Interactive Procedural Street Modeling

Guoning Chen [*] Gregory Esch [*] Peter Wonka [†] Pascal Müller [‡] Eugene Zhang[*]

[*]Oregon State University [†]Arizona State University [‡]Procedural Inc. / ETH Zürich

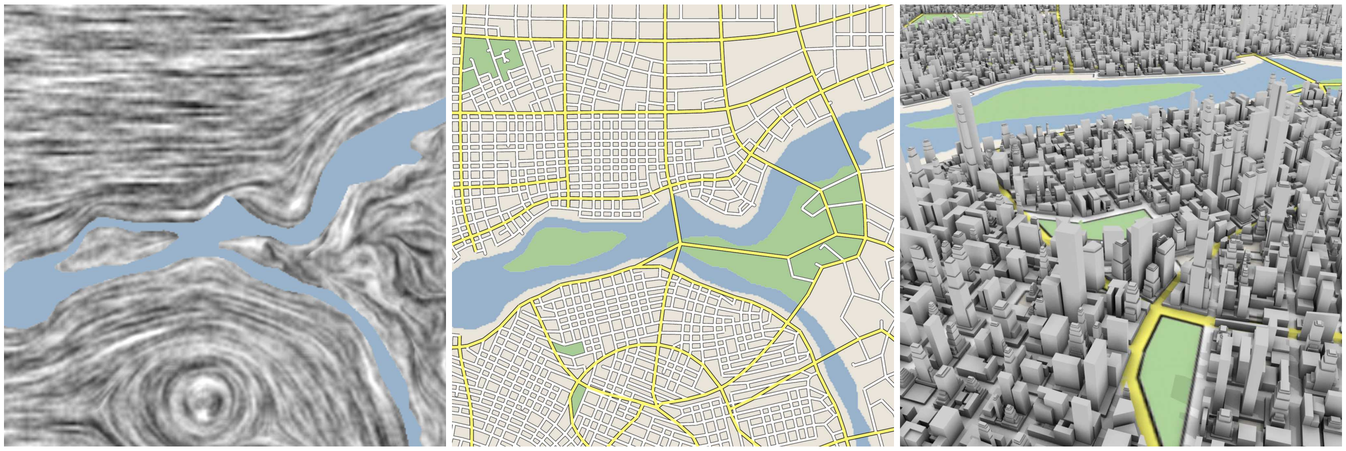


Figure 1: *This figure shows the three steps of our pipeline. The input water map is based on a stretch of the Benue River in Nigeria. Left: Starting from topographical water and park maps, the user designs a tensor field. Middle: The tensor field and further editing operations are used to generate a road network. Right: Three-dimensional geometry is created.*

Abstract

This paper addresses the problem of interactively modeling large street networks. We introduce an intuitive and flexible modeling framework in which a user can create a street network from scratch or modify an existing street network. This is achieved through designing an underlying tensor field and editing the graph representing the street network. The framework is intuitive because it uses tensor fields to guide the generation of a street network. The framework is flexible because it allows the user to combine various global and local modeling operations such as brush strokes, smoothing, constraints, noise and rotation fields. Our results will show street networks and three-dimensional urban geometry of high visual quality.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism I.6.3 [Simulation and Modeling]: Applications J.6 [Computer-Aided Engineering]: Computer-Aided Design (CAD)

Keywords: procedural modeling, street modeling, street networks, tensor fields, tensor field design

[*]{chengu|eschgr|zhange}@eecs.oregonstate.edu
[†]peter.wonka@asu.edu
[‡]pascal.mueller@procedural.com

1 Introduction

This paper presents a solution to efficiently model the street networks of large urban areas. The creation of compelling models is a crucial task in the entertainment industry, various training applications, and urban planning. However, modeling the details of large three-dimensional urban environments is very time consuming and can require several man years worth of labor. A powerful solution to large-scale urban modeling is the use of procedural techniques [Parish and Müller 2001; Wonka et al. 2003; Müller et al. 2006].

Parish and Müller [2001] are the first to note that the street network is the key to creating a large urban model, and they present a solution to model street networks based on L-systems [Prusinkiewicz and Lindenmayer 1991]. Starting from a single street segment they procedurally add more segments to grow a complete street network, similar to growing a tree [Prusinkiewicz et al. 2003]. While this algorithm creates a high quality solution, there remains a significant challenge: the method does not allow extensive user-control of the outcome to be easily integrated into a production environment. While the user can use a traditional modeling tool to move the vertices in the procedurally generated graph, the graph often requires a significant amount of editing in order to match user expectations. When this happens, the user will need to regenerate the complete environment but the results are *not* guaranteed to be more desirable.

To address this limitation of a purely procedural approach, we provide an alternative to street modeling that supports the integration of a wide variety of user inputs. The key idea of this paper is to use *tensor fields* to guide the generation of street networks.

An important aspect of street patterns is the existence of two dominant directions due to the need for efficient use of space. Interestingly, tensor fields give rise to two sets of *hyperstreamlines* (defined in Section 4): one follows the major eigenvector field, and the other the minor eigenvector field. These observations have inspired our approach in which interactive tensor field design techniques are used to guide the road network generation. This concept is illustrated in Figures 1 and 3. The user can interactively edit a street network by either modifying the underlying tensor field or by changing the graph representing the street network. This allows for efficient modeling because we can combine global and local modeling operations, constraints, and procedural methods.

Major Contributions of this paper are:

- *Insight:* We realize the connection between tensor fields and street graphs.
- *Pattern Analysis:* We analyze street patterns and derive suitable modeling operations on tensor fields and graphs.
- *Modeling Pipeline:* We arrange these modeling operations into a consistent framework (pipeline) that allows us to produce high quality results.
- *Technical Novelties:* We effectively integrate existing techniques for graph and tensor field editing into our framework. In addition, we make several new technical contributions to tensor field design and graph editing that include a novel brush interface, the use of rotation fields to modify tensor fields, hierarchical segmentation and editing of tensor fields, tensor field computation from boundaries, the ability to handle tensor field discontinuities, an improved hyperstreamline tracing algorithm, and a hybrid algorithm to modify graphs using tensor fields.

Paper Structure: After reviewing related work in Section 2, we provide a system overview in Section 3 and briefly review relevant background on tensor fields in Section 4. The two major parts of our system are tensor field generation (Section 5) and street graph generation (Section 6) We show results in Section 7 and discuss our system and possible future work in Section 8.

2 Related Work

In this paper we focus on the modeling of street networks which we augment with the generation of three-dimensional street geometry. To obtain a complete urban environment our system can be complemented with shape grammars [Wonka et al. 2003; Müller et al. 2006] for architecture. In the following, we review literature describing road construction and graph modeling algorithms.

Road Construction: Information about the geometry of road construction can be found in literature from civil engineering. We recommend the text [AASHTO 2004] as a comprehensive overview. Other useful resources are the Highway Capacity Manual [Board 2000] and the textbook by Mannering et al. [2005]. Street graphs present a fascinating modeling challenge, because they exhibit a mixture of fairly regular and organic patterns. Some more high level ideas are presented in other books related to *urban design* [Punter 1999; Alexander et al. 1977; Hillier 1996; Hillier 1998; Gingroz et al. 2004]. However, the most informative resources are internet based map services, as we try to match street patterns and do not attempt to simulate their formation.

Graph Generation: The most successful algorithm for street modeling to date is presented by Parish and Müller [2001], who extend L-systems to grow street segments like branches in a tree until they intersect an existing street segment. L-systems have been very successfully applied to plant modeling [Prusinkiewicz and Lindenmayer 1991; Prusinkiewicz et al. 1994; Měch and Prusinkiewicz 1996; Prusinkiewicz et al. 2001] and provide an inspiration for many graph layout problems.

