Volume xx (200y), Number z, pp. 1–5

Example-based Road Network Synthesis

Q. Yu and A. Steed

Department of Computer Science, University College London, UK

Abstract

We present a novel method for automatically synthesizing road networks that are perceptually similar to given example road networks. Our algorithm can grow a new road network from an initial node or expand an existing network. For an unfinished node in the network, we search a set of candidate nodes in the example road networks that have similar node topology, and then find a best one by neighborhood edge matching. Rather than simply copy edges incident to the best matched node, we consider the local constraints including obstacle avoidance and relationships to existing roads. Our results demonstrate that the method produces good results for various typical street patterns. Furthermore, we can blend styles of road networks in an intuitive and easy to control manner.

1. Introduction

Within the urban environment roads are one of the most important structuring elements. A tool for generating a novel road network should provide for good variety of output, but allow the artist an intuitive control over that output. Thus whilst many forms of procedural generation have been proposed for generation of road networks, these suffer from the problem that the output is not easily customisable.

In this paper, we attempt to marry the goals of variety of output with artistic control by using a form of example-based modelling. Our method takes as input a single example or a set of examples of the style of road network that is desired. We can then create a new road network from scratch, or augment an existing input network. The process iteratively expands the road network from the end of a road (a node). It does this by finding similar nodes in the example data. However, rather than simply copying some segment of the example data across, we apply local constraints to that segment. In doing so, we preserve characteristics of the example's style, while making sure that the new road network is plausible.

Our algorithm allows for a lot of variety in output, as do completely procedural methods [PM01]. Indeed it includes the local constraint step that is a key component of the completely procedural models. The output is visually similar to input examples as in [AVB08], but allows flexible control of output as per [CEW*08]. The main contribution of our algorithm is in providing these properties of both of these algorithms within a single novel algorithm and framework.

2. Related work

Procedural methods have been used for automatically generating roads following various constrains [PM01, GPGB11]. Simulation-based methods [Wad02, VABW09] can capture high-level properties that are difficult to model by production rules, but still lack intuitive control. Chen *et al.* [CEW*08] propose to create street networks though operating on tensor fields. We can achieve similar results as their work, but provide control by user input samples.

Example-based synthesis approaches have achieved considerable success for creating textures that are perceptually similar with given samples [EL99]. The example-based approach has been extended to modeling various geometric objects such as curves [HOCS02] and 3D models [Mer07]. Ijiri *et al.* [IMIM08] use neighborhood matching for growing an underlying triangular mesh. This work is algorithmically the most similar to ours, but we support more general vector shapes. Having a similar goal as ours, Aliaga *et al.* [AVB08] propose an example-based method for urban layout synthesis. There are several differences between our work and theirs. First, direct reuse of the intersection points from a sample network in their method will yield an output identical to the input (see Fig. 8 in [AVB08]). To expand the sample and get variation, the user must manipulate the intersection points. Our method works more like a texture synthesis algorithm which can generate a larger output with variation automatically. Second, their method relies on intersection points encoded with street styles, which is not suitable for capturing street patterns that contain dangling roads and long curved roads. Our method does not have this limitation. Finally, our road segment generation fits within a procedural modelling framework and can be extended with other production rules.

3. The algorithm

Our algorithm represents a road network with a planar graph $G = (V, E)$, where the nodes V are intersections, dead ends, or feature points along a road, and the linear edges E are street segments between the nodes. Given a sample road network $G_s = (V_s, E_s)$, our goal is to generate a new road network $G_d = (V_d, E_d)$ that resembles the input.

Inspired by the procedural road generation techniques [PM01], our method incrementally grows the output road network at *unfinished* nodes (see Fig. 1). If starting the synthesis from an empty network, our algorithm initializes it with a single node marked as unfinished. Otherwise, we allow the user to interactively select unfinished nodes from existing nodes. At each step, we randomly pick an unfinished node v_d as a seed node. Unlike the procedural methods, we generate new segments incident to v_d by the guide of the input sample instead of production rules. Making an analogy to texture synthesis [EL99], we try to find in the sample a node v_s that has similar neighboring segments with v_d under a rotation by the angle θ. We call such a node pair and the corresponding transformation a *node match*, denoted by a triple $m = (v_d, v_s, \theta)$. We get a candidate set of node matches M by a fast estimation process (Section 3.1), and then choose a best one determined by the measure of neighborhood similarity (Section 3.2). Once the best node match has been found, we transform the segments that are incident to v_s, and use them as new proposed segments at v_d. The proposed segments are then modified or even rejected to satisfy various local constraints (Section 3.3). Finally, we remove the current seed node from the unfinished node set, and add new unfinished nodes that are introduced by the new segments. We repeat the above steps until no unfinished nodes remain.

