AN ABSTRACT OF THE THESIS OF

Gregory Esch for the degree of Master of Science in Computer Science presented on December 4, 2008.

Title: Visualization and Design Systems for Road Infrastructure

Abstract approved: ______________________________

Eugene Zhang

Transportation infrastructure provides a vital service for the functionality of a city. The efficient design of road networks poses an interesting topic in computer science for digital content developers. For civil engineers, the visualization of analysis results on infrastructure both efficiently and intuitively is crucial. The following contributions are made to these areas:

- Street Network Design - A system built around tensor field design is proposed. Necessary background information is discussed and a set of tools are developed. Generated street networks are shown to demonstrate the effectiveness of the system.

- Bridge Analysis Visualization - A unique system to visualize the analysis results of bridges is discussed. Methods used to procedural model a bridge and visualize analytical information over the generated model in an intuitive manner are detailed.

Visualization and Design Systems for Road Infrastructure

by

Gregory Esch

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented December 4, 2008
Commencement June 2009

ACKNOWLEDGEMENTS

My greatest appreciation goes out to Eugene Zhang for his years of guidance and support, without which, this work would not have been possible. Furthermore, he has been pivotal in ensuring my success throughout my course work and in my personal life as well.

I would like to thank all the people who have contributed insight and guidance for the street network design system. Guoning Chen deserves special thanks for his contributions to several areas of the street design system and without his help, this work would not have been completed. To my collaborators Peter Wonka and Pascal Müller, goes many thanks for all the insightful discussions. Patrick Neill and Madhusudhanan Srinivasan have both contributed to the supporting media of the publications, without which, the success of this work would not be ensured. For their help on the bridge visualization, I would like to thank Eugene Zhang and Michael Scott. Without Michael, the bridge visualization project would not exist. Eugene has contributed toward many discussions and provided valuable insight into ways to improve the system's visual quality.

Those who have acted as mentors will have my eternal thanks. Ronald Metoyer has been a pivotal figure in pushing me toward the various people in my college career. He led me to both Eugene Zhang and Michael Scott. Michael was vital for getting me into graduate school and finding funding to help pay for my endeavors.

Finally, my time in graduate school would not be possible without funding. My graduate school experience has been funded in part by the Oregon Department of Transportation (ODOT) MBMSFY05-036. An important source of funding for Eugene and Guoning is NSF CCF-0546881.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

DEDICATION

To my family and all those who helped me along the way.

Chapter 1 – Introduction

The majority of the world's population now lives in cities. Long being the centers of human innovation, culture, and commerce, cities have always been important to human civilization. Vital to the survival of a city, transportation infrastructure provides an interesting topic of study. Of special importance is the road infrastructure that allows people to travel between destinations quickly.

While the road infrastructure is a vital part of a city, it is often overshadowed by the architecture. The architecture of a city is often considered an important factor in the look and feel of a city. Much of the research on the digital creation of a city has concentrated around the generation and placement of the buildings [71, 43, 36]. Unfortunately, the generation of the road network itself is often overlooked. As first indicated by Parish and Müller [48], the generation of the road network is an essential first step in the digital creation of a city due to the use of the roads for determining the placement of buildings.

Large scale digital content creation is becoming increasingly important as games continue the move toward larger worlds and movies attempt to produce even more convincing cities that exist only in the story. This necessitates the creation of a model for the city. The creation of fully detailed road networks by editing individual streets is considered the de facto method. Unfortunately, this can take several man-years to complete a relatively large sized network. Proposed methods using procedural techniques [48, 71, 43] lack necessary user control to be effective. A system based on tensor field design is outlined which combines local and global constraints, giving the user the ability to design flexible and complex road networks while still giving them direct control over the road network.

The street modeling system realizes several contributions to the area of large scale street network modeling:

- The connection between tensor fields and street networks is made.
- Street patterns are analyzed and a matching pattern in tensor fields is described.
- New tensor field design tools are developed, such as a new brush stroke interface.
- An improved hyperstreamline tracing algorithm is presented which helps satisfy the unique requirements of city maps.
- Methods which allow the local editing of graphs generated by tensor fields are introduced.

While the road network is important to the digital creation of a city, roads and support structures are even more important to the general functionality of a real city. As a component of the transportation infrastructure, bridges require special consideration. Bridges are expensive to build and must be made to withstand large forces over long periods of time. Unfortunately, many of the bridges in existence today were built during the interstate boom of the 1950s. As many of these bridges near the end of their operating lifetime, they need to be inspected. In Oregon alone, manual inspection of bridges have indicated that approximately 500 bridges show signs of diagonal cracking [45, 27].

The large number of bridges in need of repair has led to the creation of a research program at Oregon State University to prioritize the bridges for repair. The system that was developed to achieve this task has two parts: analysis and visualization. The visualization system is unique in that it takes standard bridge measurements and creates a procedurally generated bridge model. After analysis, the model is then used to visualize the results of the analysis. The analysis portion of the system is not a major contribution of this thesis and is relegated to other citations. The system has been developed in close collaboration with bridge engineers at the Oregon State Department of Transportation (ODOT) to ensure the system meets their needs.

1.1 Related Work

The following is a review of the work related to the topics discussed in this thesis. Other systems for the generation of street networks and graphs, along with other directional field design systems, are discussed, followed by a discussion on topics related to bridge analysis. As a special note, the street modeling system presented in this thesis has also been published at SIGGRAPH [19, 14] and a paper on the bridge analysis visualization is under review for a special issue of the Journal of Computing in Civil Engineering on 3D Visualization in Civil Engineering [18].

1.1.1 Street Modeling

Street modeling is a problem that has been explored previously, with the most notable work being that of Parish and Müller [48]. The following sections discuss works in the areas of city modeling, road construction, and graph generation. Also discussed are systems for directional field design which are important for the system presented in this thesis.

A recent paper by Aliaga et al. [5] proposes an urban layout system that allows the user to interactively generate a city layout based on exemplars. This technique is similar to texture synthesis. While their results indicate a large number of layout patterns can be generated, their technique is less flexible and does not afford the user as much control as the system proposed here.

1.1.1.1 City Modeling

While the work on street modeling in this thesis concentrates on the design of the road networks, the system proposed here can be combined with shape grammars for architecture [71, 43, 36] to create a complete city generation system.

1.1.1.2 Road Construction

Literature on the construction of roads can be found in civil engineering texts. ASHTOO 2004 [2] provides a comprehensive review and is recommended reading. Also of important note is the Highway Capacity Manual [11] and the textbook by

Mannering et al. [37].

The area of urban design can also shed light on the layout of road networks. Some texts on the subject are Punter [57], Alexander et al. [4], Hillier [28, 29], and Gingroz et al. [21]. Despite the great deal of text in the area of urban planning, the greatest asset to the development of the system is the many online map systems freely available.

1.1.1.3 Graph Generation

Parish and Müller presented the most successful algorithm for the generation of street networks [48], extending L-systems [54]. L-systems have been used with great success in the modeling of vegetation by Prusinkiewicz and Lindenmayer [54], Prusinkiewicz et al. [56, 55], and Měch and Prusinkiewicz [40].

Other types of graph generation that may lead to unique patterns found in street networks are works done in the areas of ice ray lattice modeling by Stiny [63], work by Legakis et al. on mortar and brick layouts [35], diffuse limited aggregation by Witten and Sander [70], and cracks in Batik renderings by Wyvill et al. [73]. Unfortunately, the similarity to the discussed works is only applicable to a small set of street networks, making a useful generalization difficult.

Other areas where street patterns can be found is in works on layout algorithms using the Voronoi diagrams of randomly distributed points [10]. This idea has been applied to texture generation [72], mosaics [24], fracture patterns [60, 42], and even some street patterns [64, 22]. Jigsaw image mosaics [33] may also be used to generate a graph. Finally, an algorithm developed for the modeling of venation patterns in leafs [58] can be used to produce a graph.

1.1.1.4 Directional Field Design

Directional fields have been used in the past for many applications. Vector fields can be found in texture synthesis systems [66, 68, 51], fluid flow simulation [39, 61], and visualization [67]. Tensor fields and higher order N-way Rotational Symmetry (N-RoSy) fields have found applications in quad-dominant remeshing [6, 38, 17] and pen and ink sketching on surfaces [25, 74, 47].

Most of the previously mentioned works do not provide much interaction with the user. They mostly smooth the fields to remove noise and reduce the complexity of the directional fields. Several design systems have been proposed which are intended to address the lack of interactivity. For vector fields, the works by Zhang et al. [75] and Fisher et al. [20] are two competing methods. Both produce similar results. For tensor fields, Zhang et al. [74] propose a system using basis fields that is also the foundation for the street modeling system presented here. Palacios et al. [47] propose a design system for N-RoSy fields that allows the design of higher order tensor design systems.

1.1.2 Bridge Analysis

Several methods have been used in practice to analyze the structural integrity of bridges. Similar to the approach taken in the system proposed in this thesis is the load and resistance factor (LRFR) framework as outlined in AASHTO 2003[1] and Minervino et al. [41] and the reliability based approaches for moment and shear individually [62] and reliability based approaches for combined moment-shear effects [26]. Rating factors, or alternatively reliability indices as outlined in Akgül and Frangopol [3], are at the core of these analysis and allow bridge engineers to determine if a bridge meets an acceptable level of safety.

The augmentation of a complete visual system with a bridge analysis is unique to the system discussed here. Recently, though, there has been work in civil engineering to apply visualization to the architecture, engineering, and construction of civil infrastructures [32].

1.2 Organization

This thesis is broken up into two parts correlating to the discussion on the two systems. The first part spans chapters 2 - 7 and covers the street modeling system. The second part on the bridge analysis visualization system is presented in chapters 8 - 11.

The discussion on road modeling is in six chapter: a system overview (Chapter 2), tensor field background information (Chapter 3), tensor field design (Chapter 4), road extraction (Chapter 5), graph editing (Chapter 6), and results (Chapter

7). The overview in chapter 2 discusses in detail the problem being solved and motivation behind design choices made throughout the system. Next, to give a better understanding of tensor fields, chapter 3 is devoted to theoretical information. Chapter 4 continues the discussion on tensor fields and explains the various tools used to edit the tensor field. Once the tensor field is designed, chapter 5 discusses how to generate the road networks. The last major component of the design system, street graph editing, is discussed in chapter 6 and describes the tools given to the user to modify the street graphs directly. Finally, chapter 7 closes out the first part with a discussion on the results.

After the first part of the thesis concludes, the second part discusses the techniques used to visualize bridge analysis information. Chapter 8 gives some background information on the bridge analysis procedure. The next chapter, chapter 9, highlights the graphics techniques used to generate a visual representation of the bridge. After explaining how to generate the image of the bridge, chapter 10 explains in detail the method used to visualize a scalar field over the top of the bridge. The second part of the thesis closes out with a discussion of the bridge visualization system and presents some results in chapter 11.

At the end of the thesis are a few closing words on the techniques used and what areas can be improved in the systems presented.

Chapter 2 – Street Design System Overview

When faced with the task of designing large road networks, the limitations of using a purely procedural technique and the time consuming nature of manual editing lead to the desire for a system with design goals that take advantage of both ideas. The main advantage of using a procedural technique is the ability to create large scale road networks quickly. Unfortunately, procedural techniques [48, 5] do not give the user much flexibility and current methods to control them are unintuitive. Manual editing systems, such as Maya or 3ds Max or those proposed by Bruneton and Neyret [12], allow the user to place roads wherever they want and the tools used are quite similar, or the same, as tools they would use for other tasks.

The street modeling system presented here combines the benefits of both procedural and manual techniques and will be shown to be *powerful*, *intuitive*, and *flexible*. To be powerful, the system must allow the user to place many roads at the same time; to be flexible, the system must be able to take advantage of many different forms of user input; to be intuitive, the system must be able to give the user the ability to predict the effects of editing operations and be similar to tools already used by artists.

The power of the design system can be demonstrated through its use of patterns, which allow a user to quickly place large numbers of roads. With the limited user base and the lack of a comprehensive user study, it is difficult to say the system is truly intuitive, but it does lead the problem of street modeling more intuitive as it is easier to use and more straightforward than procedural techniques that use text based grammars. Additional editing operations can quickly and easily be applied to both the tensor field and the graph editing, leading to the system being quite flexible.

This chapter is devoted to building up a vocabulary on street networks. It will also show the connection between street networks and tensor fields. The final objective of this chapter is to build up goals for the tools offered by the system.