We have also been inspired by approaches to model ice ray lattice design [Stiny 1977], mortar in brick layouts [Legakis et al. 2001], diffusion limited aggregation [Witten and Sander 1981], and cracks in Batik renderings [Wyvill et al. 2004]. However, the similarities of their appearances to street layouts are rather remote. A very interesting class of layout algorithms uses Voronoi Diagrams [Berg et al. 2000] of (randomly) distributed points. This idea is extended to generate textures [Worley 1996], mosaics [Hausner 2001], fracture patterns [Shirriff 1993; Mould 2005], and even some street patterns [Sun et al. 2002; Glass et al. 2006]. Jigsaw image mosaics [Kim and Pellacini 2002] are another interesting extension to layout arbitrary shapes. Another powerful graph generation algorithm is proposed in the context of modeling leaf venation patterns [Runions et al. 2005]. Recently, an interesting extension of graph layout appeared in the work of image-based maze construction [Xu and Kaplan 2007], in which a directional maze is constructed by computing two perpendicular families of streamlines according to a vector field derived from region boundaries and user specified curves. While some of these algorithms can match one specific street pattern that looks like mud cracks, we propose a system that allows a wider range of more frequent street layouts. Additionally, we focus on user control and editing operations.

3 Pipeline Overview

In this section, we give an overview of our modeling pipeline. The input to our system includes four maps loaded as images: 1) a binary valued water map W, 2) a binary valued park and forest map F, 3) a height map H, and 4) a population density map P. Each of these is a discrete function of $f : [-X,X] \times [-Y,Y] \rightarrow [0,1]$ defined on a grid (512×512 in our implementation). Our system employs a three-stage pipeline (Figure 2).

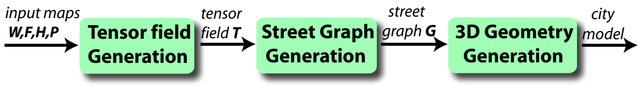


Figure 2: *The modeling pipeline.*

Stage One allows the user to produce a tensor field using a range of design operations, such as combining individual basis fields, computing tensor fields from boundaries, using a brush stroke interface, and rotating the field with noise. These tools allow the user to iteratively refine the design (Section 5). During editing, the user can manipulatea *tensor field* T and three *rotation fields* R_1, R_2, *and* R_3 which we use to rotate the eigenvector directions. The computational domain is a regular 2D grid D with the values of the aforementioned field stored at the vertices. Bilinear interpolation is used to obtain values inside the cells of D. These data structures are also the input to the next stage.

Stage Two is the street graph generation step. Streets are computed as hyperstreamlines (Section 4) of the tensor field. In Section 6 we explain how to generate the street network, edit it, and modify existing street networks using a combination of graph-based and tensor field editing. Street networks are modeled using a hierarchy: *major roads* and *minor roads*. Major roads are typically major business roads and local highways, and minor roads are usually residential and back roads. A street network is stored as a graph $G = (V,E)$ where V is a set of nodes and E is a set of edges. Nodes with three or more incident edges are *crossings*. Road attributes, such as road width, road type, pavement markings, and the type of lanes, are stored at nodes and edges.

Stage Three is a geometry generation module that creates three-dimensional street and building geometry to obtain a complete city. This is not the focus of our work; details can be found in [PROCEDURAL 2008].

To give an intuitive feeling for our system, we describe an example editing scenario (see Figure 3). First the user loads a water map (1) and places some tensor field design elements [Zhang et al. 2007]

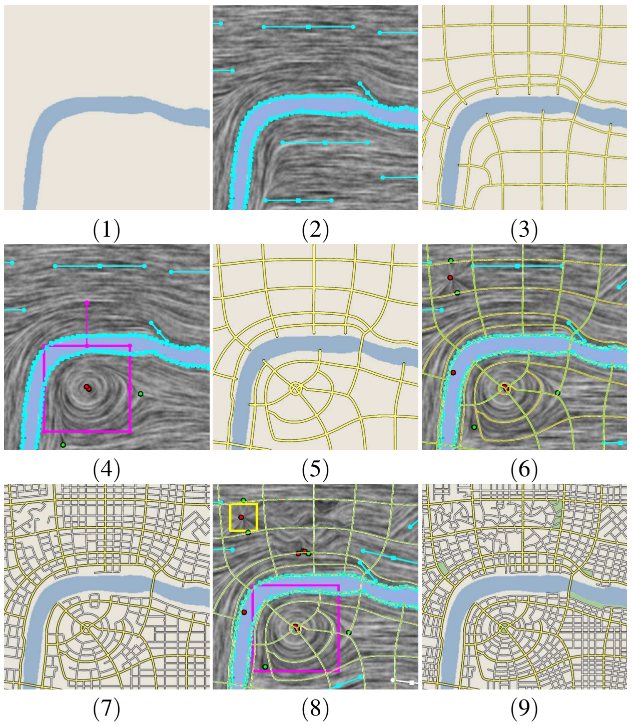


Figure 3: *An example sequence of modeling steps in our system.*

(2) that give rise to a major street network (3). Then the user refines the initial major road layout by placing a new tensor field design element inducing a radial structure in the tensor field (4) as well as the street graph (5). Using our segmentation algorithm, the user performs additional local tensor field modifications (6) and generate a minor road network (7). The user uses a rotation noise field to create irregular structures near the top (8) and produces the final result (9). The visualization of tensor fields shown in this paper is based on [van Wijk 2002; Zhang et al. 2007].

4 Tensor Field Background

In this paper, a tensor t refers to a 2×2 symmetric and traceless matrix, which is of the form $R\begin{pmatrix}\cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta\end{pmatrix}$ where $R \geq 0$ and $\theta \in [0, 2\pi)$. The major eigenvectors of t are $\{\lambda \begin{pmatrix}\cos\theta \\ \sin\theta\end{pmatrix} \mid \lambda \neq 0\}$, and the minor eigenvectors are $\{\lambda \begin{pmatrix}\cos(\theta+\frac{\pi}{2}) \\ \sin(\theta+\frac{\pi}{2})\end{pmatrix} \mid \lambda \neq 0\}$. The major and minor eigenvectors are perpendicular to each other in this setting.

A tensor field T is a continuous function that associates every point $\mathbf{p} = (\mathbf{x}, \mathbf{y}) \in \mathbb{R}^2$ with a tensor $T(\mathbf{p})$. $\mathbf{p}$ is said to be a *degenerate point* if $T(\mathbf{p}) = 0$, otherwise, it is *regular*. More theoretical details can be found in [Delmarcelle and Hesselink 1994].

Another important and relevant concept is the *hyperstreamline*, which describes curves that are tangent to an eigenvector field everywhere along their paths. A hyperstreamline is either *major* or *minor* depending on the type of the underlying eigenvector field. Note that the major and minor eigenvectors of a tensor field are *not* related to major and minor roads in a street network. For example, the tensor field corresponding to the major street network has its own major and minor hyperstreamlines. Hyperstreamlines have been used previously to visualize tensor fields [Wilson and Brannon 2005], generate pen-and-ink sketches of smooth surfaces [Hertzmann and Zorin 2000; Zhang et al. 2007], and remesh 3D geometry [Alliez et al. 2003; Marinov and Kobbelt 2004; Zhang et al. 2007].

