa_X^* \sim \mathcal{N}(\bar{\kappa}_X, V(\kappa_X))$ are a random variable with a Gaussian distribution, and the weight t is controlled by the user. This interpolation yields the target graph having a mixture of two different styles. For example, a regular grid and a very curvy grid can make a mildly curved grid (Figure 9c).

6. Results

We have used our method to produce road networks using examples from all over the world. Our system is implemented in C++, using OpenGL, and reads OSM files as examples. All growing is performed on an Intel Core i7 CPU and takes 2 minutes on average for generating around 300 km of roads (Table 2).

Figure	Total length	Generation time
1	278 km	49 seconds
10a	35 km	1 seconds
10b	150 km	7 seconds
11e	195 km	28 seconds
11f	228 km	40 seconds
11g	314 km	128 seconds
12c	210 km	83 seconds
13c	330 km	90 seconds
13f	337 km	131 seconds
14e	342 km	130 seconds

Table 2: *Computation time for road generation in this paper.*

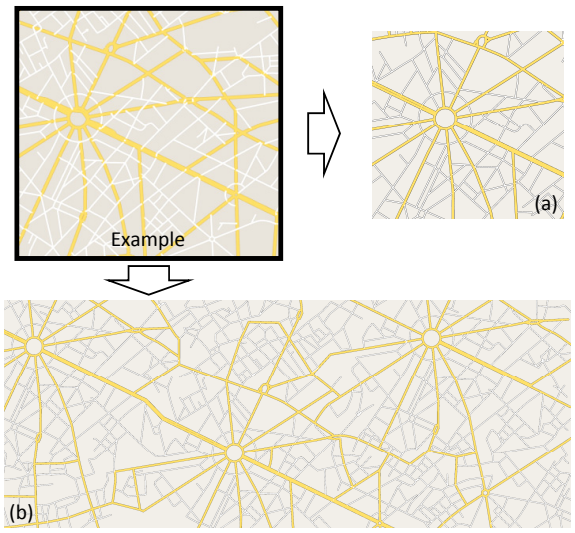

Figure 10: *Single-example road generation: roads were generated by using an example road network from Paris. Roads were grown a) from one initial seed and b) from three initial seeds in the target area.*

As results, we first show a closer view of a single-example road generation process. The generated roads maintain a look and feel similar to the example. We used our program to design a new road network using the roads from an example fragment in Paris. The user specified the example area, the target area, and one initial seed for the smaller result and three initial seeds for the larger result in this example. Our system generated the target road network automatically. Notice how the circular plaza structure appears multiple times and in a manner similar to the example.

Figures 11 and 12 show some examples of our synthesis operations. Figure 11 demonstrates blending with different blending values. The roads on the left side exhibit a curved pattern from Canberra. The roads on the right side exhibit a dense grid pattern from Tokyo. The roads in the middle area of Figures 11e-g show a blended pattern obtained by spatially varying the blending value according to the distance from the initial seeds. This blending demonstrates a smooth user-controlled transition between two different patterns.

Figure 12 shows our warping operation. A road network fragment from Paris is used as an example. The warping operation is applied according to the user-specified guideline describing an "S" shape. In contrast, using existing image-based warping, such as the Puppet Tool from Adobe Photoshop, distorts those same meaningful structures, and thus results in undesired road networks. Notice that interesting patterns in the example are preserved by our warping operation, while some unnatural shapes of blocks might be generated because of the sharp curve of the user guideline. Determining the validity of an urban layout is a complex issue, and it depends on the application [LSWW11], so we provide warping as an optional operation to enhance user expressiveness.

Figures 13 and 14 show the result of two design sessions. In Figure 13, a user starts with an empty target area. The user first draws a sketch of a city and provides various example road network fragments tagged to be from various different terrain elevations — in total 6 examples: Amsterdam, Barcelona, Canberra, New York, Canberra, and Quito. Then, a plausible terrain is produced from the sketch and roads are automatically generated. As part of the content creation process, the user decides to modify the terrain by adding some mountains. Our system re-segments the terrain and searches, indexed by terrain height, for the most suitable example fragments to use. In Figure 14, a user performs interactive exploratory planning for an urban redevelopment project using our system. The user starts with an urban area and replaces some parts by different road styles using warping, blending, and interpolation operations. Then, the user adds a park with some hand-drawn roads. A large area of roads can be quickly designed in a similar manner. In the final result (Figure 14g), the generated road network covers 200 km^2 and contains over 3,500 km of roads. Finally, an entire city model is created (Figure 14h).