2.1 Analysis of Map Features

To determine a basic set of tools that a user might want to utilize, features of real city maps were examined. While the specific character of each city is unique, every map contains similar basic properties. This section covers those features of a city which have been identified to have the greatest impact on the visual character of a city and to develop a vocabulary to be used in the rest of the discussion.

2.1.1 Geographical Features

There are many factors that contribute to the appearance of a road network. Perhaps one of the most influential factors is the geography of the area that the roads are being built on. Common influential features include rivers, lakes, terrain, and parks. The following takes a closer look at these elements, most of which are visible in Fig. 2.1.

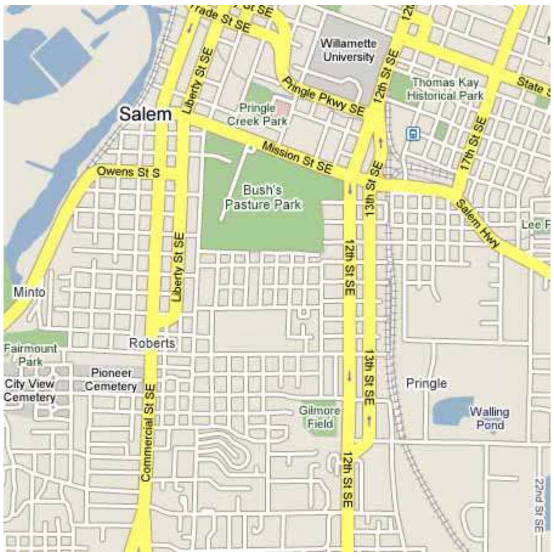


Figure 2.1: A map from Salem, Oregon showing various features of maps.

- **Parks** - Designated areas in a city that are set aside for recreational or conservation purposes. These are shown as green in Fig. 2.1. A park can also be characterized as a region on a map where the road density decreases drastically across a boundary. Many parks cross several road boundaries and have roads which outline them. Larger parks may also have sparse roads running through them.

- **Rivers and Lakes** - Bodies of water usually block the path of roads. They also typically act as a boundary which roads follow. Rivers also have bridges which are built over them to cross. In Fig. 2.1, a river can be seen on the left part of the image and several small lakes are visible throughout.

- **Road Density** - Road density typically changes as a person travels further into the outskirts of a city. As roads travel toward the center of a town, the density typically increases to provide for increased flow of traffic. As the density of roads is a function of the population density, highly populated regions in a city typically have a very high road density as well. Fig. 2.1 shows this change from north to south.

- **Orientation Transition** - In many road networks, the orientation of the roads shift from one direction to another. In Fig. 2.1, the roads can be seen to change from a north to south direction in the center of the figure to a northeast to southwest at the top of the figure.

- **Terrain** - When the terrain becomes too steep, roads usually curve around hills and mountains to allow vehicles to need less power to traverse them. This is more prevalent in mountainous road networks, and Fig. 2.1 does not have an example.

2.1.2 Road Hierarchy

Street networks are typically separated into different types of roads. In North American cities, there are typically three levels of roads in a city map: highways, major roads, and minor roads. Fig. 2.2 shows a section of a city map to be used in this discussion. Highways, as shown in orange, are large thoroughfares that allow large concentrations of traffic to travel at high speeds. They are typically identified as large roads that follow none of the patterns discussed in the previous section. Major roads, marked in yellow, are roads which carry large volumes of traffic and provide access across different parts of a city. Finally, the minor roads, shown in white, are the roads which provide local access only and carry a low volume of traffic.

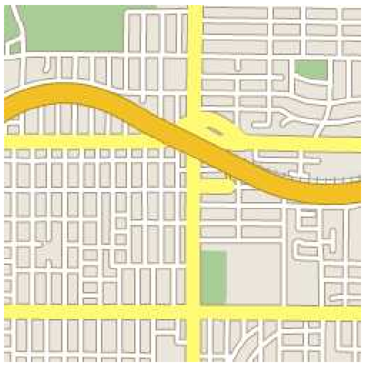


Figure 2.2: A map has three sets of roads forming a hierarchy. Highways are in orange, major roads in yellow, and minor roads in white.

2.1.3 Road Patterns

Street networks in real cities contain many different *patterns*. In the context of street networks, a pattern refers to features that affect more than a single road, and usually impacts several roads. Some patterns affect the entire road network, while other patterns are local to a small set of roads. Below is a classification of the various features that can be seen in a road map.

- **Regular Grid** - A regular grid is characterized by roads which form consistent rectangular parcels. These are the most common road pattern, as they allow for the most efficient use of space. Fig. 2.3(a) shows an example of a regular grid.

- **Radial** - Important on a larger scale for cities with a central monument or major area of congregation, the circular pattern is characterized by roads which travel around a central point. In Fig. 2.3(b), a radial pattern is shown. This pattern is from a designed neighborhood,F so the pattern is compact.
- **Individual Roads** - In most cities, individual roads do not follow the pattern of the surrounding roads. While not classified as a pattern, these roads are sometimes important to the character of a city map. An example from Chicago, Illinois is shown in Fig. 2.3(c)
- **Noisy** - In several parks, roads don't seem to follow a specific orientation, but appear windy. Fig. 2.3(d) shows a section of roads in the mountainous regions of Oregon also follow a winding path. These roads have been classified as being *noisy*.

2.2 Street Networks and Tensor Fields

The road patterns presented in Section 2.1.3 can be shown to correlate to *tensor fields*. Intuitively, a tensor field can be thought of as an orientational field. At any given point in a field, the orientation of a line can be found. In essence, this means that a tensor field can give a path to follow, but it does not tell which direction

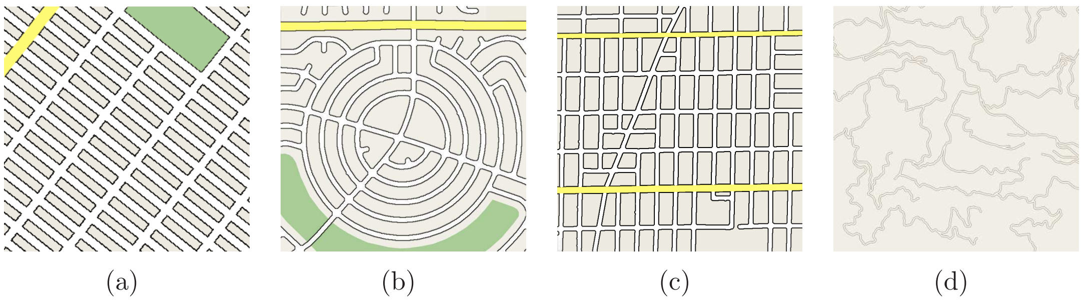


Figure 2.3: Examples of common patterns in road networks. a) A regular grid in Brooklyn, New York. b) A radial pattern found in Scottsdale, Arizona. c) An individual street following its own path in Chicago, Illinois. d) A noisy pattern found in the Oregon Cascade Mountains.

to follow along the path. This is important for road networks, as traffic can travel both directions along a road, so a road can be looked at as the path.

The connection can be further solidified by looking at the pattern of paths traced through various tensor fields. As shown in Fig. 2.4, a regular field exhibits behaviors similar to that of the regular grid pattern of a street.

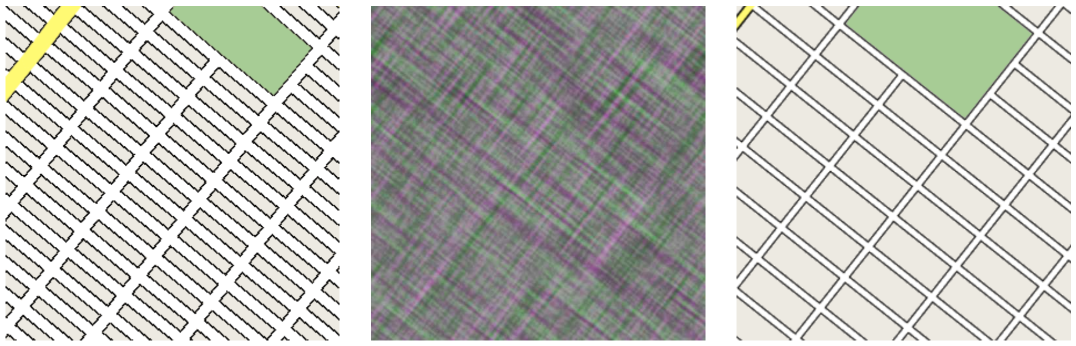


Figure 2.4: The regular grid from Fig. 2.3(a) and a tensor field and extracted roads to produce a similar road network.

Another common pattern, the radial pattern can be found in several street maps of European cities. The sequence of images in Fig. 2.5 show that a corresponding street pattern can be generated using tensor fields as well.

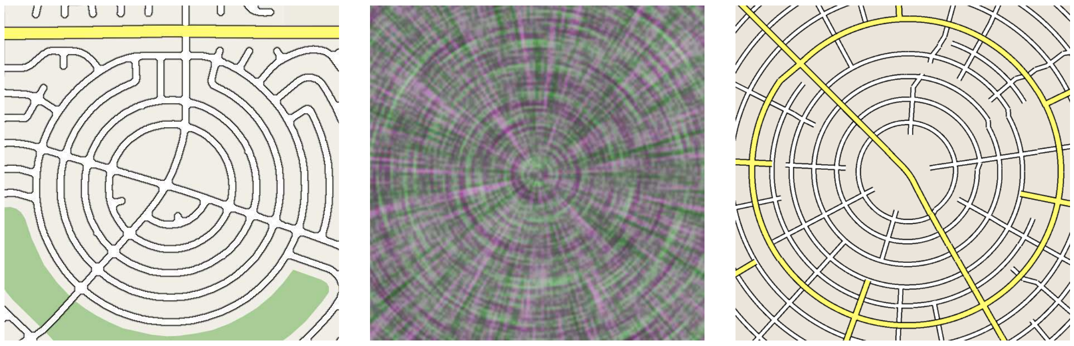


Figure 2.5: The radial pattern from Fig. 2.3(b) and a corresponding tensor field and extracted roads.

While the regular field and radial patterns are the most common, isolated examples of other tensor field patterns can be found as well. The following chapter will discuss how to use these patterns to create a comprehensive tool.

2.3 Pipeline Overview

To allow the user to produce a street network, a three stage process is proposed. Fig. 2.6 shows these steps.

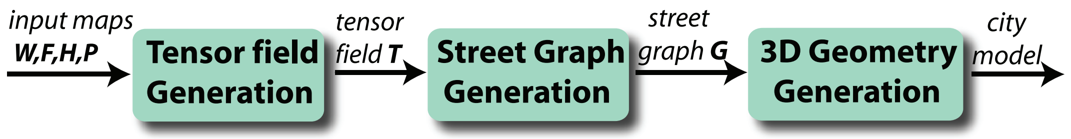


Figure 2.6: The modeling pipeline. Stage 1 is the tensor field design stage, roads are extracted in stage 2, and stage 3 is a geometry creation stage.

1. The first stage is a tensor field generation step. This stage is discussed in

detail in chapter 4. The input into the system at this point is a set of information maps. There are four separate maps: a water map W, a forest and park map F, a heightmap H, and a population density map P. Each of the maps represents a discrete function $f : [-X, X] \times [-Y, Y] \rightarrow [0, 1]$. W is represented as a binary image that indicates where a value of 1 maps to water, such as lakes, rivers, and oceans. A binary image also represents F, where 1 represents a forest or park. The forest and parks map is only used as a guide to the user. H is a grayscale image that corresponds to the height of the terrain at any given point. The final map, P, is also a grayscale image where 1 represents places of high density. The tensor field design stage ends with the tensor field, T, being exported to the next stage.

2. During the second stage, the street network is extracted from the tensor field. The process to do this is the topic of chapter 5. The input of the second stage is the tensor field T. This field is then used to generate a set of orthogonal *hyperstreamlines*, to be defined in chapter 3. The hyperstreamlines are represented as a set of polylines. The polylines are then merged to generate a graph $G = (V, E)$ that represents the road network, where V is the set of intersections and E is the set of road segments.

3. The third stage is a geometry creation stage. This stage is not a contribution of this thesis. To generate the geometry seen, the software package CityEngine from Procedural, Inc. [53] is used.