5 Tensor Field Generation

In this section, we describe how to generate a tensor field in the domain using our system. The approach is to edit tensor fields by specifying constraints such as regular and radial patterns, brush strokes, topography information, and rotation fields. While we borrow some vector and tensor field design techniques such as the use of basis fields and field smoothing from previous work [Zhang et al. 2006; Zhang et al. 2007; Chen et al. 2007], we contribute the application of the idea to street network modeling and introduce a novel brush interface that facilitates the specification of user constraints, the use of rotation fields to relax the orthogonality in a tensor field network, the combination of noise and tensor field design, hierarchical segmentation and editing, automatic incorporation of water and height maps in the generation of a tensor field, and the introduction of discontinuities.

5.1 Generation of Basis Fields

The tensor field is generated based on user constraints (desirable patterns) and topography information (water and park boundaries, terrain height, etc). Near the city center, the user may wish to create a typical North-South and East-West pattern. In contrast, near the coastline, it is often natural to design the road network to follow the coastline. To provide sufficient flexibility in addressing these different and often competing needs, we seek a tensor field design framework that allows both global and local control.

We allow the user to specify desired street network patterns (e.g., regular, radial, etc) at needed locations. Each of the specified constraints is converted into a *basis* tensor field defined over the whole domain. These fields are then blended using decaying radial basis functions, which allows desired patterns to be maintained at specified locations. To respect features in the topography maps, we also generate basis tensor fields that respect the boundaries of features such as the boundaries of rivers and lakes. Such basis tensor fields can then be combined with user-specified basis fields, which will respect both user constraints and natural boundaries. Next, we provide examples on how to compute the basis tensor fields based on the input.

Grid: An important building block for most cities is the grid pattern. Parcels are generated by two orthogonal sets of parallel roads. A grid pattern can be defined by a regular element indicating the direction of the major eigenvector field. See Figure 4 for a tensor field guiding streets in a regular grid pattern. Given the direction (u_x, u_y) defined at a point $\mathbf{p}_0$ we can compute $l = \sqrt{u_x^2 + u_y^2}$ and $\theta = \arctan(\frac{u_y}{u_x})$ and define the following basis field (the constant direction field) [Zhang et al. 2007]:

$$T(\mathbf{p}) = l\begin{pmatrix}\cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta\end{pmatrix} \quad (1)$$

Radial: Radial patterns appear in different contexts. For example, radial patterns occur at the minor level to access residential homes (see Figure 5 right for a map section from Scottsdale, Arizona). Other examples are roads around important monuments, such as the Arc de Thriomphe in Paris. However, in these contexts the radial patterns are more noisy. To create a radial pattern at $\mathbf{p}_0 = (x_0, y_0)$ we can use a center design element whose major hyperstreamlines are circles and minor hyperstreamlines emanate from the center

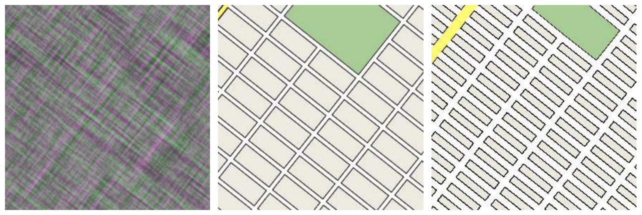


Figure 4: *Left: A tensor field encoding a regular grid. Middle: The resulting street network. Right: A regular pattern found in Brooklyn, New York.*

point. The basis field of a center element (radial pattern) has the following form [Zhang et al. 2007]:

$$T(\mathbf{p}) = \begin{pmatrix} y^2 - x^2 & -2xy \\ -2xy & -(y^2 - x^2) \end{pmatrix} \quad (2)$$

where $x = x_{\mathbf{p}} - x_0$ and $y = y_{\mathbf{p}} - y_0$.

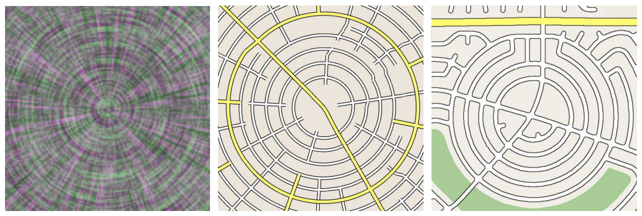


Figure 5: *A procedurally generated radial pattern (middle) and its tensor representation (left). The map shown in the right is a radial pattern found in Scottsdale, Arizona.*

Boundary Field: There are many examples of roads that are built at the boundaries of natural or man-made structures. Examples are roads next to the shoreline, such as California Highway One (see Figure 6). Other examples are roads at the boundaries of parks and roads surrounding population centers.

For example, we can extract boundary field from a water map. Since the water map we use is pixel-based, we can extract the boundary [Shapiro and Stockman 2001] of water in the map which can be either open (oceans, or rivers) or closed (lakes). From the boundary curves, we obtain a polyline approximation L, i.e., a curve consisting of a number of connected line segments.

To obtain a smooth tensor field that respects the boundary curve, we proceed as follows after obtaining the boundary polyline. For a line segment $\overline{AB} \in L$, we assign a regular element at point A determined by Equation 1 whose major eigenvector is $E_v = \overrightarrow{AB}$. The tensor field is then the combination of all of these regular elements (Section 5.2). Automatically constructing design elements from the boundaries provides the user more freedom in creating desirable patterns near the boundaries without losing the efficiency that comes with design elements. Figure 6 illustrates a street network (right) that was generated based on the coastline.

Heightfield: The natural elevation is an important constraint for most road construction. We observe that roads are often built by taking into account the gradient of the height field. To derive a tensor field from a heightfield $H(x,y)$, we compute the gradient $\nabla H = \begin{pmatrix} \partial H/\partial x, & \partial H/\partial y \end{pmatrix}$. We then use the tensor field $T(x,y) = R\begin{pmatrix} \cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta \end{pmatrix}$ whose *minor* eigenvector field matches the gradient of the heightfield everywhere, i.e. $\theta = \arctan(\frac{\partial H/\partial y}{\partial H/\partial x}) + \frac{\pi}{2}$ and $R = \sqrt{(\partial H/\partial x)^2 + (\partial H/\partial y)^2}$.

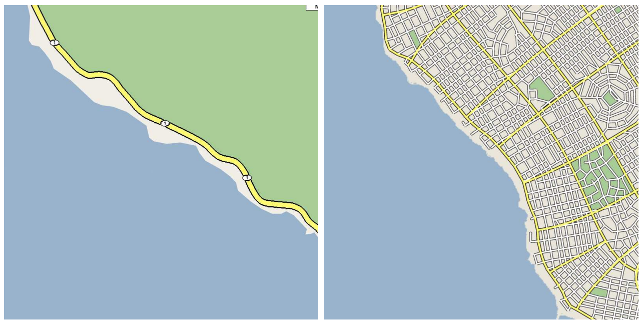


Figure 6: *Left: A map showing California Highway One. Right: A road network from a tensor field derived from the map boundary. Note a major road follows the coastline.*

5.2 Combination and Editing of Basis Fields

To obtain and modify a tensor field, we provide the following functionalities.

Combination of Basis Fields: The system allows the user to create and modify a tensor field by using *design elements*. A design element corresponds to a user-specified tensor field pattern, such as a grid or radial pattern, at a given location. Our implementation follows closely the tensor field design system of Zhang et al. [2007], in which every user specification is used to create a global basis tensor field. These basis fields are then summed using radial-basis functions (See Equation 3) such that the resulting tensor field satisfies the user specifications.

$$T(\mathbf{p}) = \sum_i e^{-d\|\mathbf{p}-\mathbf{p}_i\|^2} T_i(\mathbf{p}) \quad (3)$$

where d is a decay constant, $\mathbf{p}$ is a point in the computational domain, T_i is the basis tensor field corresponding to a design element, and $\mathbf{p_i}$ is the position of the design element. The user can also delete an existing design element or modify its location, orientation, and isotropic and anisotropic scales. Note that there are other ways of creating a directional field from user constraints, such as relaxation [Turk 2001; Wei and Levoy 2001; Fisher et al. 2007] and propagation [Praun et al. 2000]. We employ the idea of basis fields due to its simplicity and intuitiveness.