3.1. Node matching

The core part of our algorithm is to find a portion of the sample best matching the neighborhood of current seed node v_d. Ideally, we should test all feasible rigid transformations of the sample, but the computational cost of this is impractical. In our method, we greatly simplify the problem by giving priority to local topology matching at nodes over neighborhood matching.

The task in the node matching stage is to find a candidate set of *node matches* for v_d, considering only the angles of edges which are incident to v_d. An eligible match (v_d, v_s, θ) should satisfy the following criteria: after a clockwise rotation with the angle θ, any edge incident to v_d can be well matched with an edge incident to the node v_s in the sample so that the angle between the two edges is less than a user specified threshold ε.

We iterate all nodes in the sample. For each pair of v_d and v_s, we use the following method to find eligible node matches. Let e_d^k and e_s^k be the k-th edge incident to v_d and v_s, respectively. We start from guessing a match by aligning e_d^0 to e_s^0 by a rotation with the angle of α. After the rotation, for each edge e_d^i, we find an edge e_s^j to minimize $\gamma_i = angle(e_d^i, e_s^j)$. If one of the angle difference $\gamma_i > \varepsilon$, we discard the current guess; otherwise, we have found an eligible match. We repeat above steps to find more eligible matches between v_d and v_s by aligning e_d^0 to other edges incident to v_s.

For each found eligible node match, we refine its rotation angle α to make angle differences between edges evenly distributed. This is achieved with an offset rotation of angle δ by minimizing $\sum(\delta + \gamma_i)^2$, which implies that, we should take $\delta = -\sum\gamma_i/n$. Thus, the final optimized rotation angle will be $\theta = \alpha + \delta$.

3.2. Neighborhood edges matching

Once a set of candidate node matches has been found, we need to find a best match by neighborhood edges matching. To measure the similarity between two neighborhood network, we employ the concept of line segment Hausdorff distance (LHD) originally proposed in [GL02] for face matching. Compared with Hausdorff distance measure that is defined over two point sets, LHD measure has the advantage that it takes into account structural and spatial information.

We first construct a subgraph $G_n \in G_d$ around v_d, where the distance from each segment to v_d is less than r, a main parameter in our method set by the user. For a match $m = (v_d, v_s, \theta)$, we define a transformed input $G_s(m)$ by rotating the sample graph G_s anticlockwise with the angle of θ, and translating it to align v_s with v_d. Finally, we can find the best node match that minimizes the following matching cost:

$$E(m) = \frac{1}{\sum w_i} \sum_{e_d^i \in G_n} w_i \cdot \min_{e_s^j \in G_s(m)} D(e_d^i, e_s^j), \tag{1}$$

where w_i is weight factor $1/dist(v_d, e_d^i)$, and $D(\cdot)$ is line segment Hausdorff distance.

The principle to choose the size of neighborhood r is similar to that in texture synthesis [EL99]. It should be the same scale as the biggest regular feature we want to capture.

3.3. Generating road segments at seed node

Given a best node match $m = (v_d, v_s, \theta)$, we can use it to guide the growth of new road segments originating from the unfinished node v_d. For each segment incident to v_s, a new segment e^p incident to v_d is proposed. Two parameters need to be determined: the length of the segment l, and the angle to the horizontal axis ϕ. We let $l_d = l_s$, and $\phi_d = \phi_s + \theta$.