Fig. 2.7 shows the envisioned workflow for the street modeling system. The input maps provided by the user are used as input to the system (a) and used to generate an initial tensor field based on the boundaries of the water map. Additional editing operations are performed on the tensor field by the user (b) and the major road network is extracted from the field (c). Additional modifications are made to the tensor field (d) and the final major road network is again extracted from the field (e). Upon satisfaction with the major roads, the user then segments the tensor field into smaller regions and generates a tensor field for the minor road network (f). To see if the street network is acceptable, the minor roads are then extracted (g). Similar to the major roads, the editing can continue on the tensor field for the minor roads until the user is satisfied with the results (h, i).

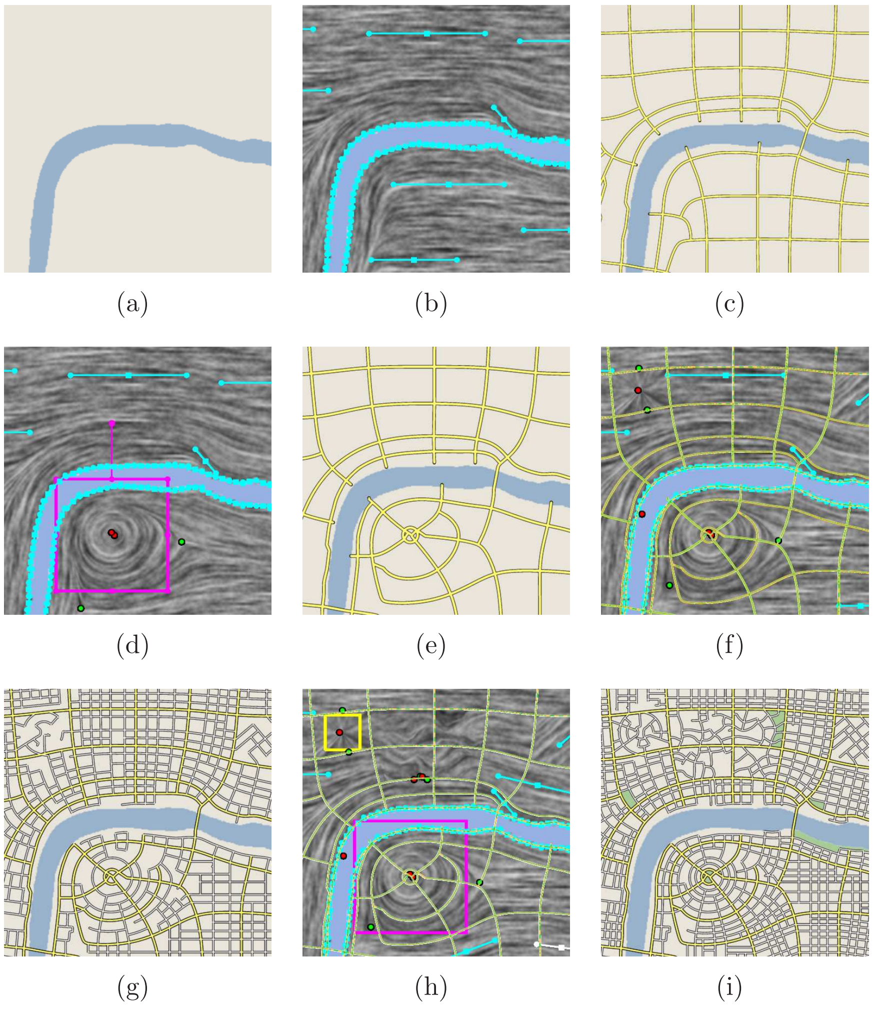


Figure 2.7: A sequence of an envisioned editing session in the street modeling system.

Chapter 3 – Tensor Field Background

Having identified tensor fields to be used as the basis for the street editing, it is necessary to give a deeper understanding of what they are and how to derive necessary values from the field. For a more in depth discussion, please view the paper by Delmarcelle and Hesselink [16].

3.1 Tensor Field Definitions

A tensor field T is formally defined for a manifold surface M as a smooth function that maps every point $\mathbf{p} \in M$ to a second order tensor value:

$$T(\mathbf{p}) = \begin{pmatrix} a(\mathbf{p}) & b(\mathbf{p}) \\ c(\mathbf{p}) & d(\mathbf{b}) \end{pmatrix} \tag{3.1}$$

where $a(\mathbf{p})$, $b(\mathbf{p})$, $c(\mathbf{p})$, and $d(\mathbf{p})$ are smooth functions.

This matrix is considered symmetric if and only if $b(\mathbf{p}) = c(\mathbf{p})$ for all $\mathbf{p}$. When $d(\mathbf{p}) = -a(\mathbf{p})$, the matrix is said to be traceless, as the sum of the trace of the matrix is 0:

$$trace(T) = \sum_{i=1}^{n} T_{i,i} = 0 \tag{3.2}$$

where $n = 2$ is the size of the matrix.

Given any symmetric matrix, the following decomposition can be used to find a traceless matrix:

$$T = S + A = \lambda I + \mu \begin{bmatrix} a & b \\ b & -a \end{bmatrix} \tag{3.3}$$

Furthermore, the matrix A can be translated to the following

$$A = \mu \begin{bmatrix} a & b \\ b & -a \end{bmatrix} = \mu \begin{bmatrix} \cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta \end{bmatrix} \tag{3.4}$$

This leads to the eigenvalues of the matrix being $\pm\mu$. The major eigenvector is found as

$$\mathbf{e_1} = \left\{ \lambda \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} \middle| \lambda \neq 0 \right\} \tag{3.5}$$

and the minor eigenvectors being

$$\mathbf{e_2} = \left\{ \lambda \begin{pmatrix} \cos(\theta + \frac{\pi}{2}) \\ \sin(\theta + \frac{\pi}{2}) \end{pmatrix} \middle| \lambda \neq 0 \right\} \tag{3.6}$$

Only the deviate tensor field $A(\mathbf{p})$ is of concern to the street modeling system. The field is equivalent to two perpendicular eigenvector fields, where $E_1 = \mu e_1(\mathbf{p})$ and $E_2 = \mu e_2(\mathbf{p})$. The two eigenvector fields are referred to as the major and minor eigenvector fields respectively. It is important to note that the major and minor eigenvector fields are not related to the major and minor road networks.

Another important property of the tensor field is the idea of degenerate points. A degenerate point, known interchangeably as a zero point or a singularity, is a point, $\mathbf{p_s}$, where $T(\mathbf{p_s}) = \mathbf{0}$. The mathematical properties at degenerate points correspond to the patterns discussed earlier. Chapter 4 discusses how to use these points to create a tensor field that contains many patterns.

3.2 Degenerate Point Properties

Degenerate points have several properties that help in identifying what type of point it is. One such property is the *tensor index.* Intuitively, the tensor index is the number of times the tensor field reorients itself when traveling on a closed path. This number can help give a better understanding of the structure of tensor fields.

3.2.1 Tensor Index

The tensor index describes how the orientation of the tensor field changes around an *isolated* degenerate point. A singularity is considered isolated when there is a closed region R around a degenerate point $\mathbf{p}_s$ such that R contains no other degenerate points, ie: $\mathbf{p}_s$ is unique in R. The tensor index is defined as the number of times the normalized major eigenvector field covers the unique circle S' on the closed path ∂R.

Tensor indices of degenerate points in a tensor field must be in increments of 1/2 due to the sign ambiguity that arises from using the eigenvector field. This makes sense intuitively, as a tensor field can be viewed as having no direction, only orientation, so the same field will be derived by rotating a field 180°.

3.2.2 Degenerate Point Types

Several uniquely named degenerate points have been discussed in previous tensor field design literature [74]. The following is a summary of these types.

- **Regular** - As shown in figure Fig. 3.1(a), a regular point **p** is defined as any point that is not a degenerate point. At **p**, the tensor index is 0.
- **Center** - Shown in figure Fig. 3.1(b), a center has a tensor index of 1.
- **Node** - A node is a degenerate point where all the hyperstreamlines travel directly to it, as shown in Fig. 3.1(c). The tensor index of a node is 1.
- **Wedge** - Fig. 3.1(d) shows an example of a tensor field around a wedge. A wedge has a tensor index of 1/2.
- **Trisector** - As shown in Fig. 3.1(e), a tensor field around a trisector has three spokes. It has a tensor index of -1/2.
- **Saddle** - The final type of degenerate point is a saddle, with four spokes as shown in Fig. 3.1(f). A saddle's tensor index is -1.

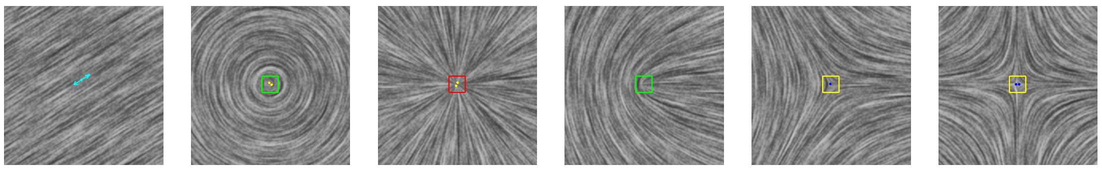

(a) (b) (c) (d) (e) (f)

Figure 3.1: Different types of degenerate points. a) Regular element. b) Center. c) Node. d) Wedge. e) Trisector. f) Saddle

3.3 Hyperstreamlines

A hyperstreamline is defined as a curve, P, that is tangent everywhere to an eigenvector field on its path. Hyperstreamlines can be defined as either *major* or *minor* depending on the underlaying eigenvector field. In the past, hyperstreamlines have also been used to visualize tensor fields [69], pen and ink sketching [25, 74] and quad remeshing of surfaces [7, 38, 74]. In the context of this paper, they are used as the individual roads in the street network. Just as with the major and minor tensor fields, the major and minor hyperstreamlines do not correlate to the major and minor road network.

Chapter 4 – Tensor Field Editing

With the necessary mathematical background of tensor fields explained, the set of tools offered by the system can now be developed. The discussion in chapter 2 revealed several points that will make a tool more usable:

- **Intuitive** - Editing operations on the tensor field must be able to be predictable. The user should be able to know what the result of an operation will be on a global scale.
- **Transparent** - The user must be able to maintain a minimal knowledge of the underlaying mathematical foundations to the greatest extent possible. This also indicates that operations must be able to provide a manner to edit the output in a way that the user can understand.
- **Flexible** - While the design system should be fairly comprehensive, incorporating new tools and ideas must be relatively simple.

All these points are great strengths for tensor field design. Through the use of graphical user interface (GUI) elements, the system is able to hide much of the mathematical foundations. For the variables that are to be input into the system without the use of symbols, the numbers should remain intuitive and their meaning clear. The rest of this chapter is devoted to explaining how the street design system incorporates these elements.

4.0.1 Computational Domain

To represent the continuous tensor field, a regular rectangular grid of $n \times n$ vertices is used. Tensor field values inside a cell is found by bilinearly interpolating [8] the tensor values at the four corner vertices of the cell.

Another advantage of using a rectangular based technique is the use of image based techniques. Since an image is used for various input maps, the rectangular representation more readily maps directly between the image domain and the tensor field domain.

Another commonly used representation for the underlaying field is a triangular mesh [74, 15, 20]. Such a domain has the advantage of mapping closely to mesh based surfaces. While such a mapping is well defined for surfaces, the mapping between images and the computational domain and several other important algorithms become more difficult. Since acquiring the tensor index is only used to give a better understanding of the tensor field and is not used in the system, using a triangular domain would not have been advantageous.

4.1 Tensor Field Visualization

The tensor field must be visualized to give the user the ability to view intermediate results without having to wait for the street network to trace. This allows the user to correct mistakes without waiting for the road extraction process to complete. To visualize the field, a modified image based flow visualization (IBFV) [67] is employed. The modifications are based on the tensor field approach in Zhang [74].

The rest of this section reviews these techniques in detail.

Image based flow visualization is an important directional field visualization technique that allows a vector field to be visualized interactively. The algorithm is flexible enough to allow many variations in appearance without significant modification. The algorithm works as follows: Given an image I_i, find the next image in the sequence I_{i+1} by blending between I_i which has been advected by the vector field and a base image B_i. The results in this thesis are made using a single noise base image.

Since a tensor fields contain a direction ambiguity and IBFV requires advecting the mesh points by a specified amount, a different formulation must be used. To deal with tensor fields, a three step process is used. Two IBFV images are generated and then blended together. Furthermore, since the IBFV technique requires a directional field, V, a field must be derived from the tensor field, T. It is important to note that it is not possible to create a field such that $S(T) = S(V)$, where $S(X)$ is the set of degenerate points in the fields. It is possible to create two directional fields, V_x and V_y, such that $S(V_x) \cap S(V_y) = S(T)$.

