Interactive Procedural Street Modeling

Greg Esch* Peter Wonka † Pascal Müller‡ Eugene Zhang*

*Oregon State University †Arizona State University ‡ETH Zürich

Abstract

This paper addresses the problem of interactively modeling large street networks. We introduce a modeling framework that uses tensor fields to guide the generation of a street graph. A user can interactively edit a street graph by either modifying the underlying tensor field or by changing the graph directly. This framework allows to combine high- and low-level modeling operations, constraints, and procedural descriptions.

CR Categories: F.4.2 [Mathematical Logic and Formal Languages]: Grammars and Other Rewriting Systems 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

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 [Müller et al. 2006; Wonka et al. 2003; Parish and Müller 2001].

Parish and Müller [2001] were the first to note that the street network is the key to create a large urban model, and they presented a solution to model street networks based on L-systems. Starting from a single street segment they procedurally add further segments to grow a complete street network, similar to growing a tree [Prusinkiewicz et al. 2003]. While this algorithm created a high quality solution, there is a significant remaining challenge: the method does not allow extensive user-control of the outcome to be easily integrated into a production environment. After a street network is created, the user can use a traditional modeling tool to move the vertices in the graph. However, often the procedurally generated graph 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 it is *not* guaranteed that more desirable results can be generated.

To address this limitation of a purely procedural approach, we provide a rather different alternative to street modeling that allows to integrate a wide variety of user input. The key idea of this paper is to use tensor fields to guide the generation of street graphs. A user can interactively edit a street graph by either modifying the underlying tensor field or by changing the graph directly. This allows for efficient modeling, because we can combine high-level and low-level modeling operations, constraints, and procedural methods. The major contributions of this paper are as follows:

- We are the first to introduce a procedural approach to model urban street networks that combines interactive user-guided editing operations and procedural methods. We will identify important patterns in street networks and important editing operations that enable the user to model these patterns.
- We are introducing a new methodology to graph modeling in general. The idea of tensor-guided graph modeling together with the tight integration of interactive editing and procedural modeling has not been explored previously in related modeling problems, such as modeling of bark, cracks, fracture, or trees.

2 Related Work

Our approach to procedural urban modeling follows the outline presented by Parish and Müller [2001], who first model a street network, then parcels, and finally three-dimensional geometry. We focus on the modeling of street networks including the generation of three-dimensional geometry, and our approach can be complemented with shape grammars [Müller et al. 2006; Wonka et al. 2003] for buildings to obtain a complete modeling system for urban environments. 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 the civil engineering literature. 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. [Mannering et al. 2005]. Street graphs present a fascinating modeling challenge, because they exhibit a mixture of fairly regular and organic patterns.

Graph Generation: The most successful algorithm for street modeling to date was presented by Parish and Müller [2001], who extend L-systems and 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 were also 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 Bantik renderings [Wyvill et al. 2004]. However, the similarities of their appearances to street layouts were rather remote. A very interesting class of layout algorithms uses Voronoi Diagrams [Berg et al. 2000] of (randomly) distributed points. This idea was extended to generate textures [Worley 1996], mosaics [Hausner 2001], fracture patterns [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. While some of these algorithms can match one specific street pattern that looks like mud cracks, we propose a system that allows a much wider range and more frequent street layouts. Additionally, we allow for a much wider range of editing operations.

*{eschgr|zhange}@eecs.oregonstate.edu, Corvallis, OR 97331

†peter.wonka@asu.edu, Tempe, AZ 85287

‡pmueller@vision.ee.ethz.ch, Switzerland

We also briefly considered a modeling system based on physical simulations. Simulation can successfully model reaction-diffusion [Turk 1991; Witkin and Kass 1991] and various methods for fracture formation on surfaces [Hirota et al. 1998; O'Brien and Hodgins 1999; Lefebvre and Neyret 2002; Federl and Prusinkiewicz 2004; Smith et al. 2001; Neff and Fiume 1999]. We chose not to work with physical simulation, because the incorporation of editing operations is traditionally very difficult and it is also unclear what type of extensions are needed to generate a wider range of street pattern.

Another powerful graph generation algorithm was proposed in the context of modeling leaf venation patterns [Runions et al. 2005]. This algorithm grows leaf veins towards Auxin sources similar to how streets in [Parish and Müller 2001] grow towards population centers.