Discussion. Our approach uses the example patches to preserve the styles from examples, while the procedural-based growth is used to fill the gap between patches. Even though the procedural-based growth uses some statistical features from the source examples, most of the interesting details are lost. Thus, the rate of example-based growth indicates how much the styles are preserved in the generated road networks (Table 3). The rate becomes low when the underlying terrain is not flat or the road growth starts from multiple seeds.

Figure	Rate of example-based growth
10a	91 %
10b	66 %
11e	54 %
11f	63 %
11g	58 %
13c	49 %
13f	44 %

Table 3: *Rate of example-based growth.*

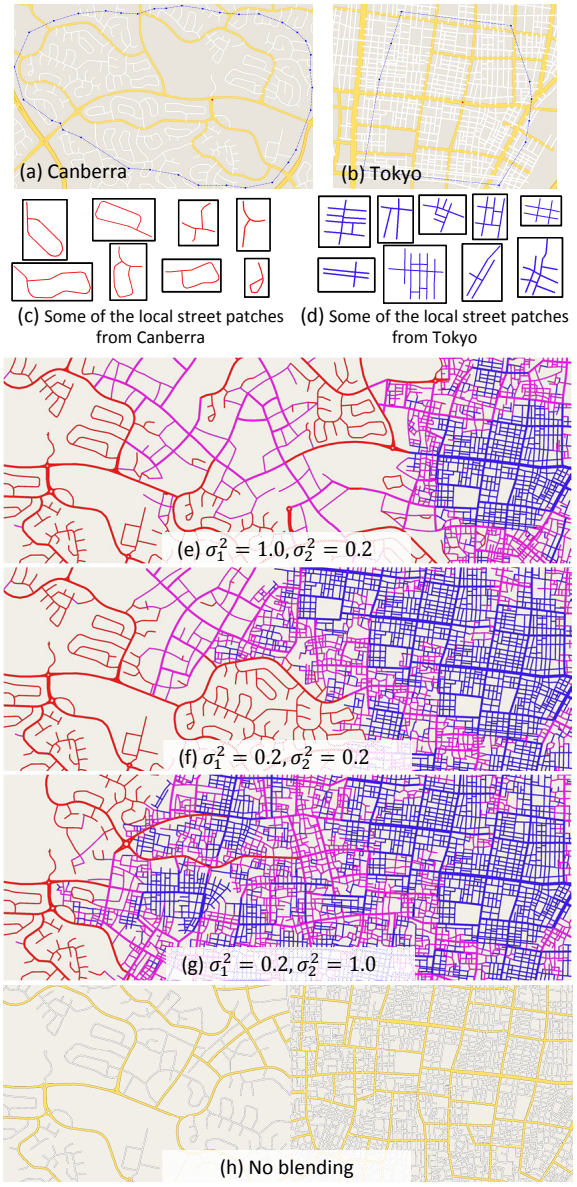

Figure 11: *Blending: a) the curved roads from Canberra are joined together with b) the grid pattern from Tokyo. The patches of c) Canberra and d) Tokyo are extracted from the example roads. The blending operation makes the roads in the middle area exhibit a smooth transition between two different styles. The user can control the transition by changing the blending values as shown in e), f), and g). The patches from Canberra and Tokyo are colored red and blue, respectively, while the roads generated by procedural-based growth are colored purple. h) Adjoining two styles of roads without blending causes a sudden change in the road network along the border that cannot easily be solved with image-based blending.*

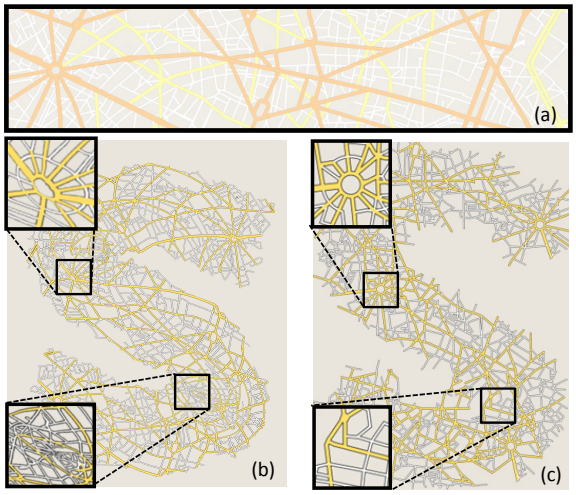

Figure 12: *Warping: a) given an example road network from Paris, b) image-based warping distorts, or collapses, meaningful road shapes. c) In contrast, our warping operation preserves the local shapes such as plazas.*