To obtain such fields, V_x and V_y are constructed such that they are nonnegative everywhere in the x and y direction respectively. In the following discussion, only the major eigenvectors are considered without a loss of generality. The following uses the polar coordinate representation of the eigenvector with ρ being the magnitude of the vector and θ being the orientation. The following methodology satisfies this requirement:

$$V_x = \begin{cases} \rho \begin{pmatrix} \cos\theta \\ \\ \sin\theta \end{pmatrix} if\cos\theta \geq 0, \\ -\rho \begin{pmatrix} \cos\theta \\ \\ \sin\theta \end{pmatrix} otherwise \end{cases} \tag{4.1}$$

$$V_y = \begin{cases} \rho \begin{pmatrix} \cos\theta \\ \\ \sin\theta \end{pmatrix} if\sin\theta \geq 0, \\ -\rho \begin{pmatrix} \cos\theta \\ \\ \sin\theta \end{pmatrix} otherwise \end{cases} \tag{4.2}$$

These two derivations are used to create two IBFV images, I_x and I_y which are then blended together. The blend function is simply $\alpha_x = \cos^2\theta$ and $\alpha_y = \sin^2\theta = 1 - \alpha_x$. As such, the final image is derived as $I_{i+1} = \alpha_x I_x + \alpha_y I_y$.

4.2 Element Based Design

There has been much discussion on the patterns that can be found in many city maps. In this section, a method to combine the various patterns is presented. The first idea to introduce is the concept of *basis fields*. A basis field is a tensor field that contains only zero or one degenerate points. They make for an easy representation by allowing the easy description of a tensor field and additional properties for a specific pattern. The element based design system presented here is based on previous works by Zhang et al. [74] and van Wijk [67].

4.2.1 Regular

As a city has many patterns that orient themselves to a grid, the regular element is a common pattern. Given a direction (u_x, u_y) at a point $\mathbf{p}$, the magnitude can be found as $l = \sqrt{u_x^2 + u_y^2}$ and the orientation $\theta = \arctan \frac{u_y}{u_x}$. Then, for every point in a region, a regular element can be defined by the following function:

$$T_r(\mathbf{p}) = \begin{pmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{pmatrix} \tag{4.3}$$

4.2.2 Center

Centers are the basis field for the radial pattern. As highlighted previously, they show up in many European cities or specifically designed modern cities. A center is defined at the origin as

$$T_w(\mathbf{p}) = \begin{pmatrix} p_y^2 - p_x^2 & -2p_x p_y \\ -2p_x p_y & -(p_y^2 - p_x^2) \end{pmatrix} \tag{4.4}$$

4.2.3 Node

Centers and nodes can be viewed as sister elements. A node is a center that has been rotated locally by 90°. They can be used commonly in areas where using a center would cause an undesirable sharp change in the orientation of the tensor field. A node is at the origin is defined as

$$T_w(\mathbf{p}) = \begin{pmatrix} p_x^2 - p_y^2 & 2p_xp_y \\ 2p_xp_y & -(p_x^2 - p_y^2) \end{pmatrix} \tag{4.5}$$

4.2.4 Wedge

Wedges are not quite as common in a city, but may be found on peninsulas. To design a basis field with a wedge at the origin, the following definition is used.

$$T_w(\mathbf{p}) = \begin{pmatrix} p_x & p_y \\ p_y & -p_x \end{pmatrix} \tag{4.6}$$

4.2.5 Trisector

Trisectors are commonly seen near areas where two major roads merge into a single road. They are also fairly uncommon. A trisector is defined as

$$T_w(\mathbf{p}) = \begin{pmatrix} p_x & -p_y \\ -p_y & -p_x \end{pmatrix} \tag{4.7}$$

4.2.6 Saddle

The final element is a saddle. The saddle is probably the least common of all the elements. No examples of saddles have been found in the study of maps. They are given as

$$T_w(\mathbf{p}) = \begin{pmatrix} p_x^2 - p_y^2 & 2p_xp_y \\ 2p_xp_y & -(p_x^2 - p_y^2) \end{pmatrix} \tag{4.8}$$

4.2.7 Radial Basis Functions

Once all the elements have been defined, they can be combined using radial basis functions. A radial falloff function is a function that has the property $\lim_{\|\mathbf{p}\|^2 \to \infty} f(\mathbf{p}) = 0$. The radial basis function provides a way to combine the previously specified fields so that the influence of the field falls off as $\mathbf{p}$ is further from the origin of the element. One such commonly used function is $f(\mathbf{p}) = e^{-d\|\mathbf{p}\|^2}$. Combining a basis field with the radial falloff function gives rise to

$$T(\mathbf{p}) = \sum_i e^{-d\|\mathbf{p}-\mathbf{p_i}\|^2} T_i(\mathbf{p} - \mathbf{p_i}) \tag{4.9}$$

where d is a falloff constant, T_i is one of the basis fields as specified above, and p_i is the center of the element. As shown in Fig. 4.1, this produces a smooth transition between two elements in the tensor field.

4.2.8 Global Rotation

Giving the user the ability to globally rotate a basis field is important to allow the user to orient the features of a basis field in the direction of their choice. To rotate an element, the rotation matrix R is found as

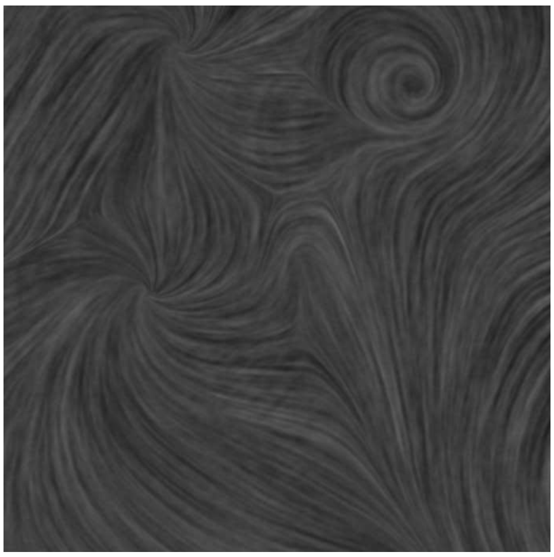


Figure 4.1: A tensor field generated using many design elements.

$$R(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \tag{4.10}$$

where θ is the desired amount of rotation. This is applied to the tensor matrix as

$$T_o = R^T T_i R \tag{4.11}$$

Here, T_i is the input tensor, T_o is the output tensor, and R is the rotation matrix in equation 4.10.

4.3 Tensor Field Smoothing

Tensor field smoothing simplifies the field inside a user specified region, ensuring the field inside the region smoothly interpolates the field at the boundary to the inside. Smoothing can be used to remove unwanted features, or reduce the noisiness of the field. To perform the smoothing operation, component wise discrete Laplacian smoothing is performed [7, 38, 74]. The operation is performed on a collection of points such that the point becomes the average of its neighbors. This has effect of removing any elements that are of opposite type. It will also take all the elements and make a single element in the middle of the smoothed region. The following shows how to setup the equations to be solved.

$$T(\mathbf{v_i}) = \sum_{j \in J_i} \omega_{i,j} T(\mathbf{v_j}) \tag{4.12}$$

where $\mathbf{v_i} \in M$ is a vertex in the grid, $T(\mathbf{v})$ is the tensor field function at vertex $\mathbf{v}$, J is the set of indices such that there is an edge between v_i and v_j, and $\omega_{i,j}$ is the weight between vertices v_i and v_j. Since the grid is regular, $\omega_{i,j}$ is $\frac{1}{|J_i|}$ for all edges. Due to the assumption of a deviate tensor, the following formulation can be used instead.

$$\begin{pmatrix} a(\mathbf{v_i}) \\ b(\mathbf{v_i}) \end{pmatrix} = \sum_{j \in J_i} \omega_{i,j} \begin{pmatrix} a(\mathbf{v_j}) \\ b(\mathbf{v_j}) \end{pmatrix} \tag{4.13}$$

For those vertices that are on the boundary of the region, $a(v_i)$ and $b(v_i)$ are held constant. Equ. 4.13 forms a linear set of equations with the matrix being a weighted adjacency matrix. Since the matrix is sparse and dominated by diagonal entries, the conjugate gradient solver is employed to solve the set of equations [52].

4.4 Brush Stroke Interface

One of the novel editing techniques introduced in the street design system is the brush stroke interface. Similar to a brush tool in an image editing application, the interface generates a field around and guided by a user given path. The user is able to set the distance from the line to be used. The interaction consists of two distinct parts: finding the vertices that the tool affects and determining the tensor values inside the region.

To determine which vertices are impacted by the tool, the first step is to find the cells that the user given path intersects. Once the cells are found, the classical fast marching algorithm [59] is used to determine the final set of vertices. This method works by building up a wavefront until some criteria have been matched. Fast marching is quite flexible in that the stopping criteria can be made to be anything from the orientation with respect to the nearest point on the path to the distance to the path. In the current implementation, only the distance to the path is taken into consideration.

Once the region is found, the same method discussed for smoothing is employed. Instead of using the vertices on the border of the region as fixed, the vertices which the user given path passes through are first calculated based upon building a tensor where the major eigenvector of the vertex is the tangent of the nearest point on the path.

The end effect is the desired result of having the field inside the region travel in the same direction as the input line as shown in Fig. 4.2.

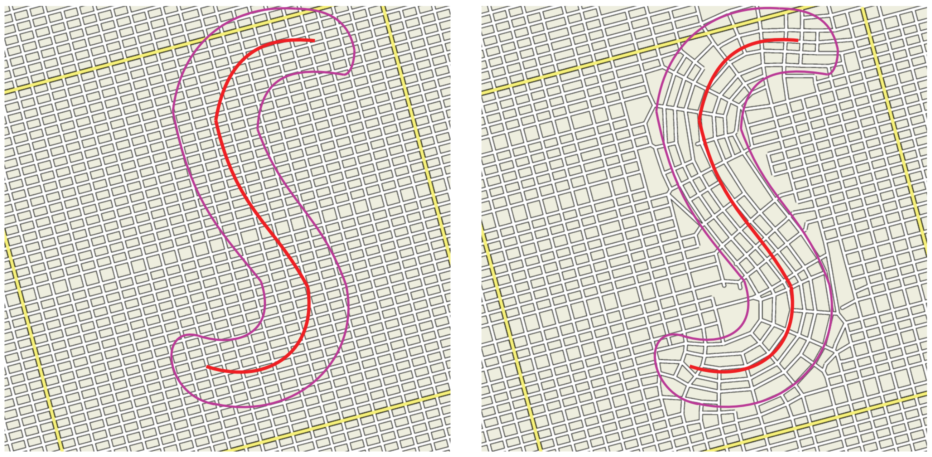


Figure 4.2: Example of the brush stroke interface. The image on left shows the network before the brush stroke was applied and right shows the effects of the brush stroke.

4.5 Boundary Fields

As rivers and oceans form natural paths for roads to follow, it is important to develop a tool that allows the roads along features to follow them correctly. To afford for this requirement, the system allows the user to input a binary image W, with 0 corresponding to locations where there is land and 1 corresponding to the places where there is water. This single image contains information on all the bodies of water, including all the rivers, lakes, and oceans. There is no restriction on the size of the image.

Unfortunately, due to the coarseness of the image based representation, the image data cannot be used directly. Instead, using the technique of *opening* [23], small features in the water map are removed. Without this process, the images will contain small features that can have a large impact on the output field.

To be able to use the technique of opening, two additional methods must be defined, *dilation* and *erosion*. Dilation is the act of making every pixel in the map neigboring to a water pixel into a water pixel, whereas erosion makes every pixel corresponding to water neighboring a pixel designating land into a land pixel. Opening is then defined as an erosion operation followed by a dilation operation. The opposite of opening, *closing*, can be used to emphasize water features in the map and is defined as a dilation operation followed by an erosion operation.

Once the map has been filtered, the boundary can then be extracted. First, a border pixel in the map is defined as a pixel with a value of that has a neighbor with a value of 0. Once such a pixel is found, the next pixel corresponding to a boundary must be extracted. To do this, the neighboring pixels are examined in a clockwise order to find a neighboring pixel that is also a neighboring pixel. Once such a point is found, the point is added to a list and the search continues at the new point. This process continues until either the original pixel is discovered again, or the edge of the image is found. If the edge of the image is found, the algorithm returns to the original pixel and continues the search in the opposite direction.