3 Overview

In this section, we explain the major idea of the paper, the structure of the paper, and definitions and concepts important for the understanding of later sections.

Street Networks: We model a hierarchy of streets: *major roads* and *minor roads*. Major roads are typically major business roads and local highways, and minor roads are usually residential and back roads . We store a street network as a graph $G = (V,E)$ where V are a set of nodes and E are a set of edges. Nodes with three or more incident edges are *crossings*. We store attributes with nodes and edges, such as road width, road type, pavement markings, and the type of lanes. One of the most fascinating aspects about street graphs is the wide variety of different patterns. We need a modeling methodology that can handle these patterns with a wide range of regularity. In section 4, we will highlight some of the challenges.

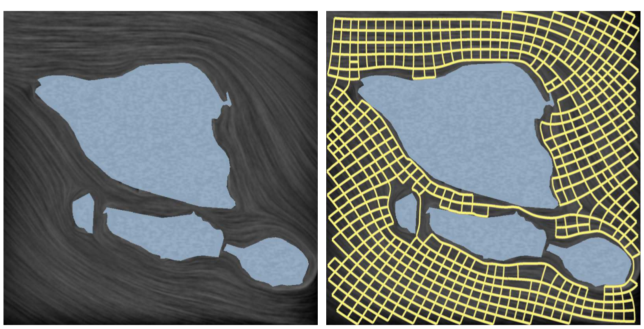

Figure 1: This figure illustrates how a designed tensor field (left) can guide the generation of a street graph (right).

Street Networks as Streamlines of Tensor Fields: A dominant aspect of street patterns is the existence of two dominant directions. This observation inspired us to use tensor fields to guide the street placement. Tensor fields give rise to two sets of tensor lines: One follows the major eigenvector field, and the other the minor eigenvector field. Our solution to street modeling is to interactively create a tensor field that guides the road network generation. This concept is illustrated in figure 1. Tensor lines have been used previously to visualize tensor fields [Wilson and Brannon 2005], to generate pen-and-ink sketching of smooth surfaces [Hertzmann and Zorin 2000; Zhang et al. 2007], and to remesh 3D geometry [Alliez et al. 2003; Marinov and Kobbelt 2004; Zhang et al. 2007].

Workflow: Our system employs a three-stage pipeline. First, terrain and population density maps are either procedurally generated, painted, or extracted from real data sets. Next, the user creates a tensor field on the the terrain using the editing tools provided by our system. At the end of this step, nicely-spaced major and minor tensor lines are generated according to the tensor field. These lines form a graph. Finally, the user can modify the graph. This graph can then be used as input to a procedural modeling tool to create three-dimensional geometry for roads, buildings, and vegetation.

Paper Overview: To describe our system, we will first demonstrate how suitable tensor fields can be found to match important street patterns (section 4). Second, we will explain in section 5 what operations are important to interactively edit and combine tensor fields. Third, section 6 explains how we generate road networks from a tensor field and additional graph-based processing operations, and section 7 explains how three-dimensional geometry is generated from the road network. We show some renderings in section 8 and discuss our contribution, applications, and comparison to related work in section 9. Conculsions are given in section 10.

Tensor Field Definitions: 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, and together they form a cross.

A tensor field T is a continuous function that associates every point $\mathbf{p} \in \mathbb{R}^2$ with a tensor $T(\mathbf{p})$. $\mathbf{p}$ is said to be a *degenerate point* if $T(\mathbf{p}) = 0$. Otherwise, it is *regular*. A degenerate point $\mathbf{p}$ is *isolated* if there exists a compact neighborhood N of $\mathbf{p}$ such that $\mathbf{p}$ is the only singularity in the interior of N and there are no singularities on the boundary of N. An isolated singularity can be characterized using its *tensor index*, which is defined in terms of the *winding number* of the *Gauss map*. Another important and relevant concept is *tensor lines*, which describe curves that are tangent to an eigenvector field everywhere along its path. A tensor line is either *major* or *minor* depending on the type of the underlying eigenvector field. Please note that the major and minor eigenvectors of a tensor field are not related to major and minor roads.

4 Street Patterns and Tensor Fields

In this section we show important concepts of street networks and show how to encode these concepts as tensor fields. We build upon classifications made by Parish and Mueller [2001], but our methodology to encode street patterns is totally different to allow for interactive editing. In the following we will first explain how to create some idealized elements and then give our solution to create more variations in the street pattern.