Limitations. Our method is not without limitations. For instance, without at least some user guidance our system can only produce road configurations present in an example or resulting from our interpolation method. Further, our interpolation only works for procedural-based growth and not example-based growth. In addition, our naïve queue-based growth process might not find the optimal growth sequence in the sense of most accurately and efficiently producing desired road geometry.

7. Conclusions and Future Work

We presented an interactive road designing system using the patches and statistical information extracted from the examples to grow roads. Using a sketching interface, high-level synthesis operations (i.e., warping, blending, and interpolation), user-defined seeds, and probability distributions our approach quickly produces realistic details comparable to those in the example road networks. Once roads are generated, an entire city model is instantiated using procedural modeling. Our results suggest that our example-driven road growing framework is very flexible and delivers a fast tool for road network design and exploration.

There are many possible extensions to our work. First, we would like to add a road-geometry search interface so that a user can sketch a desired road network shape and the system finds it amongst a large database of examples. To define the similarity between the user

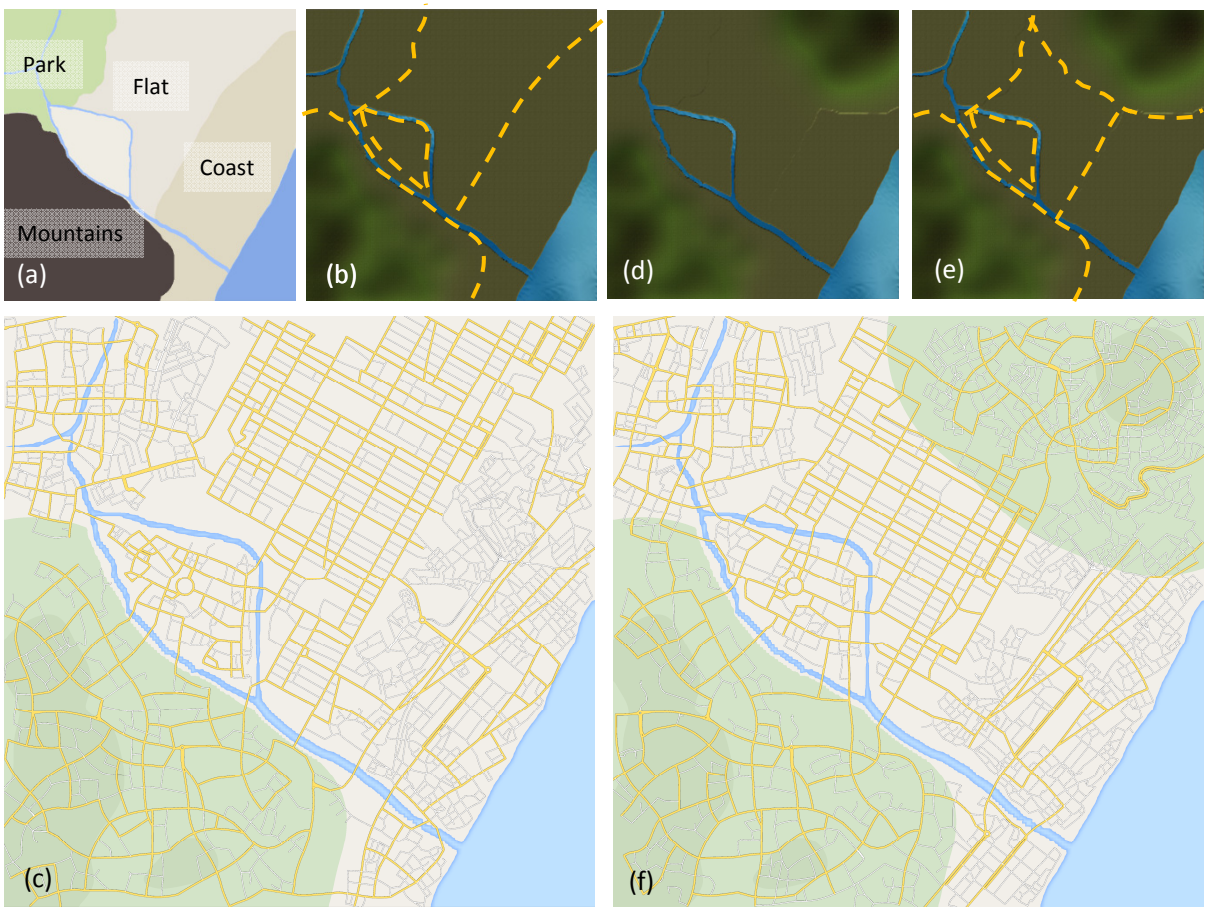

Figure 13: *Content creation: this figure demonstrates a session in which the user designs an entire city. a) The user draws a sketch of a city defining different land-use areas. b) Then, the system automatically generates a terrain from the sketch. The undulations of the mountains are randomly generated by the system. The system finds similar examples for each area from a database of examples using the average elevation in the area, and c) the road networks are generated. d) The user updates a part of the terrain – mountains are added in the top right corner, and e) the segmentation is automatically updated. f) The examples are re-selected based on the updated segmentation and the average elevation of each area, and the roads are re-generated based on the examples.*