The extracted boundary is then used to derive an approximation for a polyline, L. The polyline is a collection of line segments. For each line segment $\overline{AB} \in L$, a regular element is constructed at the midpoint of $\overline{AB}$ such that the direction for the element is the direction $\overrightarrow{AB}$. Using this derivation for the boundary edges allows the user to remove unwanted boundaries from the field using the tools in place for design elements.

Fig. 4.3 shows an example of a map made from the tensor field derived from the coastline.

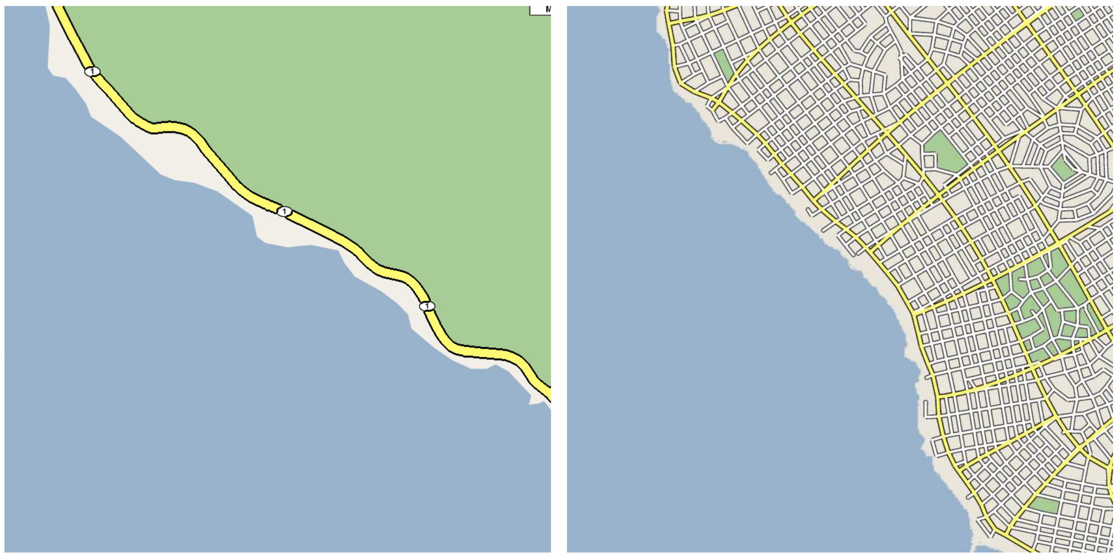


Figure 4.3: (Right) Highway 1 in California, USA follows the coastline. (Left) A street network generated from the boundary field reproduces a similar road.

4.6 Modifying Tensor Fields Using Rotation Fields

So far, the design system so far only describes a system that allows for perpendicular intersections. However, road networks in real cites may contain several intersections that are not perpendicular. A method to handle non-perpendicular road intersections is now presented.

The design system offers three scalar fields, R_1, R_2, and R_3, that the user can design. These three fields are applied as rotational fields. $R_1 \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ rotates the tensor value. $R_2 \in \left[-\frac{\pi}{4}, \frac{\pi}{4}\right]$ and $R_3 \in \left[-\frac{\pi}{4}, \frac{\pi}{4}\right]$ rotate the major and minor eigenvectors respectively.

It is important to note that this is not the only way to attain a street network with intersections that are not perpendicular. Work by Zheng and Pang [76] in the area of *asymmetric tensor field* analysis can also be used to achieve similar results. The approach presented here was found to be simpler to use and implement.

4.6.1 Morse Function Design

To give the user the ability to manually edit the rotation fields, Morse function design is used [44]. In Morse function design, the user specifies a set of points with specific values. A linear set of equations is then solved to find the values at the rest of the points in the edit region. The process to solve this linear set of equations is very similar to the process outlined in section 4.3.

Given a set of points input by the user, S with values s_i and positions p_i, the input points are snapped to the nearest grid point p'_i. The following equation is then solved.

$$f(\mathbf{v_i}) = \sum_{j \in J_i} \omega_{i,j} f(\mathbf{v_j}) \tag{4.14}$$

where J and $\omega_{i,j}$ are defined as before and f is a scalar function for each vertex. Here, the values for $f(p'_i) = s_i$ are held constant.

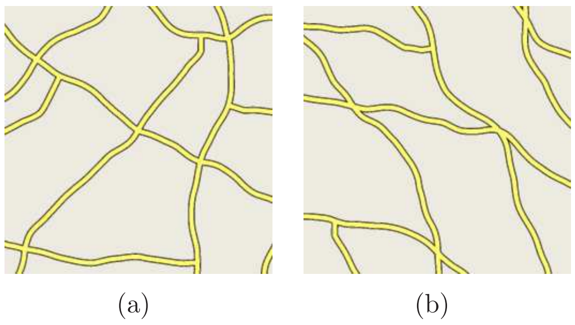


Figure 4.4: This figure shows how the rotation fields can impact the street network. a: $R_2(\mathbf{p}) = 0°$; $R_3(\mathbf{p}) = -20°$. b: $R_2(\mathbf{p}) = 20°$; $R_3(\mathbf{p}) = -20°$

4.6.2 Noise Functions

Another useful way to modify the rotation fields is through the use of functions. In the design system, one type of functions that is applied is Perlin noise [49], which is used to modify R_1. Perlin noise makes a rotation field that gives the appearance of meandering roads, such as those that can be found in mountainous terrain or through parks. Fig. 4.5 is an example of noise being applied to a street network.

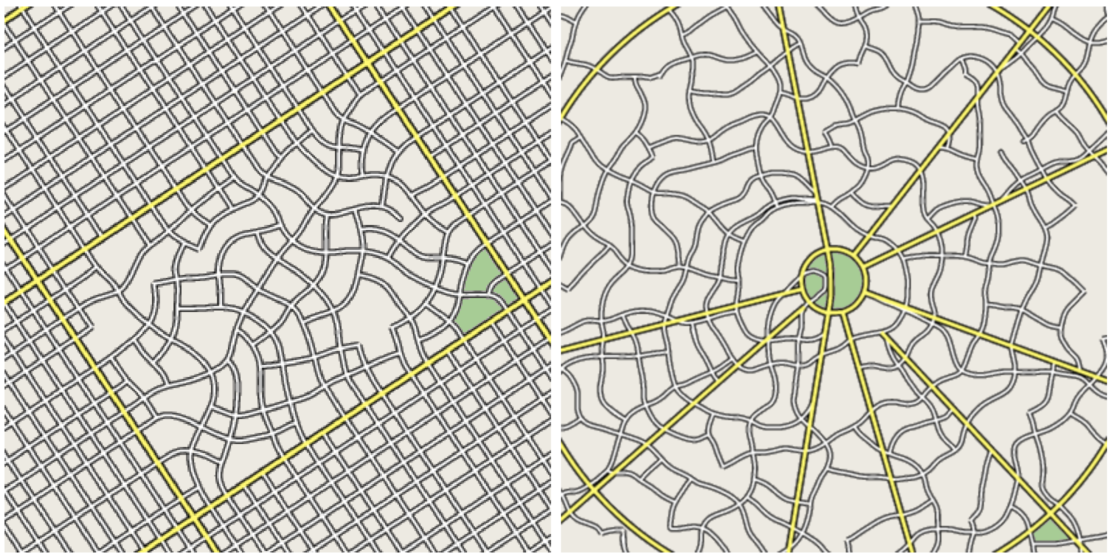


Figure 4.5: Noise produces curved minor roads over major roads. Left: Over a regular major road grid. Right: Over a radial major road grid.

Chapter 5 – Road Extraction

Since designing a tensor field only gives the orientation of the roads and not the roads themselves, the next step is to extract the roads. The algorithm developed for extracting the roads developed through an understanding of the requirements for the roads. From the discussion on road networks, the major road network is less dense and contains few if any dead ends, whereas minor roads may contain many dead ends and may be connected to any road. These different requirements give an indication that different algorithms are needed.

This chapter starts with a discussion on the tracing of the minor roads, and then concludes with the major road algorithm. While tracing actually occurs in the opposite order in the system, this order gives a clearer understanding of the algorithms used.

5.1 Minor Road Tracing

The algorithm used for tracing the minor roads follows closely the evenly spaced streamline algorithm for vector fields [30]. A point in the computational domain is chosen and placed into a queue. This queue contains all the points that are to be tested to see if a hyperstreamline can pass through it. A hyperstreamline can pass through the point if there is not another hyperstreamline within the user specified distance d_{sep}. The algorithm takes a point, $\mathbf{s}$, off Q and uses it to seed the tracing of a single hyperstreamline, l, that is generated. Additional seed points are then pushed onto Q d_{sep} distance away from l on either side.

The algorithm successfully creates a set of hyperstreamlines for a single eigenvector field. To generate the entire set of hyperstreamlines for a road network, first the major eigenvector field is used to trace hyperstreamlines, then the minor eigenvector field. The results are then merged together to form a graph. This graph is then used as the road network.

5.1.1 Tracing Single Hyperstreamlines

Tracing a single hyperstreamline involves selecting an initial direction, $\mathbf{d}_0 = \mathbf{e}_{\mathbf{v_0}}$, where $\mathbf{e}_{\mathbf{v_0}}$ is the eigenvector of the tensor at $\mathbf{v_0}$. This direction is then used to find the next point. After the initial point, the directional ambiguity of tensor fields necessitates the need to determine the direction to be picked for the next integration step. To remove the sign ambiguity, the next integration direction is picked by $\overrightarrow{\mathbf{d}_{i+1}} = \mathbf{e}_{\mathbf{v_{i+1}}} sign(\mathbf{e}_{\mathbf{v_{i+1}}} \cdot \overrightarrow{\mathbf{d}_i})$. The next integration point is then found through an adaptive Runge-Kutta scheme [13]. Tracing stops once a series of conditions are met: 1) it hits the edge of the computation region; 2) a degenerate point is reached; 3) tracing returns to the point at which it started; 4) the hyperstreamline becomes longer than a user specified tracing distance; 5) the streamline violates d_{sep}. Section 5.3 discusses modifications to the ending conditions to allow for the requirements of major roads.

5.1.2 Density Variations

Variations in the density of roads is a common feature in many street networks. One of the input maps, the density map, is a tool for the user to achieve such a variation. The density map is a gray scale image where white is a region of high density and black corresponds to an area of low road density. This map is used as follows:

$$d_{sep}(\mathbf{p}) = d_{low} + m(\mathbf{p}) * (d_{high} - d_{low}) \tag{5.1}$$

where d_{low} and d_{high} are user supplied constants for the lowest and highest densities, m is a function that maps the point $\mathbf{p}$ to the density stored in an image, and d_{sep} is the road separation distance as used in the algorithm above. The density variation affects the maps by making a road terminate sooner as the value of $m(\mathbf{p})$ decreases, or roads to be traced closer together when $m(\mathbf{p})$ increases. This gives the user the flexibility to create many city centers in a single map.

5.1.3 Seed Placement

Seed point placement is determined through the analysis of the various input maps. Common features of road networks are for a street to cross a narrow strip of land, be traced through a dense area of town, or traced through a town center. While tracing typically fills the entire computational domain, small features may not have a road pass through them. The only way to guarantee these features will contain roads is to place the seed points of the roads on the feature.

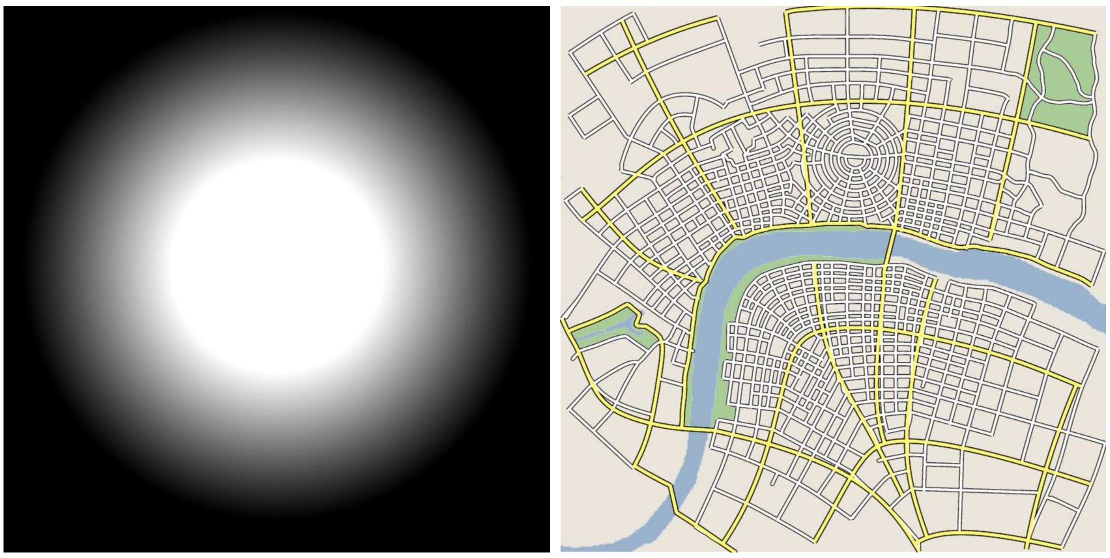


Figure 5.1: Left: grayscale density map generated by the user. Right: the resulting street network.