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 tensor field element defining the direction of the major eigenvector. See figure 2 for a tensor field guiding streets in a regular grid pattern. Given the direction (v_x, v_y) defined at $\mathbf{p}_0$ we can compute $l = \sqrt{v_x^2 + v_y^2}$ and $\theta = \arctan(\frac{v_y}{v_x})$ and define the following basis field:

$$T(\mathbf{p}) = e^{-d\|\mathbf{p}-\mathbf{p}_0\|^2} l \begin{pmatrix}\cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta\end{pmatrix} \quad (1)$$

where d is a decay constant.

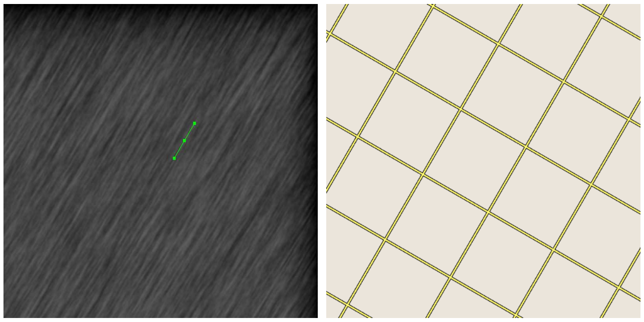

Figure 2: Left: A tensor field encoding a regular grid. Right: The resulting street network.

Radial: Radial pattern appear in different contexts. For example, radial patterns occur at the minor level to access residential homes (see figure 3 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 pattern is more noisy. To create a radial pattern at $\mathbf{p}_0 = (x_0, y_0)$ we can use a center design element, whose major tensor lines are circles and minor tensor lines emanate from the center point. The basis field of a center element (radial pattern) has the following form:

$$T(\mathbf{p}) = e^{-d\|\mathbf{p}-\mathbf{p}_0\|^2} \begin{pmatrix} y^2 - x^2 & -2xy \\ -2xy & -(y^2 - x^2) \end{pmatrix} \quad (2)$$

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

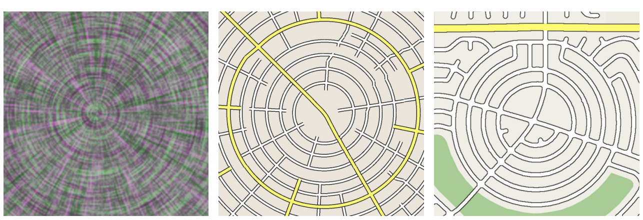

Figure 3: A procedurally generated radial pattern (middle) and its tensor representation (left). The image shown in the right is a radial pattern found in Scottsdale, Arizona.

Boundary: There are many examples of roads that are built at the boundary of natural or man-made structures. Example are roads next to the shoreline, such as the highway one in California (see figure 4). Other examples are roads at the boundary of parks and roads surrounding population centers. To define a tensor field for a boundary pattern we proceed as follows.

The boundary of a region is represented as a polyline, which consists of a number of connected line segments. Note the boundary can be either open (coastline) or closed (boundary of a park). We first extract the triangle strip $\{T_1, \ldots T_n\}$ that contains the polyline. We then assign vector values to the vertices of the triangles in the strip according to the orientations of the polyline inside the triangles. For example, if a line segment $\overline{AB}$ is inside a triangle T_i, we assign the vector $v = \overrightarrow{AB}$ to the three vertices of T_i. If a vertex is shared by more than one triangles in the same strip, the average is used. The vector values at these vertices will then be treated as part of the boundary conditions for field smoothing in order to obtain smooth transitions into unspecified regions.

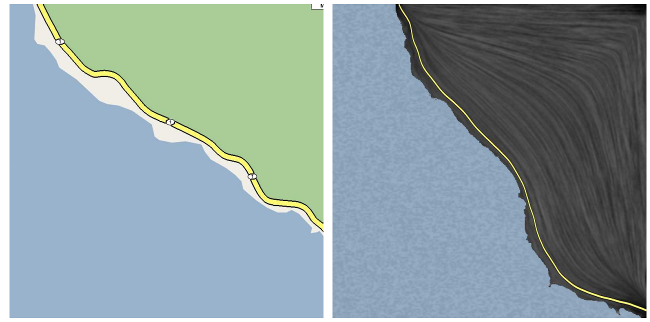

Figure 4: Left: A map of the highway one in California. Right: A tensor field and a road generated by the coast line.