sketch and the roads in the database, the underlying terrain will be useful to consider. Second, even though our local constraints consider the rivers and the slope of the mountains, there are still some poorly connected roads across rivers with sharp turning directions. This issue can be improved by thoroughly and globally taking into account the underlying terrain during growth (e.g., [GPMG10, GPGB11, EBP*12]). Also, as a post-process, some global optimization can be applied to make the generated road networks more plausible based on some user-defined criteria. Third, an extension is to incorporate more high-level editing operations such as merging by using graph cut [LSWW11].

8. Acknowledgements

We would like to thank the reviewers for their helpful feedback. This research was partially funded by NSF 1250232, NSF 0964302, and NSF 1302172.

References

[ABS02] Anoshkina E., Belyaev A. G., Seidel H.-P.: Asymptotic analysis of three-point approximizations of

vertex normal and curvatures. In *Vision, Modeling, and Visualization* (2002), pp. 211–216. 5

[ABVA08] Aliaga D. G., Beneš B., Vanegas C. A., Andrysco N.: Interactive reconfiguration of urban layouts. *IEEE Computer Graphics and Applications 28*, 3 (2008). 4

[AVB08] Aliaga D. G., Vanegas C. A., Beneš B.: Interactive example-based urban layout synthesis. *ACM TOG 27*, 5 (2008). 3

[BWS10] Bokeloh M., Wand M., Seidel H.-P.: A connection between partial symmetry and inverse procedural modeling. *ACM TOG 29*, 4 (2010). 2

[BWSK12] Bokeloh M., Wand M., Seidel H.-P., Koltun V.: An algebraic model for parameterized shape editing. *ACM TOG 31*, 4 (2012). 3

[CEW*08] Chen G., Esch G., Wonka P., Müller P., Zhang E.: Interactive procedural street modeling. *ACM TOG 27*, 3 (2008). 3

[Cit] CityEngine: http://www.esri.com/software/cityengine. 2

[EBP*12] Emilien A., Bernhardt A., Peytavie A., Cani M.-P., Galin E.: Procedural generation of villages on arbitrary terrains. *Visual Computer 28*, 6-8 (2012), 809–818. 12

[EL99] Efros A. A., Leung T. K.: Texture synthesis by nonparametric sampling. In *IEEE ICCV* (1999), vol. 2, pp. 1033–1038. 2

[FKS*04] Funkhouser T., Kazhdan M., Shilane P., Min P., Kiefer W., Tal A., Rusinkiewicz S., Dobkin D.: Modeling by example. *ACM TOG 23*, 3 (2004). 4

[GDAU14] Garcia-Dorado I., Aliaga D. G., Ukkusuri S. V.: Designing large-scale interactive traffic animations for urban modeling. *Computer Graphics Forum 33*, 2 (2014), 411–420. 2

[GPGB11] Galin E., Peytavie A., Guerin E., Beneš B.: Authoring hierarchical road networks. *Computer Graphics Forum 30*, 7 (2011), 2021–2030. 12