To determine which seed point needs to be selected first, they are assigned priorities and placed onto a priority queue. Seeds are created along the boundaries extracted from the water map and randomly throughout the computation domain. They are assigned weights based upon the function $\omega(s) = (1 - w(\mathbf{s}))(e^{-d_b} + e^{-d_s} + e^{-d_p})$ where d_b is the distance of s to the nearest boundary, d_s is the distance to the nearest singularity, d_p is the distance to the nearest local maximum in the population density map, and $w(s)$ is a binary function that returns whether the seed s is in water or on land. In practice, the seeds that are in water are discarded.

5.2 Finding Intersections

The final step in creating a street network is to create a graph from the hyperstreamlines. A simple brute force pair matching technique was used. To help increase the speed of the algorithm, each segment was assigned to a cell in the computational domain and then only segments within the same cell are compared against each other.

While reducing the size of the problem does increase its speed, the algorithm is not considered more efficient. In both cases, the algorithm runs in $\mathbf{O}(n^2)$ time. There are algorithms with a better runtime. A nice feature of extracting hyperstreamlines in this fashion is that the a set of hyperstreamlines does not have any self intersections and some algorithms can take advantage of this [9]. As the reduced brute force method produced reasonable results, alternative algorithms were not pursued farther.

5.3 Tracing Major Roads

One unfortunate side effect of extracting roads from the major and minor eigenvector field independently is the collected roads are disjointed and there is no guarantee that the set of streamlines will even intersect. This makes the algorithm insufficient for the major roads, as they must be connected and roads must terminate at another road. The proposed method to handle this is to alternately trace hyperstreamlines in the major and minor eigenvector fields.

5.3.1 Alternating Tracing

The requirement on the algorithm to connect the roads together leads to a unique derivation of the original algorithm. The tracing algorithm first starts as before by selecting a seed point from a set of seed points and placing it onto a queue Q_M. Q_M is a queue containing seed points to be used to trace hyperstreamlines in the major eigenvector field.

The algorithm performs its tracing by taking a seed off Q_M and tracing a single hyperstreamline in the major direction. Seed points are then generated along the hyperstreamline at intervals of d_{sep} and placed into a queue Q_m. Q_m is a queue containing the set seed points to be used to trace hyperstreamlines in the minor eigenvector field. As with the major hyperstreamlines, seeds are placed into Q_M at intervals of d_{sep} along the newly traced minor hyperstreamline. A seed is then taken off Q_M and the process is continued until all seeds in Q_M and Q_m are used.

5.3.2 Termination Conditions

By basing the seed points of one field on the hyperstreamlines of the other field, the sets of hyperstreamlines are guaranteed to intersect. Unfortunately, there is no guarantee that the map will contain few dead ends. To ensure that there are no dead ends except at the edges of the edit region, the termination condition is also modified. The termination conditions from the previous algorithm are used, except that when the hyperstreamlines are within d_{sep} of another streamline, the hyperstreamline is tagged as a zombie line. This zombie line will then continue to trace until it is found to intersect a hyperstreamline in the opposite direction.

Another modification to the termination conditions is the occurance when it hits water. Major roads may contain bridges, which would allow a road to continue tracing over a narrow segment of water. When tracing the hyperstreamline enters water, the following takes place. First, θ, the angle of intersection with the water boundary, is determined. If $\theta < \theta_b$, where θ_b is a user specified angle corresponding to the maximum angle of incidence for a bridge, the tracing is allowed to continue in a straight path for a user specified maximum bridge length, l_b. If the road finds land again during this tracing, the road continues and the previously inserted segments are flagged as a bridge. If no land is found, then the tracing terminates and the extended section is removed.

Fig. 5.2 shows how using both the alternative tracing and modified ending conditions can improve the connectivity of the final road network over the method used to trace the minor roads.

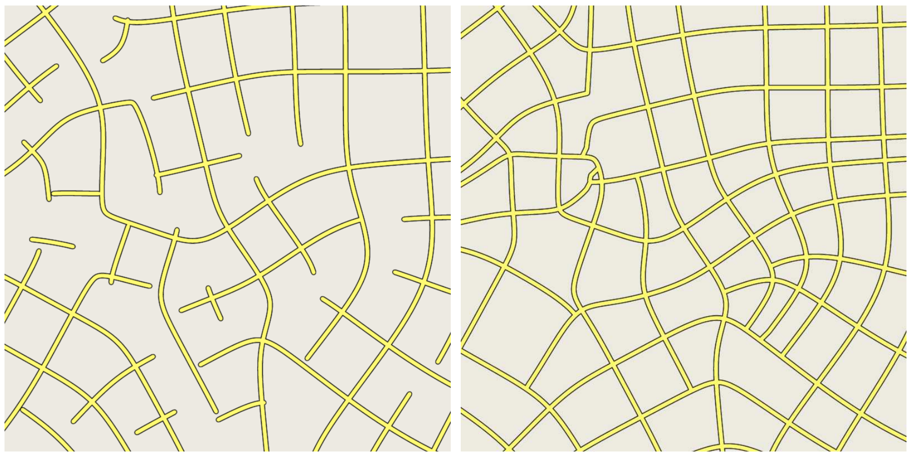


Figure 5.2: Example comparison of Jobard's method [30] on left and the one outlined in this paper (right).

5.3.3 Minor Roads from Major Roads

One feature of a city map is that minor roads can make a discontinuous transition in orientation when they intersect a major road. To handle these transitions, the street design system allows the user to segment the computational domain into several small regions based on the major roads. This is a two step process: 1) The blocks are extracted from the major road network, 2) the set of vertices inside each block is found.

To extract the blocks from the road network, the road network graph edges are made into two half edges. These half edges are then placed onto a queue. These edges are then removed one at a time. When an edge is pulled off the queue that has not been assigned to a block, a new block is generated and the half edge is assigned to the block. The next step is to find the edges of the block. This is done by following the half edges in a forward direction until an intersection is encountered. In this design system, there may be two or three edges to choose from. In general, there are an unknown number of choices. To determine the next edge to choose from, the edges are stepped over, finding the one that has the lowest angle between it and the incoming edge. This process continues until the original edge is found again. Dead ends are easily detected and handled by finding a vertex in the graph with only a single incident edge. In this case, the next half edge is just the sibling of the current edge.

Once the blocks have been extracted, the set of vertices inside the blocks must be found. To do this, a rendering process is used. The first step is to clear the framebuffer to black, and then using a specified color, render into the framebuffer polygons formed by the boundary of the water map. Each block in the graph is then assigned a unique color and then rendered into a framebuffer. The framebuffer is then used by each vertex to extract the color that was assigned to the blocks. This vertex is then assigned to the block as well.

Finally, when an editing operation is performed, the framebuffer is used to determine which block the element covers. The set of vertices to be used is then looked up based on the block and the set is then used as the input to the editing operations discussed in chapter 4.

Due to the large number of segments, it's necessary to give the user a tool that can copy a single tensor element across many regions. The system provides a "sweep" tool that allows the user to pull an element across many regions, copying the value. The effects of the element in the old region are maintained, giving the appearance that the element was copied. Since the element is not maintained in the old region, tensor data is not maintained when further edits affect the region.

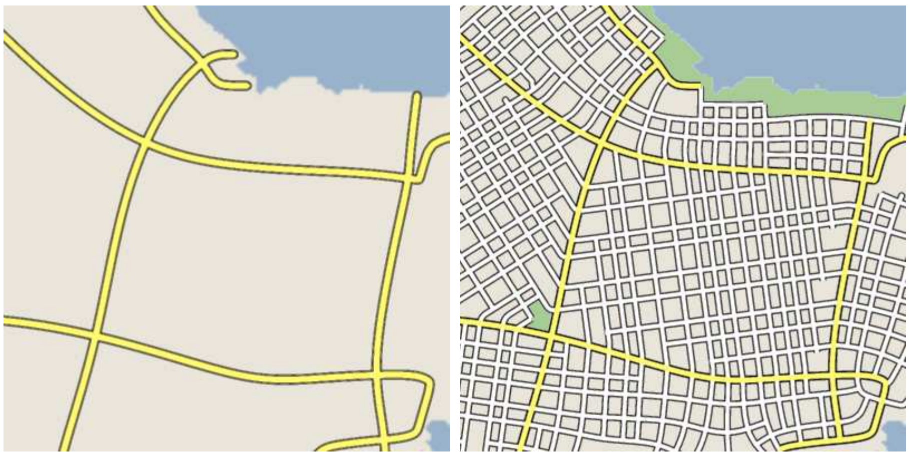


Figure 5.3: The major roads (left) are used to segment the editing domain for the minor roads (right).

Chapter 6 – Street Graph Editing

Once the street graph has been generated, the user may wish to continue to edit the road network to make final adjustments. The system offers several tools to do this, including graph manipulation, manual creation of roads, and local editing. Of major note is the local editing process. Through local editing, the design system is able to achieve *upstream editing*. Upstream editing is the process of taking an existing street network and editing it using the complete design system. This chapter is devoted to highlighting these parts of the system.

6.1 Graph Manipulation

The user is given three tools that directly affects the street graph itself.

- **Manual Manipulation** - The edges and nodes of the street graph may be manually edited through drag and drop operations.
- **Procedural Jittering** - Perlin noise [49] is used to jitter the nodes of a graph. Fig. 6.1 shows an example of this.
- **Algorithm Noise** - Due to the nature of the tracing algorithms, noise is inserted into the street graph through stochastic sampling using Halton sequences [50]. While it may seem important to remove such noise, it is vital

to the appearance of a city and can be seen in almost any city map. Fig. 6.2 shows examples of this noise from Manhattan in New York City, New York and Denver, Colorado.

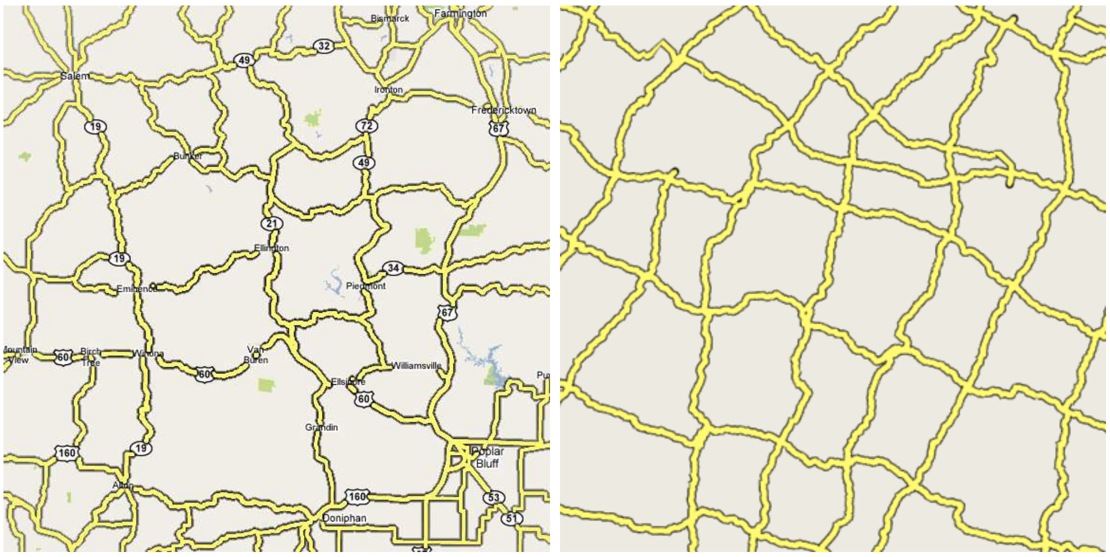


Figure 6.1: Crack patterns in Missouri are generated using road jittering in the system.