Tensor Field Smoothing: The user can reduce the complexity (i.e. the number of degenerate points) in the tensor field by performing component-wise Laplacian smoothing [Alliez et al. 2003; Marinov and Kobbelt 2004; Zhang et al. 2007]. Such an operation can be performed either globally or locally. In the latter case, the tensor values on the boundary of a local region serve as the constraints in relaxation.

Brush Interface: We also use the idea of a brush-based interface, in which the user produces tensor values by moving the mouse to form a curve or loop. Then a region is found to have a pre-defined distance to the curve [Sethian 1996]. Finally, the tensor values inside this region are computed by treating the user-specified curve as the constraint. The brush-based interface therefore allows a tensor field to be created locally instead of globally. More importantly, if desired, the tensor field can become discontinuous along the boundary of the region. An example operation is illustrated in Figure 7.

To implement the brush interface, we first extract the cell strip $\{S_1, \ldots S_n\}$ $(S_i \in D)$ that contains the polyline representing the brush curve. We then assign tensor values to the vertices of the cells in the strip according to the orientations of the brush stroke. For example, if a line segment $\overline{AB}$ is inside a cell S_i, we assign the tensor whose major eigenvector is $E_v = \overrightarrow{AB}$ to the four vertices of S_i. If a vertex is

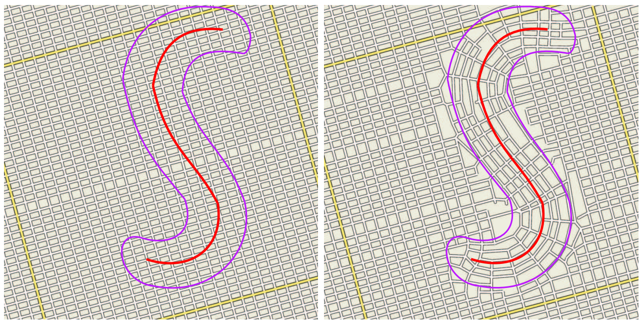


Figure 7: *This figure shows the use of the brush stroke interface to orient streets.*

shared by more than one cell in the strip, the average of the tensor values is used. A similar approach has been used to create periodic orbits in vector field design [Chen et al. 2007].

To extrapolate tensor values to other vertices in the region, we solve the following discrete Laplacian equations where the known tensor values serve as the boundary conditions:

$$T(v_i) = \sum_{j \in J_i} \omega_{ij} T(v_j) \tag{4}$$

in which $T(v)$ represents the tensor values at vertex v, J_i consists of the indexes of vertices that are adjacent to vertex v_i, and $\omega_{ij} = \frac{1}{N_i}$ where N_i is the number of vertices adjacent to v_i. Equation 4 is a sparse linear system, which we solve by using a conjugate gradient solver [Press et al. 1992].

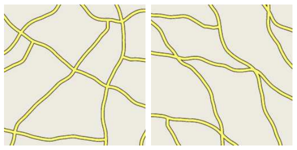
Discontinuities: To handle discontinuities across two neighboring regions A and B, our system provides two options. In the first approach, which we refer to as the *symmetric* case, the two regions have equal priority. Therefore, roads from the first region A will be clipped inside the second region minus the intersection region $B \setminus A$, and vice versa. In the second case, which is *asymmetric*, the end points of the roads from A inside the region of intersection $A \bigcap B$ are used as seed points to generate road in the second region B.

5.3 Modifying Tensor Fields Using Rotation Fields

In real street networks we observe various forms of irregularities that seem to have stemmed from slight distortions of regular or smooth patterns. Additionally, given a symmetric tensor field T, the major and minor hyperstreamlines always intersect at a right angle except at the degenerate points where they are not well-defined. While orthogonal intersections are dominant and preferred for construction, we also need to take into account non-orthogonal intersections. To model these phenomena we make use of three different scalar fields R_1, R_2 and R_3 that model rotations of the minor and major eigenvectors: 1) the first rotation field is used to rotate both major and the minor eigenvectors with R_1 degrees in opposite directions, i.e. the tensor value at (x,y) is altered such that the major and minor eigenvectors are rotated by an angle of $R_1(x,y)$ and $-R_1(x,y)$, respectively, where $R_1 \in [-\frac{\pi}{2}, \frac{\pi}{2}]$. 2) R_2 rotates the major eigenvector only, and 3) R_3 rotates the minor eigenvector. While in theory only two scalar fields are necessary, we have found that the use of three scalar fields provide additional intuition.

The modeling of rotation fields is treated as a height field design problem. We provide the user with two options to design such a height field. While we can also load a rotation field as image, we do not use this option in our results.

Morse Function Design: We borrow the idea from the fair Morse function design approach of Ni et al. [2004]. The user specifies the value of the rotation field at desired locations, and a Laplacian system similar to Equation 4 is solved. Notice in this case only one variable is being solved, which is the rotation field R. In the images at right, we compare two portions of street networks without using rotation field (left) and with a rotation field where $R_2 \in [0, 20°]$ (right). Note that after including the rotation field into the computation, we obtain a street network whose intersections do not form right angles. We are aware of the work on *asymmetric tensor field* analysis [Zheng and Pang 2005] which can be used to model non-orthogonal intersections in the street networks as well.



Noise: We use Perlin Noise [Perlin 1985] to generate a scalar field in the range of $[-\frac{\pi}{2}, \frac{\pi}{2}]$. We then use the obtained scalar field to rotate the tensor field and produce more organic-like street patterns (Figure 8).

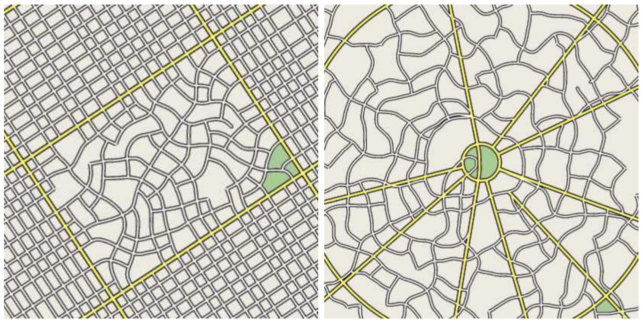


Figure 8: *This figure shows a regular major road grid (left) and a radial major road pattern (right) over slightly curved minor roads.*

6 Street Graph Generation

In this section, we describe how to generate a street network from a tensor field. We also describe how our system allows an existing street network to be modified directly as a graph or through local tensor field design.

6.1 Major Street Graph Generation from Tensor Fields

Our hyperstreamline placement algorithm is based on the work of [Jobard and Lefer 1997] for evenly spaced streamline placement. Given a second-order symmetric tensor field $T(x,y)$, we can produce two families of hyperstreamlines corresponding to the major and minor eigenvector fields, respectively. There are two difficulties unique to our application that cannot be handled properly by the original framework of Jobard and Lefer [1997]. First, tracing major and minor hyperstreamlines independently often leads to a disconnected street network. (shown in Figure 9 (left)). This is especially the case when a minor hyperstreamline does not intersect with any major hyperstreamlines. Second, important points such as those on the narrow passages are not reached by any hyperstreamline, causing undesirable street patterns. This often occurs on small protrusions near the the coastlines. We address these difficulties by introducing modified tracing and seeding algorithms described next.