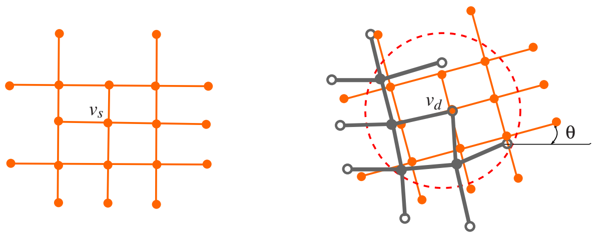

Figure 1: *Algorithm overview. For an unfinished node v_d in the existed road network (*middle*), we find in the input sample (*left*) a node v_s that has similar neighboring segments (inside red circle) under a rotation of θ. New road segments are then created at v_d by copying segments around v_s while adapted to local constraints,* e.g.*, snapped to nearby existed node (*right*).*

Before adding a proposed segment at v_d, we use procedural methods to check if it can be adapted to the local environment and existing road segments. As in [WMWG09], the check could include: (1) avoid generating new nodes in obstacles, such as water area, or out of the target region; (2) snap to nearby road segments; and (3) avoid cross intersection with nearby segments. In addition, we use the topology of nearby roads in the sample as constraints. One benefit of this is to avoid using an arbitrary value as snapping distance as in previous pure procedural methods. For example, we keep a dangling road copied from the sample still be a dangling road.

After the proposed segment is modified and finally accepted by all the local constraint checks, we insert it to the output road network. When the new road ends at a newly created node, we mark the node as *unfinished* unless (1) its corresponding edge e_s^p is an edge with valence 1 in the sample, or (2) it falls out of the legal area. Finally we remove v_d from the unfinished node list.

4. Results and applications

We have tested our algorithm for various applications. Our method produced good results for a wide range of street patterns. In the experiments we set $\varepsilon = 30^\circ$, and the neighborhood radius r three times the average road segment length.

Figure 2 demonstrates a synthesis in a river scene. Our method can avoid the river and procedurally generate bridges. We have built the algorithm in to a tool that allows refinement of a road network. In the example shown in Figure 3, we deleted a part of a grid road network, and filled in the gap with roads based on another pattern.

Our basic algorithm determines the rotation of samples by node matching. This works well for general road networks, especially those dominated by curved roads. However, we are not obliged to do this. For grid like patterns, we prefer disabling the rotation. We can also determine the rotation by user given functions, or curves to control global patterns (Figure 4). This kind of control can easily make roads follow curved boundaries such as river banks.

Our algorithm allows the use of more than one sample as input. Figure 5 demonstrates smooth transition between two street patterns. For each unfinished node, we use a user-defined function to select a sample for node matching and neighborhood matching.

We have also extended the algorithm to work with multiple scales of road network. We do this by using examples of road at different scales (e.g. minor road and major roads). Each segment in each network is labelled with its scale. When matching nodes we then need to match the level of the segment as well as the angle. Figure 6 shows an example with two levels of curved roads.

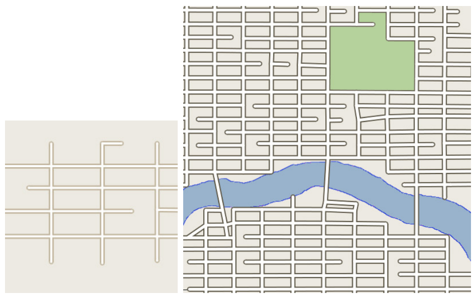

Figure 2: *Given a small patch of sample roads, a raster image defining obstacles (park and water), and a vector curve representing the river, we create a new road network that resembles the input. Procedural rules ensure avoiding roads in obstacles, and generating reasonable bridges.*

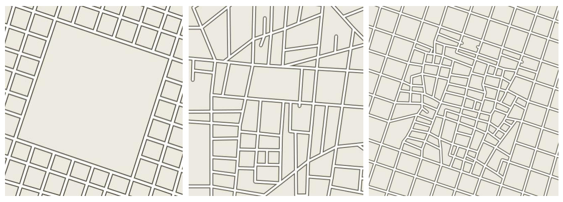

Figure 3: *We fill in a hole in a grid pattern (left) with real world irregular streets, and get a new, well connected street network (right) . The synthesis starts from the nodes on the boundary of the hole, which are manually marked as unfinished.*

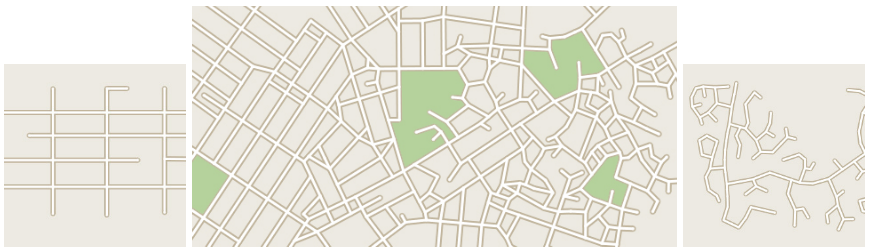

Figure 5: *Given a grid example (left) and a curve example (right), our method synthesizes a new road network gradually varying from grid pattern to curve pattern from left to right (middle). Given a normalized x coordinates of a node being processed, we choose the grid example when $f < x$, and choose the the curve example when $f > x$, where f is a uniformly distributed random number between* 0 *and* 1*.*