6.2 Road Overlays

Some roads are extremely difficult to model using any technique using global patterns. These roads do not follow the typical rules discussed previously and may have an unusual beginning and end. Roads which do not follow a set pattern are handled by allowing the user to specify individual roads. These roads are then treated as polylines and merged into the street graph by determining the intersections of the graph with the polyline. Fig. 6.3 shows such an example.

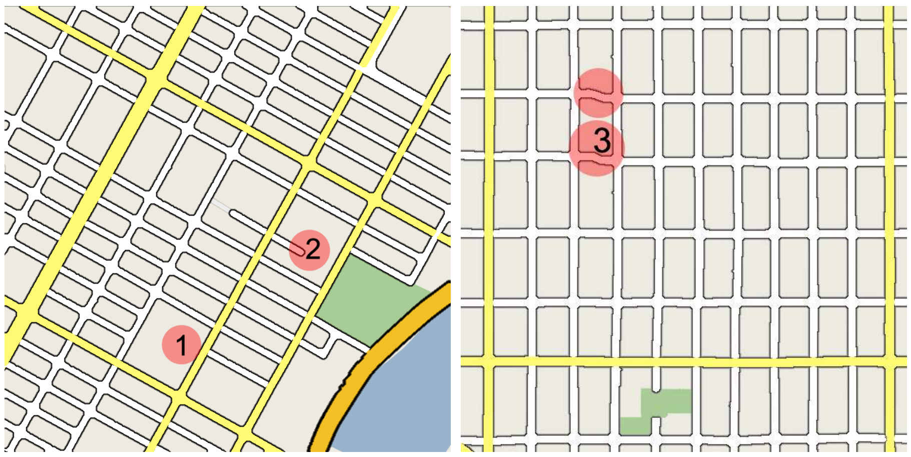


Figure 6.2: Sometimes cells are merged (1) or partially split by dead ends (2). Regular grids sometimes suffer slight shifts (3).

6.3 Local Editing Using Tensor Fields

One of the design goals of the system was to allow the user to edit a map that was previously generated without destroying the entire graph. This is made especially important when the user wishes to edit a road network derived from another process or where the original edit information was lost. Local editing also allows the road network to be constructed incrementally and makes inserting parks simple. To perform local editing, several steps have been identified: removal of roads within a user specified region, design of a local tensor field, and tracing of a new road network.

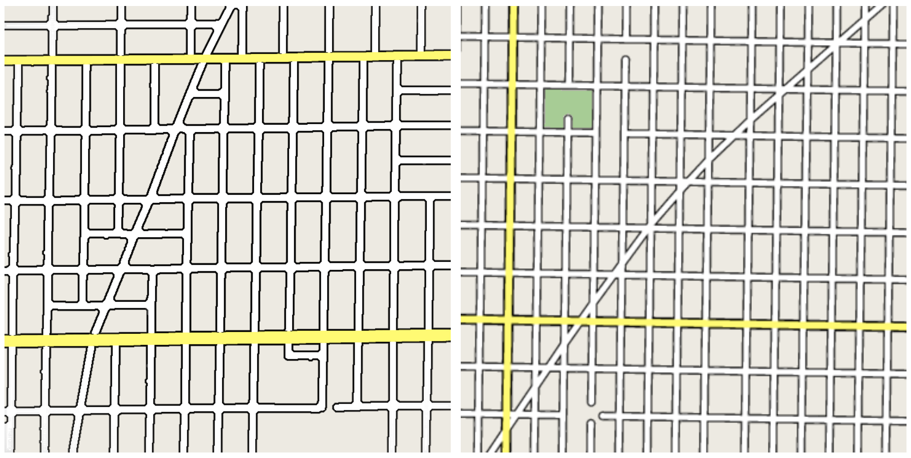


Figure 6.3: A section of Chicago, Illinois shows a street that does not follow the underlaying grid pattern. A road is manually inserted into the graph to simulate this in the design system.

6.3.1 Removal of Roads

Making a local edit to the road network begins with the user specifying the region to edit, D. After the user specifies a region, roads inside the region must be removed. The first step to this is labeling graph nodes as inside or outside. This is done using a simple point inside polygon test.

Once all points have been labeled, those segments with either node inside the region are completely removed. The rest of the graph is left untouched. While splitting the roads along the boundary would seemingly be the more logical choice, odd irregularities tend to occur due to a mismatch in the road density difference in the old road network and the one being created.

6.3.2 Tensor Field Editing

Creating a new field in the region is a trivial task, as all the tools are already in place. Similar to editing on the segmented region, the input to the tools is the region the user has selected to edit over.

6.3.3 Tracing New Roads

Road tracing occurs only on the minor road network. This restriction was a design choice, due to the need for major roads to form a graph containing only cycles. This restriction would make attempting to connect the roads to a pre-existing network sufficiently hard.

To trace the roads, the algorithm for extracting the minor roads is used. As the hyperstreamlines are traced, some of them will leave the editing region. When this occurs, they are marked as a zombie hyperstreamline. They then continue to travel straight for a user specified distance until they intersect a road of the old street network. Once a zombie hyperstreamline stops tracing, the nearest node in the old graph within a user specified angle of θ with respect to the travel direction and the node of the last non-zombie point is found as well. The new hyperstreamline is then connected to the nearer of the points. Note that if no intersection is made during the tracing, the zombie line is just connected to the nearest node.

Fig. 6.4 shows an example where a park is inserted into the road network by removing streets previously generated.

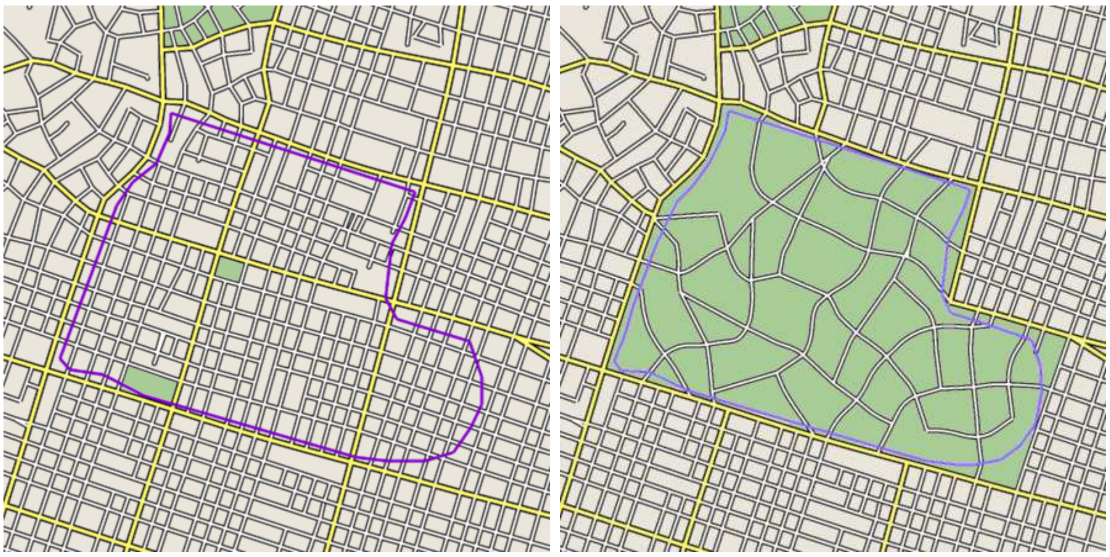


Figure 6.4: A park can be inserted using local editing by removing streets in the region where a park is to be, then tracing lower density roads.

Chapter 7 – Road Network Modeling Results

The street network system allows the user to generate systems of roads which reflect the appearance of real street networks. These results were created in a reasonable amount of time which left plenty of time to test out various street patterns. The street network shown in Fig. 7.1 was created in under five minutes. As can be clearly seen, the results show a street network can be generated that exhibits patterns found in many street networks.

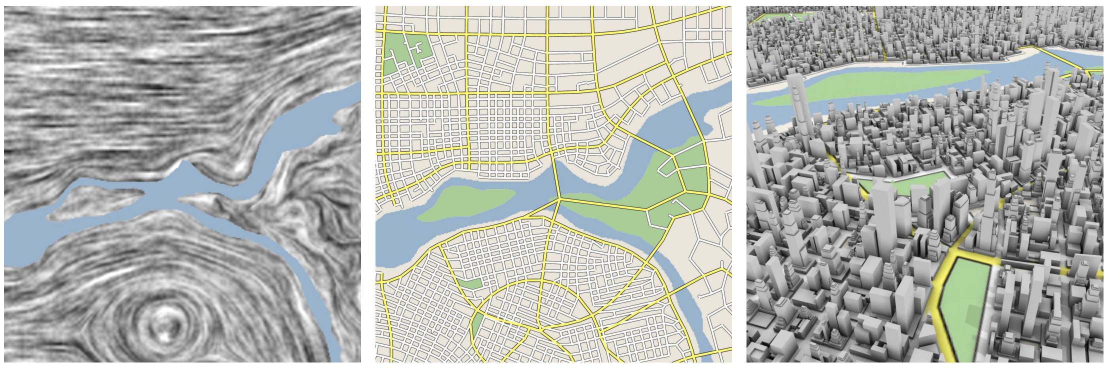


Figure 7.1: A virtually generated street graph with the water map based on a stretch of the Benue River in Nigeria.

The street networks in Fig. 7.2 and Fig. 7.3 were each created in under an hour. Most of the hour was spent trying different layouts and performing minor tweaks to the street graph. Fig. 7.4 is provided as a way to compare the system presented here against the results of previous systems. Finally, some screen captures from a video of a flythrough of the city shown in Fig. 7.1 are shown in Fig. 7.5. Only the road network of the image is generated in the system. The buildings were created using CityEngine [53]. After the results of the street modeling system are presented, the next chapter starts the second part of this thesis: bridge analysis and visualization.

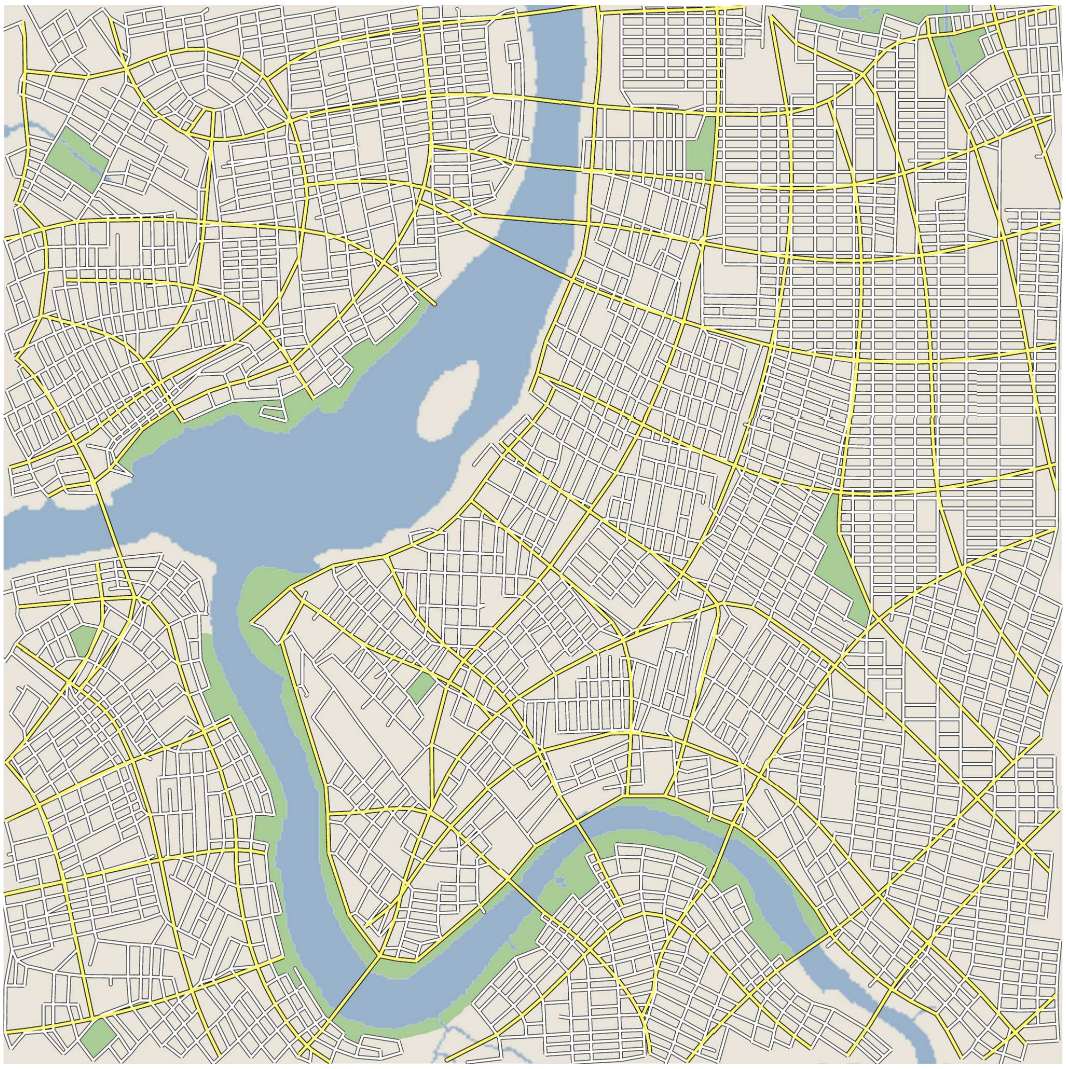


Figure 7.2: A generated street graph based on a water map from downtown Taipei, Taiwan.