[GPMG10] Galin E., Peytavie A., Maréchal N., Guérin E.: Procedural generation of roads. *Computer Graphics Forum 29*, 2 (2010), 429–438. 12

[KST93] Kobler J., Schöning U., Toran J.: *The Graph Isomorphism Problem*. Birkhäuser Computer Science, 1993. 9

[LSWW11] Lipp M., Scherzer D., Wonka P., Wimmer M.: Interactive modeling of city layouts using layers of procedural content. *Computer Graphics Forum 30*, 2 (2011), 345–354. 10, 12

[LYFD12] Lu J., Yu F., Finkelstein A., DiVerdi S.: Helpinghand: Example-based stroke stylization. *ACM TOG 31*, 4 (2012). 2, 4

[MM08] Merrell P., Manocha D.: Continuous model synthesis. *ACM TOG 27*, 5 (2008). 2, 4

[MWA*13] Musialski P., Wonka P., Aliaga D. G., Wimmer M., Gool L. V., Purgathofer W.: A survey of urban reconstruction. *Computer Graphics Forum 32*, 6 (2013), 146–177. 3

[OSM] OSM: http://www.openstreetmap.org. 2

[Pac04] Pach J.: Towards a theory of geometric graphs. *Contemporary Mathematics 342* (2004). 2

[PM01] Parish Y. I. H., Müller P.: Procedural modeling of cities. In *ACM SIGGRAPH* (2001), pp. 301–308. 2, 7

[STBB14] Smelik R. M., Tutenel T., Bidarra R., Beneš B.: A survey on procedural modeling for virtual worlds. *Computer Graphics Forum 33*, 6 (2014), 31–50. 3

[TLL*10] Talton J. O., Lou Y., Lesser S., Duke J., Měch R., Koltun V.: Metropolis procedural modeling. *ACM TOG 30*, 2 (2010). 2, 3

[TYK*12] Talton J. O., Yang L., Kumar R., Lim M., Goodman N. D., Měch R.: Learning design patterns with bayesian grammar induction. In *Proceedings of the 25th UIST* (2012), pp. 63–74. 2, 3

[VABW09] Vanegas C. A., Aliaga D. G., Beneš B., Waddell P. A.: Interactive design of urban spaces using geometrical and behavioral modeling. *ACM TOG 28*, 5 (2009). 3

[VAW*10] Vanegas C. A., Aliaga D. G., Wonka P., Müller P., Waddell P., Watson B.: Modeling the appearance and behavior of urban spaces. *Computer Graphics Forum 29*, 1 (2010), 25–42. 3

[vBM*10] Šťava O., Beneš B., Měch R., Aliaga D. G., Krištof P.: Inverse procedural modeling by automatic generation of l-systems. *Computer Graphics Forum 29*, 2 (2010), 665–674. 2

[VGDA*12] Vanegas C. A., Garcia-Dorado I., Aliaga D. G., Beneš B., Waddell P.: Inverse design of urban procedural models. *ACM TOG 31*, 6 (2012). 2, 3

[Wad02] Waddell P.: Urbansim: Modeling urban development for land use, transportation and envionrmental planning. *Journal of the American Planning Association 68*, 3 (2002), 297–314. 2

[WLKT09] Wei L., Lefebvre S., Kwatra V., Turk G.: State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR* (2009). 4

[WSL13] Wilkie D., Sewall J., Lin M.: Flow reconstruction for data-driven traffic animation. *ACM TOG 32*, 4 (2013). 2

[YBY*13] Yeh Y., Breeden K., Yang L., Matthew F., Hanrahan P.: Synthesis of tiled patterns using factor graphs. *ACM TOG 32*, 1 (2013). 4

[YWVW13] Yang Y., Wang J., Vouga E., Wonka P.: Urban pattern: layout design by hierarchical domain splitting. *ACM TOG 32*, 6 (2013). 3