Heightfield: The natural elevation is an important constraint for most road construction. We can observe that roads are built according to 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 find 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 $R = \sqrt{(\partial H/\partial x)^2 + (\partial H/\partial y)^2}$.

Transitions in Density: At city borders the road density decreases. For example, figure 5 left shows an example from the north of Denver. If we look at horizontal cross sections of the map and count the major roads (yellow), we can see a gradual transition from a perfect square mile raster to only one road at the top. Transitions in density are a phenomenon of the street graph and not the underlying tensor field. We use road density maps (or population density maps) to control the road tracing algorithm described in section 6. See figure 5 right for a result from our modeling system.

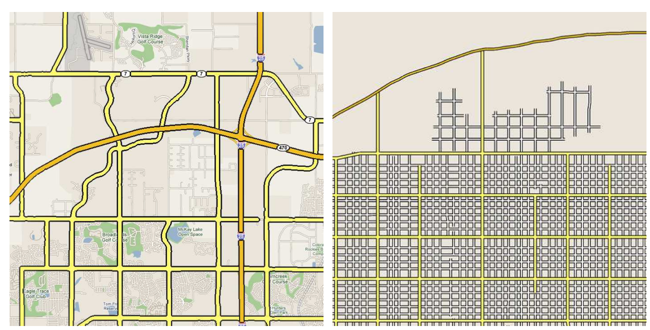

Figure 5: This figure shows transitions in street density in Denver (left) and a generated density transition on the right.

Irregularities: The previously described tensor fields are all smooth and would give rise to perfectly regular structures. In real street networks we can observe various forms of irregularities. We will briefly describe how to classify these irregularities and give a strategy to implement them. In our modeling framework, some of the irregularities are implemented as distortions of a tensor field, and other irregularities are better implemented on the graph level:

- *Deleted Street Segments*: There are many examples where a street stops and later restarts. Figure 6 left shows an example from Manhattan in New York City. The deleted street segments result in merged adjacent parcels in the regular grid or dead ends if street segments are only deleted partially. The

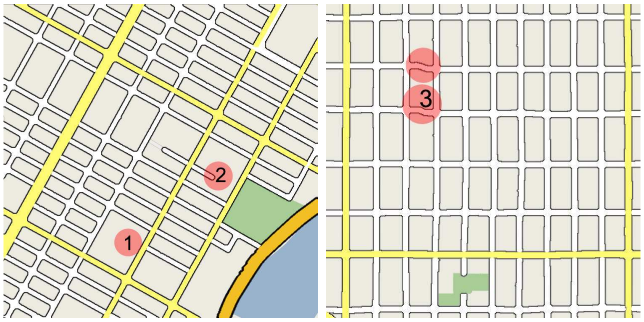

Figure 6: Left: Occasionally cells are merged together (1) or partially split by dead ends (2). Right: Slight irregularities can be seen in a regular grid(3).

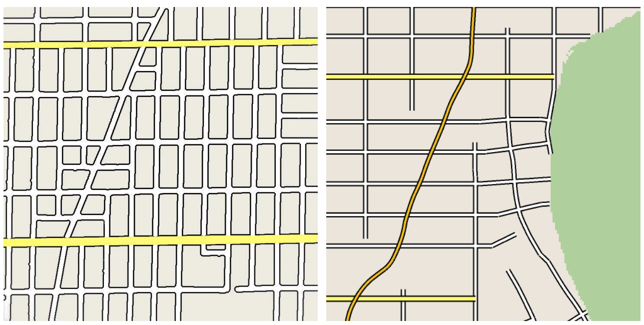

Figure 7: Left: This map shows an example from Chicago, where a single street is laying over an otherwise regular north-south grid pattern. Right: A similar pattern was created using our system.

important insight is that these irregularities have to be modeled on the graph level, by procedurally or manually selecting the street segments that should be deleted.

- *Layered Patterns*: A seemingly random street cuts across an otherwise regular street network. The street can have a random beginning and a random end. See figure 7 for an example.
- *Noise*: Most street patterns occur in a slightly distorted fashion. See figure 8 for examples. In our modeling system we use Perlin Noise [Perlin 1985] to either rotate the tensor field or alter the street segments and nodes in the graph.