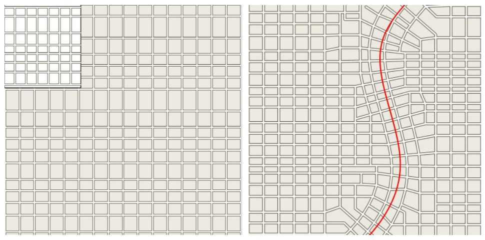

Figure 4: *Left: input network and a synthesized output without direction control. Right: A synthesized road network following a user given brush stroke.*

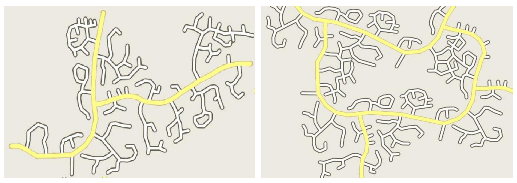

Figure 6: Left: *Input a real road network (South Woodham Ferrers, UK) with major and minor curved roads.* Right: *Starting from a random initial node, our method arrive a road network that resembles the input.*

5. Conclusion and future work

We have presented an example-based technique for automatically synthesizing road networks. Our method grows the output road network segment by segment as a procedural method, and it avoids using production rules or parameters that are non-trivial to configure. New road segments are generated according to a matched portion in the sample road network. The method can synthesize a wide range of road styles implicitly modeled by the examplars, including long curved roads, under various local constraints. Furthermore, we demonstrated its applications for complicated edit operations such as extending an existing road network, and connecting or blending two road networks. Last but not least, our method is simple to implement and intuitive for users.

A number of problems remain open. A straightforward extension is to synthesize roads in different levels with different samples. Thus we can not only capture local patterns but also global patterns. It would be interesting to incorporate the shape and size of street blocks into our neighborhood similarity measure. Finally, we would also like to explore a reverse problem: automatically extracting style examplars from real world data.

References

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

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

[EL99] EFROS A. A., LEUNG T. K.: Texture synthesis by non-parametric sampling. In *Proc. ICCV* (1999), pp. 1033–1038. 1, 2

[GL02] GAO Y., LEUNG M. K. H.: Line segment hausdorff distance on face matching. *Pattern Recognition 35*, 2 (2002), 361 – 371. 2

[GPGB11] GALIN E., PEYTAVIE A., GUÉRIN E., BENEŽ B.: Authoring hierarchical road networks. *Computer Graphics Forum 30*, 7 (2011), 2021–2030. 1

[HOCS02] HERTZMANN A., OLIVER N., CURLESS B., SEITZ S. M.: Curve analogies. In *Proc. Eurographics Workshop on Rendering* (2002), pp. 233–246. 1

[IMIM08] IJIRI T., MÊCH R., IGARASHI T., MILLER G.: An example-based procedural system for element arrangement. *Computer Graphics Forum 27*, 2 (2008), 429–436. 1

[Mer07] MERRELL P.: Example-based model synthesis. In *ACM-SIGGRAPH Symp. Interactive 3D Graphics (I3D)* (2007), pp. 105–112. 1

[PM01] PARISH Y. I. H., MÜLLER P.: Procedural modeling of cities. In *Proc. ACM SIGGRAPH* (2001), pp. 301–308. 1, 2

[VABW09] VANEGAS C. A., ALIAGA D. G., BENEŠ B., WADDELL P. A.: Interactive design of urban spaces using geometrical and behavioral modeling. *ACM Trans. Graph. 28* (2009), 111:1–111:10. 1

[Wad02] WADDELL P.: Modeling urban development for land use, transportation, and environmental planning. *Journal of the American Planning Association 68*, 3 (2002), 297–314. 1

[WMWG09] WEBER B., MUELLER P., WONKA P., GROSS M.: Interactive geometric simulation of 4D cities. *Computer Graphics Forum 28*, 2 (2009), 481–492. 3