Interleaving Tracing Scheme: To handle the first problem, we interleave the tracing of major and minor hyperstreamlines as follows. Starting from an initial seed, we trace a hyperstreamline along the major eigenvector field until it stops. We then compute a seed point on the obtained hyperstreamline at d_{sep}, a user specified distance for the control of hyperstreamline density, away from the previous seed. Next, we start from the obtained seed and trace a hyperstreamline following the minor eigenvector field until it stops. Similarly, we compute a seed on this hyperstreamline with d_{sep} away from the previous seed, which will be used to start the next iteration. The tracing algorithm stops when no more valid seed points are available. Figure 9 shows the difference between original method and our strategy.

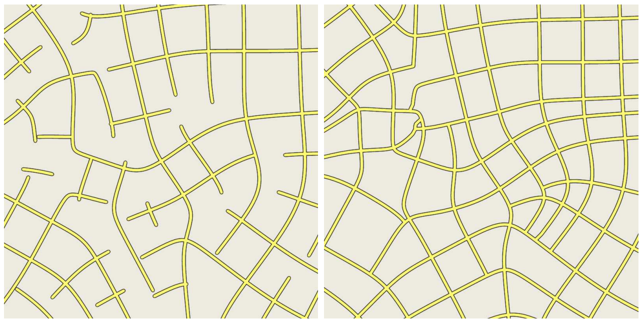


Figure 9: *This figure compares a street network in which major and minor hyperstreamlines are traced independently (left) and one using our approach (right). Notice that with our approach the street graph has fewer dangling edges.*

Single Hyperstreamline Tracing: An adaptive Runge-Kutta scheme [Cash and Karp 1990] is used to compute a hyperstreamline, which has been modified to handle tensor fields. Given a position of the current end point, we extract the direction in which the hyperstreamline grows by finding the major eigenvector value E_v at the end point. Let V_{pre} be the previous direction we use to compute the current point, to remove the sign ambiguity in eigenvector directions, we select the direction satisfying $E_v \cdot V_{pre} \leq 0$. The next integration point is then found using the numerical scheme. A hyperstreamline stops growing on the following stopping criteria: 1) it hits the boundary of the domain, 2) it runs into a degenerate point, 3) it returns to its origin which indicates a loop, 4) it exceeds a user-defined maximum length, or 5) it is too close to an existing hyperstreamline by violating d_{sep}. Additionally, we improve connectivity by continuing the tracing for a distance $d_{lookahead}$ to search an intersection with other hyperstreamline even when stopping criteria 4 or 5 is met. We also allow the tracing to cross relatively narrow water regions to form bridges depending on the required length of the bridge and the angle of the intersection with the coastline.

Seeding Scheme: The initial seed points for the tracing process can be either specified by the user or generated procedurally. The seed points are placed in a priority queue. The priority ω_{p_s} of a seed p_s can be computed using $\omega_{p_s} = e^{-d_b} + e^{-d_s} + e^{-d_p}$, where d_b is the distance from p_s to the closest water boundaries, d_s is its distance to the closest degenerate points of the field and d_p is its distance to the closest population centers. Additionally, we extract locations where narrow passages exist and place seeds there with higher priorities than seeds placed in elsewhere. The user can also assign a weight for a seed explicitly to force the tracing to start from a specific location. Next, we use an iterative process in which a hyperstreamline is generated based on the top element in the queue. During tracing of a hyperstreamline new seeds are added to the queue.

Generating Major Street Graph: The two families of hyperstreamlines can be used to generate a graph $G = (V, E)$. This is done by finding the intersection points between any pair of major and minor hyperstreamlines. V is the collection of intersection points, and E is the set of segments between two consecutive intersection points along a major or minor hyperstreamline. The graph G can be turned into a polygonal mesh by identifying the polygons in the graph. This is highly desirable when the user wishes to add buildings or other structures in between roads.

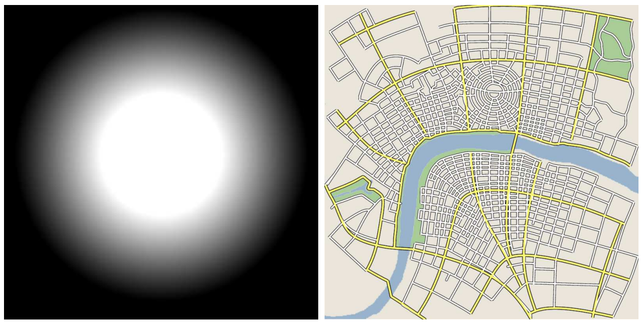


Figure 10: *This figure shows a density map (left) (white represents high population density value while black indicates lower density) and a generated density transition on the right.*

Transitions in Density: At city borders the road density decreases. Transitions in density are a phenomenon of the street graph and not the underlying tensor field. In our system, we use road density maps (or population density maps) to control d_{sep} in the road tracing algorithm described above. Figure 10 provides an illustrative example demonstrating how our system imitates the transition of density.

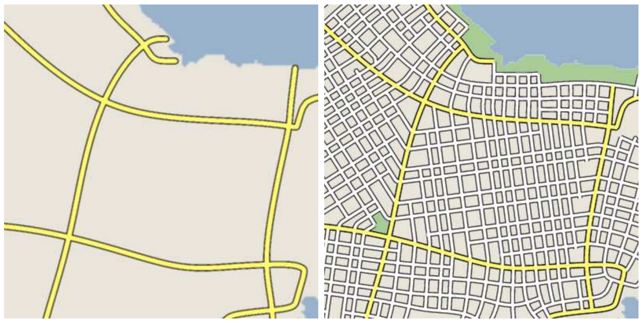


Figure 11: *This figure shows a minor road network (right) generated based on the major road network (left)*

6.2 Minor Street Graph Generation from Tensor Fields

Once the major street graph G_M has been constructed, it can be used to generate the minor street graph G_m. The process of generating G_m is similar to that of G_M with the following key difference. The edges in G_M and the boundaries of topographical features divide the domain into regions, inside each of which the user creates a continuous tensor field (see Figure 3 (6)). The tensor field can be discontinuous across region boundaries, i.e., major roads. This implies the tensor field used for minor road tracing is not necessarily the same as we use for major road tracing above. Figure 11 shows the minor road network generated based on the major road network. Note that minor roads do not necessarily follow the same directions as major roads. We point out that the idea of *flow tiles* proposed by Chenney [2004] for modeling a vector field can also be adopted to achieve the wealth of minor road patterns.

6.3 Street Graph Editing

Once a street network has been generated, it can be further modified using the following graph editing operations that we provide.

1. **Road Segment Manipulation:** The system enables the user to create and remove segments in the graph that was generated from the tensor field.

2. **Vertex Manipulation:** the user can move vertices in the street graph (by using drag and drop operations).

3. **Seed Point Creation:** the user can insert new streets by adding seed points at specified locations.

4. **Street Displacement:** the user can move a street by retracing a hyperstreamline from a nearby location.

5. **Layered Editing:** A seemingly random street may cut across an otherwise regular street network. The street can have a random beginning and end. See Figure 12 for an example. During implementation, we allow the user to indicate a random street by hand drawing it on top of the current street network. Our tool then converts the sketched road into a polyline divided by the underlying regular grid, which is used to search the intersections of the road with existing streets. The street network is updated accordingly.