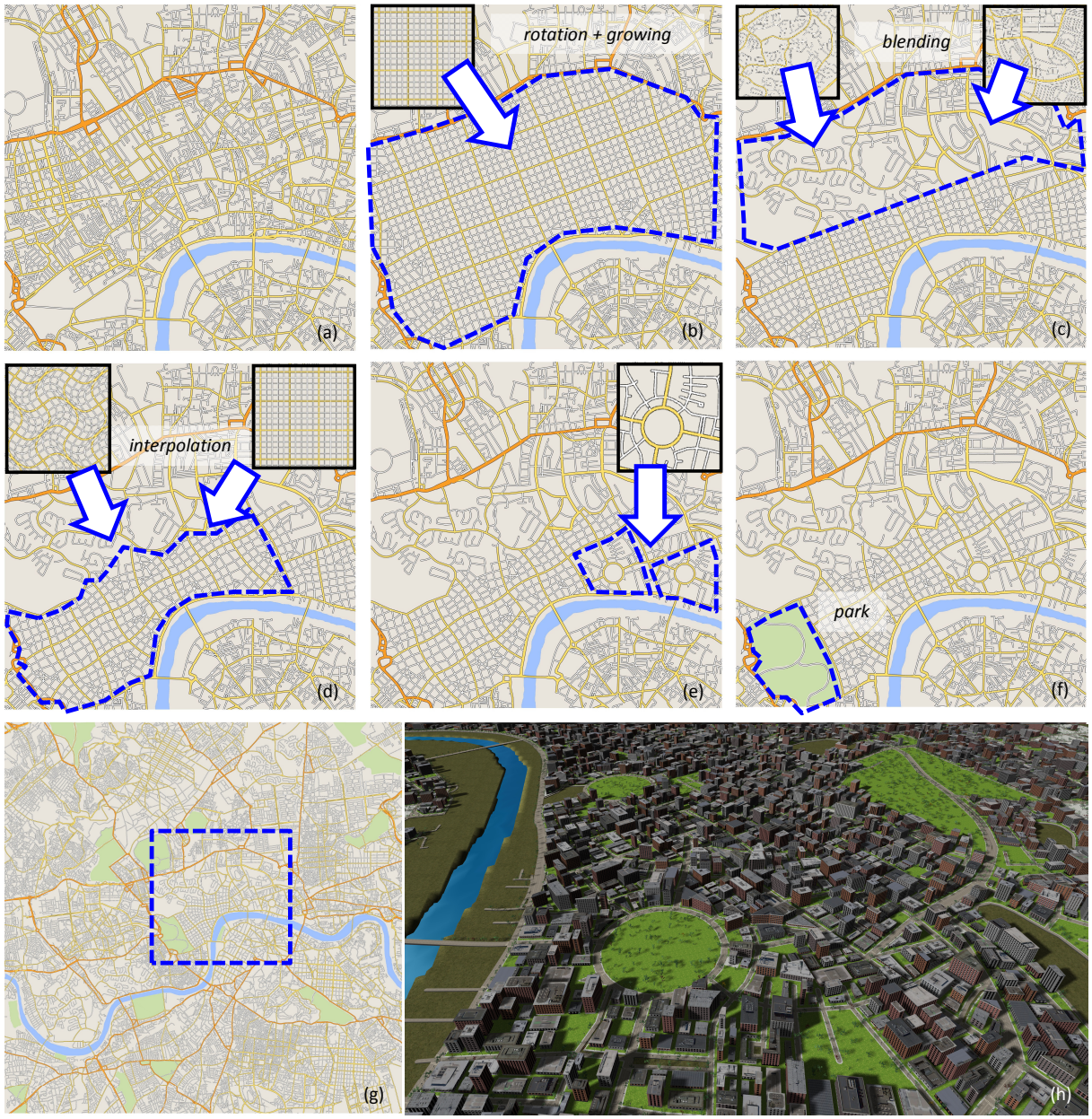

Figure 14: *Redevelopment of a city: our method allows untrained users to quickly design roads with complex styles. The user can start designing roads from scratch, or can start with any existing roads in OpenStreetMap file format. a) The user starts with an urban area and b) replaces the target area with a regular grid pattern of roads having a slight rotation. c) The user places a residential area road network in the top half based on two blended examples. d) In the bottom half, an interpolated pattern of the regular grid and of a curved grid is added. e) The user places plaza-shaped roads at the central business district by warping some example roads. f) Finally, the user creates a park and adds some hand-drawn roads. g) In a similar manner, the user can expand the result from step (f) to create a large area of roads that fills* $14km \times 14km$*. Note that the blue dashed box highlights the area in step (f). h) After road generation, an entire city model is procedurally created using this road network.*