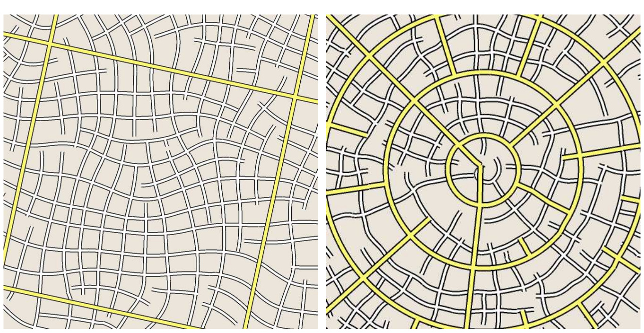

Figure 8: This figure shows a regular major road grid (left) and a radial major road pattern(right) over slightly curved minor roads.

Crack Patterns: There are some instances where road networks share some similarities with fracture patterns. One example are major roads in rural Missouri (see figure 9 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 9: This figure shows crack patterns in Missouri (left) and a procedurally generated patterns using our system (right) .

5 Editing Tensor Fields and Street Graphs

Overview: There are two levels of editing operations that we provide the user with. First, the user can change the street network by modifying the underlying tensor field. Second, the user can directly change the street network by adding, modifying, or removing street segments. However, such changes can be lost if any changes are made to the underlying tensor field afterwards. In the following we will first describe the editing operations on tensor fields followed by editing operations on the graph structure. Figure 10 illustrates several steps in an editing session.

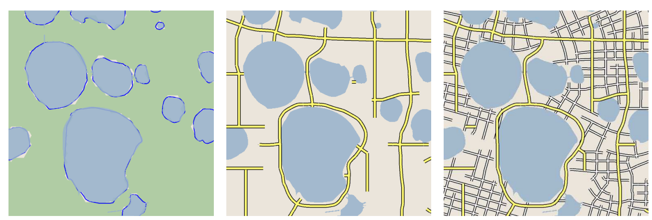

Figure 10: This figure shows a work flow through our system. Typically a user first creates a layout of the major roads and then fills in minor road patterns.

Tensor Field Editing: To change the tensor field, we provide the following functionalities.

1. **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 constant directions or radial patterns near a given location. Our implementation follows closely the tensor 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 such that the resulting tensor field satisfies the user specifications. The user can also delete an existing design element or modify its location, orientation, and isotropic and isotropic scales. Note that there are other ways of creating a tensor field from user constraints, such as relaxation [Turk 2001; Wei and Levoy 2001] and propagation [Praun et al. 2000]. We choose the idea of basis fields due to its simplicity and intuitiveness.
2. **Tensor Field Smoothing:** The user can reduce the complexity in the tensor field by perform componentwise Laplacian

smoothing. Such an operation can be performed either globally or locally. In the latter case, the tensor values on the boundary of local region serve as the constraints in relaxation. Smoothing tends to greatly reduce the complexity in the tensor fields.

3. **Topological Editing:** The user can explicitly control the number and location of the degenerate points in the field. This is achieved by employing the degenerate point pair cancellation and movement operations. Singularity pair cancellation allows a degenerate point pair to be removed simultaneously, while degenerate point movement enables a degenerate point to be moved to a more favorable location. Notice both operations provide topological guarantees that no other degenerate points are affected.

4. **Brush Interface:** We also use the idea of a brush-based interface, in which the user produce tensor values by moving the mouse to form a curve or a loop. Then a region is found to have a pre-defined distance to the curve. Finally, the tensor values inside this region are computed by treating the user-specified curve as the constraint. Notice this is similar to creating a tensor field with constraints such as coastlines and boundaries of a park. The difference, however, is that the brush-based interface allows tensor field to be created locally instead of globally and supports discontinuities in the tensor field. More importantly, tensor field can become discontinuous along the boundary of the region. An example operation is illustrated in figure 11

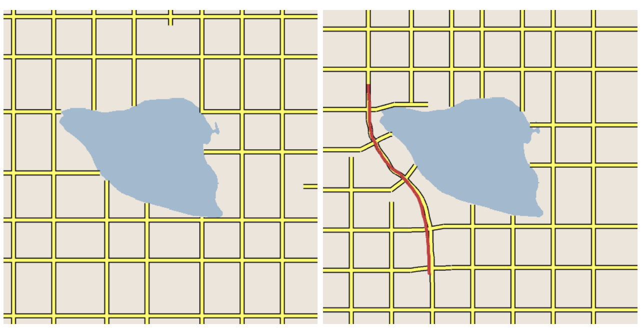

Figure 11: This figure shows the application of the brush tool to orient streets along a brush stroke.