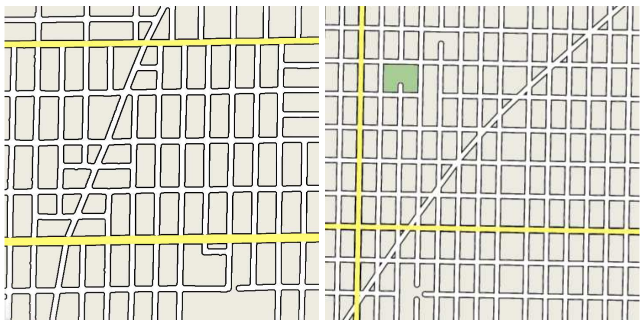


Figure 12: *Left: This map shows an example from Chicago, where a single street is laying over an otherwise regular north-south grid pattern. Right: A similar pattern is created using our system.*

6. **Graph Noise:** There are many examples where streets stop and later restart or connected slightly irregularly. Figure 13 shows some examples from Manhattan in New York City. The main idea is to model these patterns by deleting complete or partial street segments. We make use of stochastic sampling using Halton sequences [Pharr and Humphreys 2004] to create these patterns.

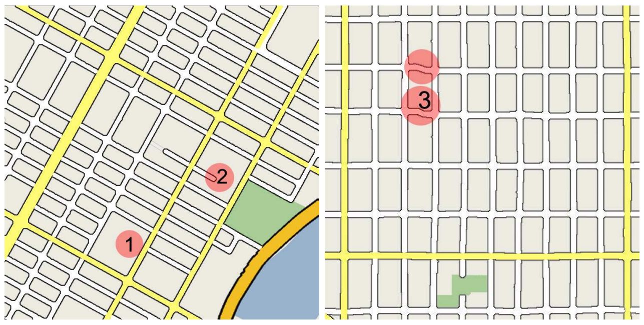


Figure 13: *This figure shows example maps from Manhattan, New York City. Left: Occasionally cells are merged together (1) or partially split by dead ends (2). Right: Slight irregularities can be seen in a regular grid (3).*

In addition, we provide the functionality that a segment in the street graph can be rotated as well. There are some instances where road networks share some similarities with fracture patterns. One example are major roads in rural Missouri (see Figure 14 left). In this case local topography dominates the road layout. We have some possibility to match these patterns with a tensor field and added noise. Figure 14 (right) shows a map generated using rotation on the graph (i.e. rotating the street segments). The rotation field is generated using Perlin noise discussed in Section 5.3.

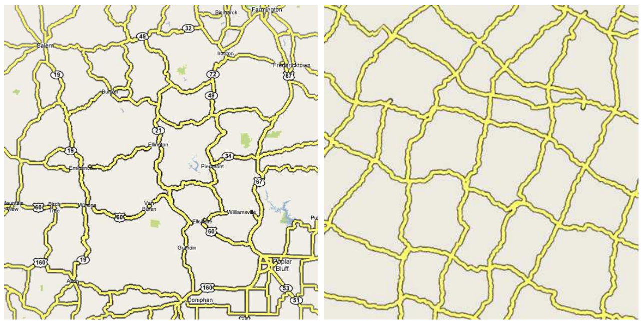


Figure 14: *This figure shows crack patterns in Missouri (left) and a procedurally generated patterns using our system (right).*

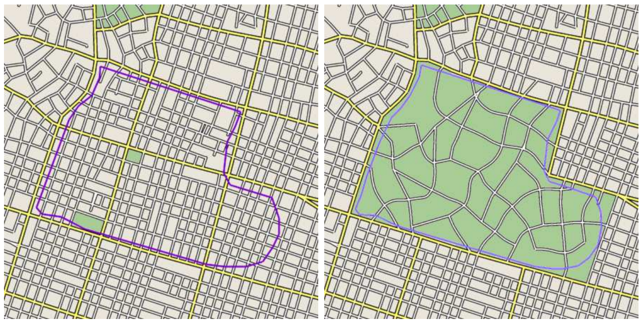


Figure 15: *This figure shows that a park can be inserted into an existing street network (left). Notice that the roads in the park region have a sparser density (right).*

6.4 Local Street Graph Editing using Tensor Fields

Our system can generate a street network by allowing the user to modify an existing street network such as those obtained from Google Maps. In this case, the input is a street graph $G = (V, E)$.

Our system allows the user to specify regions inside which the existing street network is erased and replaced with one that is created from a locally defined tensor field. Such an approach lends the power of tensor field design to graph editing. In our system, the user can explicitly specifies a region to modify or uses the brush interface that we discussed in Section 5 to obtain a region.

The portions of the original street network inside these regions will be erased, and resulting dangling edges in the remainder of the graph will be removed.

The original street network (outside the regions) and the user generated network (inside the regions) are connected by tracing boundary street segment forward until they hit the other network. Figure 15 illustrates this approach.

7 Results

To demonstrate the capabilities of our approach we show a number of street graphs generated using our system. Figure 16 shows a sec-

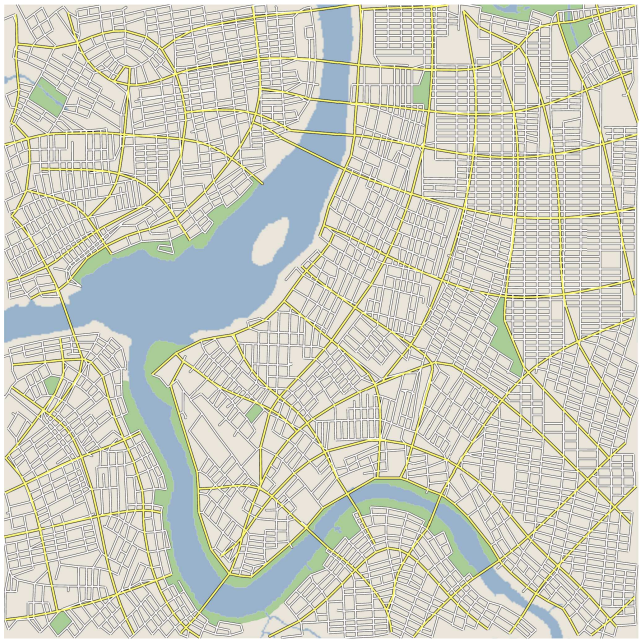


Figure 16: *A generated street graph for downtown Taipei.*

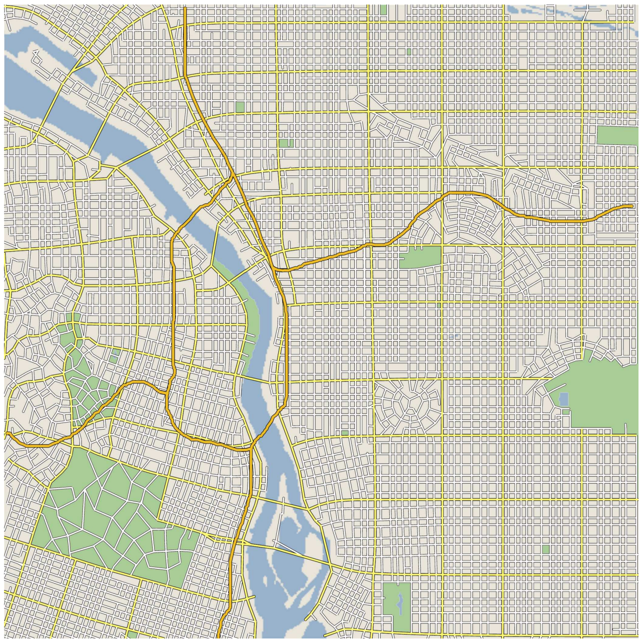


Figure 17: *This figure shows a generated street graph for downtown Portland, OR, USA. Note that the high ways (the orange roads) are hand drawn on top of the designed street network.*

tion of downtown Taipei which we have modeled. In Figure 17, a section of the Willamette River in Portland, OR is modeled. A road network for Manhattan is shown in Figure 18. Note that our goal is to generate maps inspired by real world maps, but not to exactly replicate the existing cities. In our experiments, a city with reasonable complexity can be modeled within five minutes, such as the fictional city in Figure 1, and the cities in Figures 16 and 17 took about five minutes for the main layout, but required an additional thirty to sixty minutes to fine tune the details and to experiment with different designs. The final images of three-dimensional geometry were created using RenderMan with ambient occlusion. See Figure 19 for four frames of a fly through shown in the accompanying video.

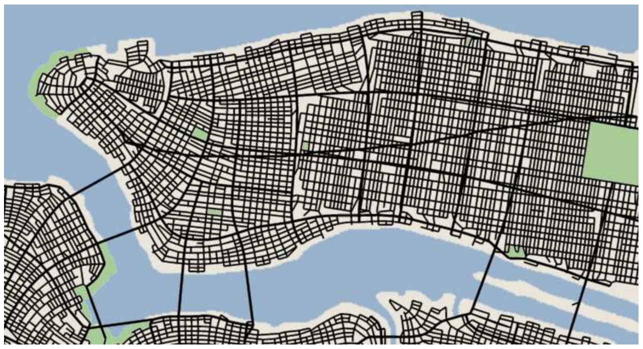


Figure 18: *A street graph for Manhattan, NY, USA generated using our tool.*

8 Discussion

In this paper we have presented a solution to the interactive modeling of street graphs. The main ideas of this paper are to (1) use tensor field design to guide the generation of a graph and (2) to integrate procedural modeling with interactive editing. These two concepts show promises to generate street networks, and we plan to extend this strategy to other graphics modeling problems. In the following, we discuss strengths and limitations of our approach and our contributions to computer graphics research.

Strengths: The inherent strengths of tensor fields include the possibility to model street patterns, which usually contain two preferred directions that are often mutually perpendicular. Furthermore, tensor field design allows the user to quickly generate an initial street layout which can be further modified at either the tensor field level or the graph level. This flexibility is unmatched by editing tools that only operate on the graph level, especially when creating the typical street patterns such as the regular East-West and North-South patterns.

Limitations: Currently, our system only assumes a single-level spatial resolution, which makes it difficult to modify the tensor field at significantly different scales. We plan to enhance our system by adding multi-scale editing capabilities.

Comparison to Related Work in Engineering: An interesting question is to compare our street modeling tool to street modeling in real urban environments. The most important distinguishing characteristic is scale. We are mainly concerned with efficient large-scale modeling of urban environments with high visual quality. In contrast, road construction in civil engineering is concerned with smaller project but pays significantly more attention to construction details. Examples of important factors are noise regulations, the turning paths of larger vehicles, ownership of land, legal regulations, and geological characteristics of the soil. Civil engineering software has some tools for intersection generation that would be interesting for our design system. However, the generation of three-dimensional geometric intersection details is a very complex subject that is beyond the scope of our research project.

Application: The main benefactors of this research are applications that require efficient content creation. Important examples are the entertainment industry with a strong demand to create content for computer games and movies. In recent years, modeling has evolved to be the most significant bottleneck in production. As a solution, procedural methods can be successful to drastically decrease modeling times. However, it has been our experience, that

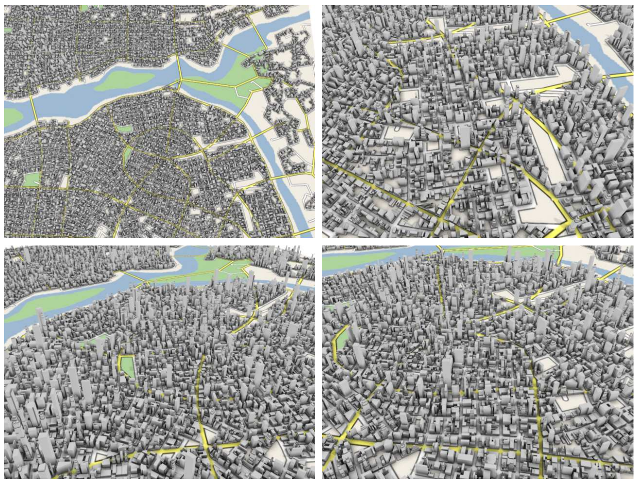


Figure 19: *Frames from a fly over of the virtual city shown in Figure 1.*

most companies are reluctant to adopt procedural methods if they do not have significant control to fine-tune the outcome. Therefore, the proposed modeling framework is an attempt to integrate procedural methods with high- and low-level user input to give the modelers the freedom they seek in designing their environments.

Future Work: This paper makes an important contribution to graph modeling problems in general. Even though several graph layouts appear to be fairly random, closer inspection will reveal a distinct pattern of two preferred directions. We believe that our methodology to use tensor fields to guide the generation of graphs can be very useful for related design problems, such as the modeling of cracks, fracture patterns, leaf venation patterns, bark, and ice crystals. We want to explore some of these potential connections as our future work. Furthermore, the two preferred directions of the street network induced by underlying tensor fields can be relaxed by resorting to latest work on *N*-way rotational symmetry fields [Palacios and Zhang 2007; Ray et al. to appear]. We are also interested to extend our work to include image-based editing techniques similar to [Aliaga et al. 2008].

Acknowledgments

We wish to thank Andreas Ulmer for helping with the rendering and modeling of the 3D cities. We also appreciate the discussions with William Brendel. We are grateful to Madhusudhanan Srinivasan and Patrick Neill for helping in preparing the videos. We would like to thank the reviewers for their valuable remarks and suggestions. Guoning Chen and Eugene Zhang are supported by NSF CCF-0546881. Gregory Esch is partially supported by Oregon Department of Transportation (ODOT) MBMSFY05-036. Peter Wonka is supported by NSF IIS-0612269, NSF CCF-0643822, and NGA grant HM1582-05-1-2004.

References

AASHTO. 2004. *A Policy on Geometric Design of Highways and Streets, 5th edition*. American Association of Highway and Transportation Officials.

ALEXANDER, C., ISHIKAWA, S., AND SILVERSTEIN, M. 1977. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, New York.

ALIAGA, D. G., BENEŠ, B., VANEGAS, C. A., AND ANDRYSCO, N. 2008. Interactive reconfiguration of urban layouts. *IEEE Computer Graphics and Applications 28*, 3, 38–47.

ALLIEZ, P., COHEN-STEINER, D., DEVILLERS, O., LÉVY, B., AND DESBRUN, M. 2003. Anisotropic polygonal remeshing. *ACM Transactions on Graphics 22*, 3, 485–493.

BERG, M. D., KREVELD, M. V., OVERMARS, M., AND SCHWARZKOPF, O. 2000. *Computational Geometry*. Springer-Verlag.

BOARD, T. R. 2000. *Highway Capacity Manual; U.S. Customary Version*. Transportation Research Board.

CASH, J. R., AND KARP, A. H. 1990. A variable order Runge-Kutta method for initial value problems with rapidly varying right-hand sides. *ACM Transactions on Mathematical Software 16*, 201–222.

CHEN, G., MISCHAIKOW, K., LARAMEE, R. S., PILARCZYK, P., AND ZHANG, E. 2007. Vector field editing and periodic orbit extraction using morse decomposition. *IEEE Transaction on Visualization and Computer Graphics 13*, 1, 769–785.

CHENNEY, S. 2004. Flow tiles. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 233–242.

DELMARCELLE, T., AND HESSELINK, L. 1994. The Topology of Symmetric, Second-Order Tensor Fields. In *Proceedings IEEE Visualization '94*, 140–147.

FISHER, M., SCHRÖDER, P., DESBRUN, M., AND HOPPE, H. 2007. Design of tangent vector fields. ACM, New York, NY, USA, vol. 26, 56.

GINGROZ, R., ROBINSON, R., CARTER, D. K., JR., B. J. L., AND OSTERGAARD, P. 2004. *The Architectural Pattern Book: A Tool for Building Great Neighborhoods*. W. W. Norton & Company.