Notice that the first three functionalities follow closely of the tensor field design system of Zhang et al. [2007]. On the other hand, the brush interface is novel. It is easy to use, provides local control, and allows discontinuities to be created in the tensor field.

Graph Editing:

1. **Road Segments 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 (user clicks on vertex and moves using drag and drop)

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

4. **Move Streets:** the user can move street a street in the tensor field so that it is retraced from a close-by location.

To handle discontinuities across two neighboring regions, we allow 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 will be clipped inside the second region minus the intersection region, and vice versa. In the second case which is *asymmetric*, the end points of the roads inside the region of intersection are used as seed points to generate road in the second region.

To demonstrate the capabilities of the editing tool we show two edits of a scene using a water map from the Mission Bay in San Diego (see figure 12 and figure 13 for an added node in the top right corner).

Figure 12: This figure shows a generated street graph for the Mission Bay in San Diego.

6 Street Graph Generation from Tensor Fields

Our streamline tracing algorithm is an adaptation of [Jobard and Lefer 1997], which has been used in pen-and-ink sketching of 3D shapes [Hertzmann and Zorin 2000] and quad-dominant remeshing of surfaces [Alliez et al. 2003; Marinov and Kobbelt 2004; Zhang et al. 2007].

Given a second-order symmetric tensor field $T(x,y)$, we produce two family of streamlines corresponding to the major eigenvector field $E_1(x,y)$ and the minor eigenvector field $E_2(x,y)$, respectively. To trace the major streamlines, we start from a set of initial seed points. The seed points can be either specified by the user or generated procedurally, and they are placed in a priority queue. Next, we enter an iterative process in which a streamline is generated based on the top element in the queue, while new seeds are added to the queue. To trace a single streamline, we use an adapted Runge-Kutta scheme [Cash and Karp 1990] that has been modified to handle tensor fields. Given a position of the current end point, we find the direction in which the streamline grows by finding the major eigenvector value at the end point. To remove the sign ambiguity in eigenvector directions, we use the direction in which the current point has come from. The next integration point is then found using the numerical scheme. A streamline stops growing if it hits the boundary of the domain, runs into a degenerate point, is too close to

Figure 13: This figure shows the street graph from the previous figure with an added radial pattern.

an existing streamline by exceeding a user-defined density d_{sep}, returns to its origin which indicates a loop, or exceeds a user-defined maximum length. Once a streamline has been traced, additional seed points will placed along it at a distance of d_{sep}. Notice d_{sep} is used to control the density of the streamlines. Next, we trace the streamlines that correspond the minor eigenvector field in a similar fashion.

The two families of streamlines can be used to generate a graph $G = (V, E)$. This is done by finding the intersection points between any pair of a major streamline and a minor streamline. V is the collection of intersection points, and E is the set of segments between two consecutive intersection points along a major or minor streamline. 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 inbetween roads.

7 Three-dimensional Geometry Generation

In the last sections we described how a user can generate a street network. The street network is a graph that consists of streets and intersections. In the following we describe the steps necessary to generate three dimensional geometry from a street network. We employ a method that allows the specification templates for street segments and crossings similar to [Thomas and Donikian 2000]. The approach works as follows:

- A street segment can be specified by various attributes of the cross section. This method is typically used in urban design concepts where an urban planner would draw cross sections to convey his design (see figure 14). We store cross sections as a list of lanes. Each lane has attributes including width, texture information, and type. We implemented sidewalks, vegetation, parking lanes, lanes for cars, curbs, etc.

- An intersection can be specified by various attributes about (1) traffic lights, (2) markings on the floor including pedestrian crossings, yield lines, stop lines, and arrows , (3) texture, (4) and geometric information about the smoothness of corners, and (5) intersection type. Our current implementation allows for X-, and T-intersections and roundabouts. Unfortunately, the generation of intersection geometry and texture coordinates involves a large amount of tedious geometric computations. We refer the reader to the civil engineering literature [AASHTO 2004] for a comprehensive treatment of the topic.

Selected example models can be seen in figure 15. The figure shows three crossings together with short sections of street segments leading up to to crossing.

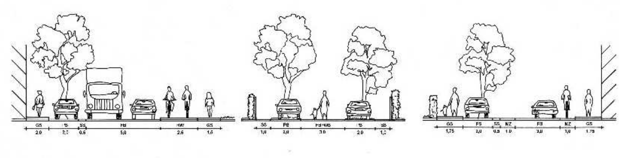

Figure 14: This figure shows two street cross sections.

8 Renderings

We combined the street networks with simple shape grammars for parcel subdivision and building mass model generation. The final images were created using RenderMan with ambient occlusion. See figure 16 for two renderings of the San Diego scene and one rendering of a radial city.

9 Discussion

In the following we discuss strength and limitations of our approach and our contribution to computer graphics research.

Strengths: The inherent strengths of tensor fields include the possibility to model street patterns, which usually contain two most preferred directions that are mutually perpendicular. Furthermore, tensor field design allows the user to quickly generate an initial street layout with which he or she can modify 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 the multi-scale editing capabilities. Another direction we wish to explore is the use of *asymmetric* tensors to model street networks whose two preferred directions are not always orthogonal.

Street Modeling for Computer Graphics: An interesting question is to compare our street modeling tool to street modeling in real urban environments. There are several important characteristics that distinguish a computer graphics application and a civil engineering application. We are mainly concerned with efficient large-scale modeling. Large-scale editing is very difficult in reality, because it is very expensive to tear down existing houses. In areas of rapid growth, such as Atlanta or Phoenix, a tool like ours could be used in early design stages to design roads in larger residential subdivisions. Road construction in civil engineering is significantly more concerned with local details. Examples of important factors

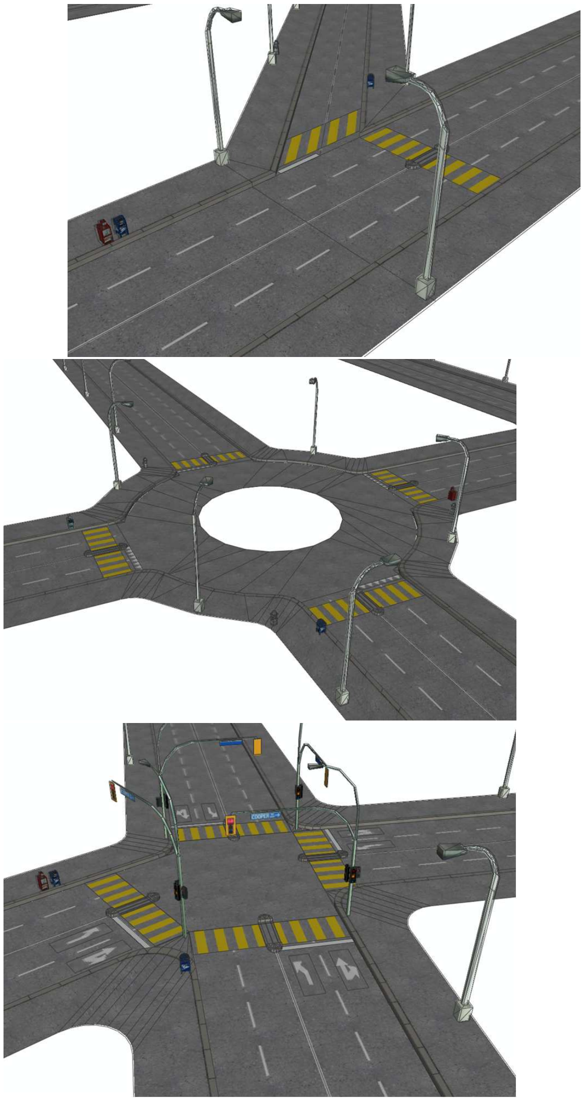

Figure 15: This figure shows street intersections.