GLASS, K. R., MORKEL, C., AND BANGAY, S. D. 2006. Duplicating road patterns in south african informal settlements using procedural techniques. In *Afrigaph '06: Proceedings of the 4th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, ACM Press, 161–169.

HAUSNER, A. 2001. Simulating decorative mosaics. In *SIGGRAPH Proceedings*, 573–580.

HERTZMANN, A., AND ZORIN, D. 2000. Illustrating smooth surfaces. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH 2000)* (Aug.), 517–526.

HILLIER, B. 1996. Cities as movement economies. In *Urban Design International*, 41–60.

HILLIER, B., 1998. The common language of space: A way of looking at the social, economic and environmental functioning of cities on a common basis.

JOBARD, B., AND LEFER, W. 1997. Creating evenly-spaced streamlines of arbitrary density. *Proc. Eighth Eurographics Workshop on Visualization in Scientific Computing*, 45–55.

KIM, J., AND PELLACINI, F. 2002. Jigsaw image mosaics. In *SIGGRAPH 2002 Conference Proceedings*, ACM Press/ACM SIGGRAPH, J. Hughes, Ed., Annual Conference Series, 657–664.

LEGAKIS, J., DORSEY, J., AND GORTLER, S. J. 2001. Feature-based cellular texturing for architectural models. In *Proceedings of ACM SIGGRAPH 2001*, ACM Press, E. Fiume, Ed., 309–316.

MANNERING, F. L., KILARESKI, W. P., AND WASHBURN, S. S. 2005. *Principles of Highway Engineering and Traffic Analysis*. John Wiley & Sons.

MARINOV, M., AND KOBBELT, L. 2004. Direct anisotropic quad-dominant remeshing. *Computer Graphics and Applications, 12th Pacific Conference on (PG'04)*, 207–216.

MĚCH, R., AND PRUSINKIEWICZ, P. 1996. Visual models of plants interacting with their environment. In *Proceedings of ACM SIGGRAPH 96*, ACM Press, H. Rushmeier, Ed., 397–410.

MOULD, D. 2005. Image-guided fracture. In *GI '05: Proceedings of the 2005 conference on Graphics interface*, Canadian Human-Computer Communications Society, 219–226.

MÜLLER, P., WONKA, P., HAEGLER, S., ULMER, A., AND VAN GOOL, L. 2006. Procedural Modeling of Buildings. In *Proceedings of ACM SIGGRAPH 2006 / ACM Transactions on Graphics*.

NI, X., GARLAND, M., AND HART, J. C. 2004. Fair morse functions for extracting the topological structure of a surface mesh. *ACM Transactions on Graphics (SIGGRAPH 2004) 23*, 3 (Aug.), 613–622.

PALACIOS, J., AND ZHANG, E. 2007. Rotational symmetry field design on surfaces. *ACM Trans. Graph. 26*, 3, 55.

PARISH, Y. I. H., AND MÜLLER, P. 2001. Procedural modeling of cities. In *Proceedings of ACM SIGGRAPH 2001*, ACM Press, E. Fiume, Ed., 301–308.

PERLIN, K. 1985. An image synthesizer. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, 287–296.

PHARR, M., AND HUMPHREYS, G. 2004. *Physically Based Rendering : From Theory to Implementation*. Morgan Kaufmann.

PRAUN, E., FINKELSTEIN, A., AND HOPPE, H. 2000. Lapped textures. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH 2000)* (Aug.), 465–470.

PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. 1992. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA.

PROCEDURAL, 2008. CityEngine. http://www.procedural.com.

PRUSINKIEWICZ, P., AND LINDENMAYER, A. 1991. *The Algorithmic Beauty of Plants*. Springer Verlag.

PRUSINKIEWICZ, P., JAMES, M., AND MĚCH, R. 1994. Synthetic topiary. In *Proceedings of ACM SIGGRAPH 94*, ACM Press, A. Glassner, Ed., 351–358.

PRUSINKIEWICZ, P., MÜNDERMANN, P., KARWOWSKI, R., AND LANE, B. 2001. The use of positional information in the modeling of plants. In *Proceedings of ACM SIGGRAPH 2001*, ACM Press, E. Fiume, Ed., 289–300.

PRUSINKIEWICZ, P., FEDERL, P., KARWOWSKI, R., AND MECH, R. 2003. L-systems and beyond. *ACM SIGGRAPH 2003 Course Notes* (Aug.).

PUNTER, J. 1999. *Design Guidelines in American Cities*. Liverpool University Press.

RAY, N., VALLET, B., LI, W.-C., AND LEVY, B. to appear. N-symmetry direction field design. *ACM Transactions on Graphics*.

RUNIONS, A., FUHRER, M., LANE, B., FEDERL, P., ROLLAND-LAGAN, A.-G., AND PRUSINKIEWICZ, P. 2005. Modeling and visualization of leaf venation patterns. *ACM Transactions on Graphics 24*, 3, 702–711.

SETHIAN, J. 1996. A fast marching level set method for monotonically advancing fronts. In *Proc. Nat. Acad. Sci.*, vol. 93, 1591–1595.

SHAPIRO, L. G., AND STOCKMAN, G. C. 2001. *Computer Vision*. Prentice Hall.

SHIRRIFF, K. 1993. Generating fractals from Voronoi diagrams. *Computers and Graphics 17*, 2, 165–167.

STINY, G. 1977. Ice-ray: a note on chinese lattice designs. *Environment and Planning B 4*, 89–98.

SUN, J., YU, X., BACIU, G., AND GREEN, M. 2002. Template-based generation of road networks for virtual city modeling. In *VRST '02: Proceedings of the ACM symposium on Virtual reality software and technology*, ACM Press, New York, NY, USA, 33–40.

TURK, G. 2001. Texture synthesis on surfaces. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH 2001)*, 347–354.

VAN WIJK, J. J. 2002. Image based flow visualization. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 745–754.

WEI, L. Y., AND LEVOY, M. 2001. Texture synthesis over arbitrary manifold surfaces. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH 2001)*, 355–360.

WILSON, A., AND BRANNON, R. 2005. Exploring 2d tensor fields using stress nets. *IEEE Visualization Proceeding*, 11–18.

WITTEN, T. A., AND SANDER, L. M. 1981. Diffusion-limited aggregation, a kinetic critical phenomenon. *Phys. Rev. Lett. 47*, 1400–1403.

WONKA, P., WIMMER, M., SILLION, F., AND RIBARSKY, W. 2003. Instant architecture. *ACM Transactions on Graphics 22*, 3, 669–677.

WORLEY, S. 1996. A cellular texture basis function. In *Proceedings of ACM SIGGRAPH 96*, ACM Press, New York, NY, USA, 291–294.

WYVILL, B., VAN OVERVELD, K., AND CARPENDALE, S. 2004. Creating Cracks for Batik Renderings. *NPAR 2004 Proceedings of the third international symposium on Non-photorealistic animation and rendering*, 61–70.

XU, J., AND KAPLAN, C. S. 2007. Image-guided maze construction. *ACM Trans. Graph. 26*, 3, 29.

ZHANG, E., MISCHAIKOW, K., AND TURK, G. 2006. Vector field design on surfaces. *ACM Transactions on Graphics 25*, 4, 1294–1326.

ZHANG, E., HAYS, J., AND TURK, G. 2007. Interactive tensor field design and visualization on surfaces. *IEEE Transactions on Visualization and Computer Graphics 13*, 1, 94–107.

ZHENG, X., AND PANG, A. 2005. 2d asymmetric tensor analysis. In *IEEE Visualization*, 1–8.