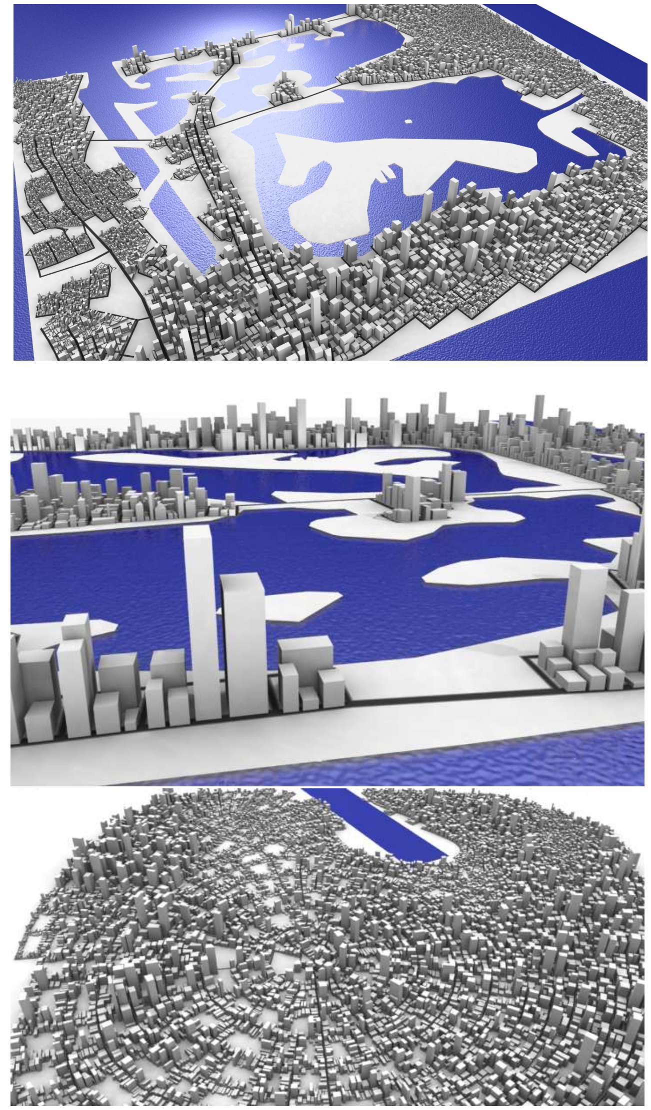

Figure 16: This figure shows two renderings of the San Diego scene and another city using a radial pattern.

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 was 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 increase modeling times. However, it has been our experience, that 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.

Graph Modeling: 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 user 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.

10 Conclusion

In this paper we presented a solution to interactively model street graphs. The main ideas of this paper are to (1) use tensor field modeling to guide the generation of a graph and (2) to integrate procedural modeling with interactive editing. These two concepts showed to be very useful to generate street networks, and we plan to extend this modeling strategy to other graphics modeling problems.

References

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

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.

FEDERL, P., AND PRUSINKIEWICZ, P. 2004. Finite element model of fracture formation on growing surfaces. In *International Conference on Computational Science*, Springer, M. Bubak, G. D. van Albada, P. M. A. Sloot, and J. Dongarra, Eds., vol. 3037 of *Lecture Notes in Computer Science*, 138–145.

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.

HIROTA, K., TANOUE, Y., AND KANEKO, T. 1998. Generation of crack patterns with a physical model. *The Visual Computer 14*, 3, 126–137.

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.

LEFEBVRE, S., AND NEYRET, F. 2002. Synthesizing bark. In *Rendering Techniques (Eurographics Workshop on Rendering - EGSR)*.

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*.

NEFF, M., AND FIUME, E. 1999. A visual model for blast waves and fracture. In *Graphics Interface*, 193–202.

O'BRIEN, J. F., AND HODGINS, J. K. 1999. Graphical modeling and animation of brittle fracture. In *Proceedings of ACM SIGGRAPH 1999*, ACM Press/Addison-Wesley Publishing Co., 137–146.

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.

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

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.).

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.

SMITH, J., WITKIN, A., AND BARAFF, D. 2001. Fast and controllable simulation of the shattering of brittle objects. *Computer Graphics Forum 20*, 2, 81–91.

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.

THOMAS, G., AND DONIKIAN, S. 2000. Modelling virtual cities dedicated to behavioural animation. *Computer Graphics Forum (Proc. Eurographics '00) 19*, 3 (Aug.), 71–80.

TURK, G. 1991. Generating textures on arbitrary surfaces using reaction-diffusion. In *Proceedings of ACM SIGGRAPH 91*, ACM Press, 289–298.

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

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.

WITKIN, A., AND KASS, M. 1991. Reaction-diffusion textures. In *Proceedings of ACM SIGGRAPH 91*, ACM Press, 299–308.

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.

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.