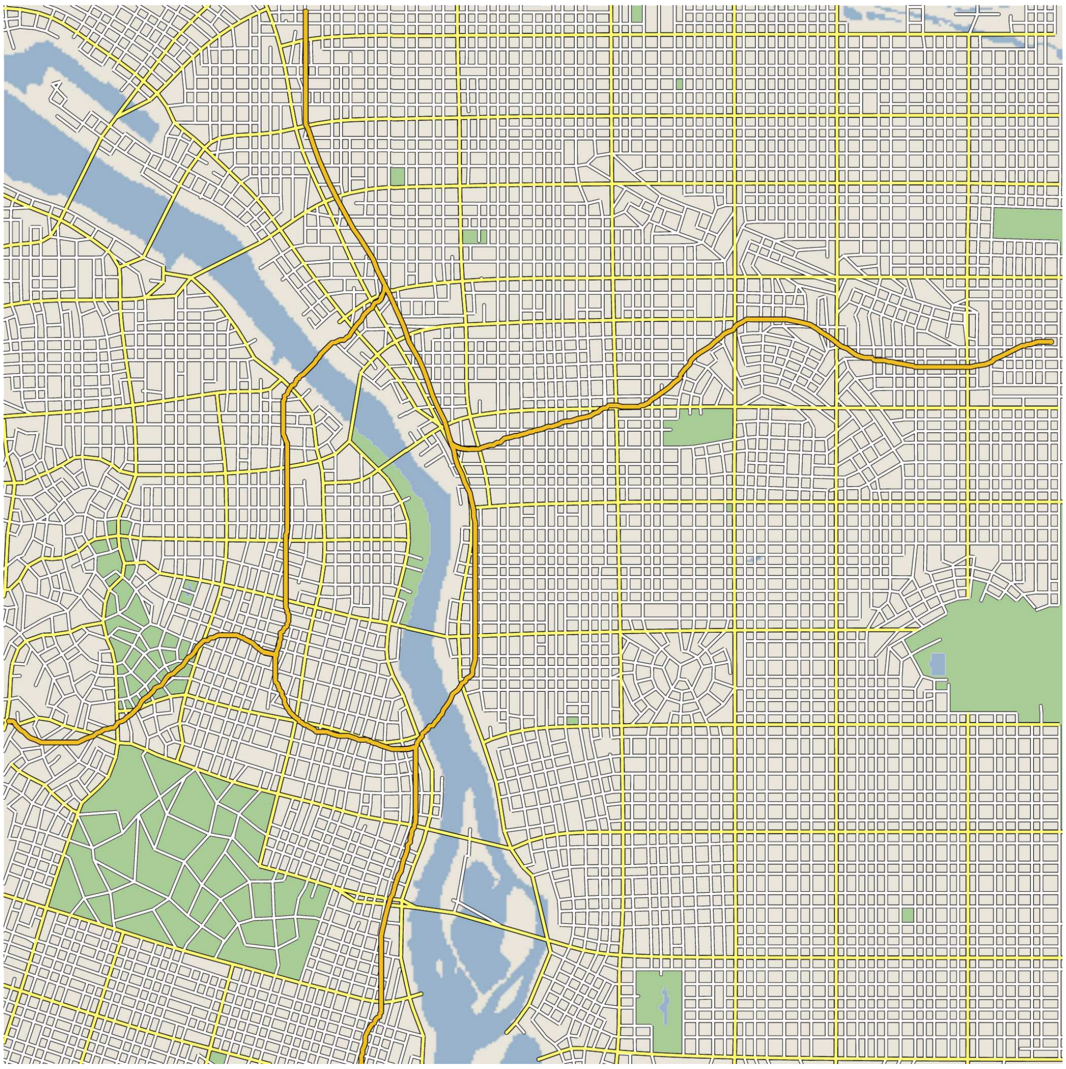


Figure 7.3: Hand drawn highways (in orange) on top of a generated street network give the map of Portland, OR, USA a more authentic feel.

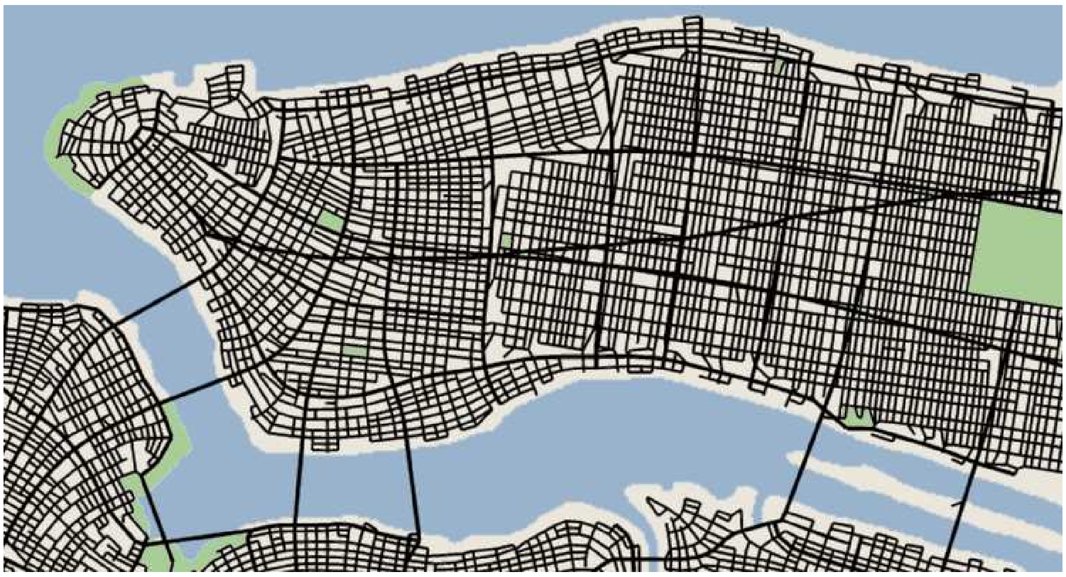


Figure 7.4: A generated map of Manhattan, NY, USA. This was designed using the system described here.

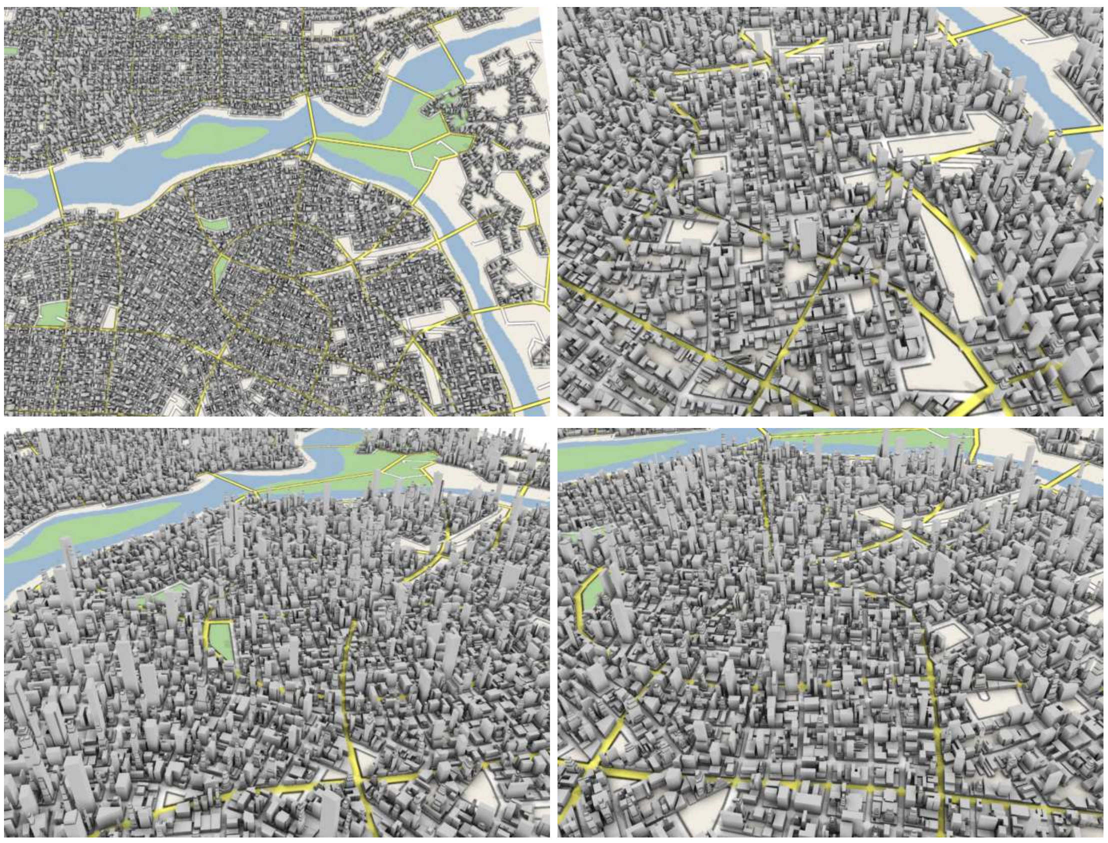


Figure 7.5: Frames from a flyover of the virtual city in Figure 7.1 shown with generated building geometry.

Chapter 8 – Bridge Visualization Overview

One of the main aspects of civil engineering is analysis of existing infrastructure to test for the need for repair or replacement. Road infrastructure poses a unique challenge as aging roads, bridges, and supports demands a large number of man-hours to be able to analyze. To reduce the amount of time an engineer needs to compute these results, a software system has been developed to carry out the analysis and display the results to bridge engineers or people using the system. This becomes increasingly important as vital infrastructure ages and the number of structures that need to be rehabilitated increases. This part of the thesis covers the visualization of a scalar field that indicates the probability of failure over the structure of a specific type of bridge.

8.1 Bridge Background

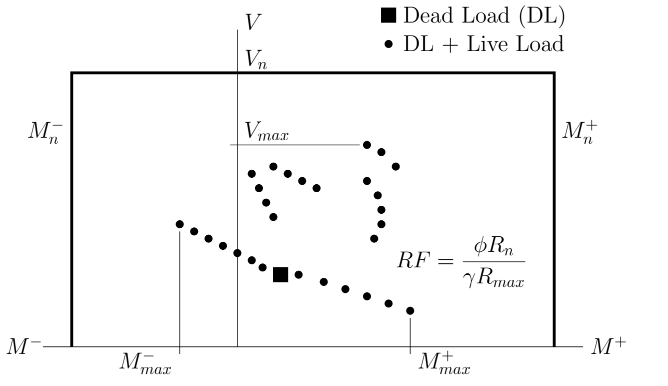
The procedure analyzes a single type of bridge, a conventionally reinforced concrete deck-girder (RCDG) bridge, which is common throughout the country and was used heavily during the interstate boom of the 1950's. Major characteristics of the bridge include tapered and haunched concrete girders, a deck, and bents.

Due to increasing volume and magnitude of truck loading, many of these bridges are in need of analysis to determine if their current weight restrictions are appropriate for modern day needs or what areas of the bridge need attention. This process can take a bridge engineer several days to complete, and the results are often cryptic and meaningful only to the engineer.

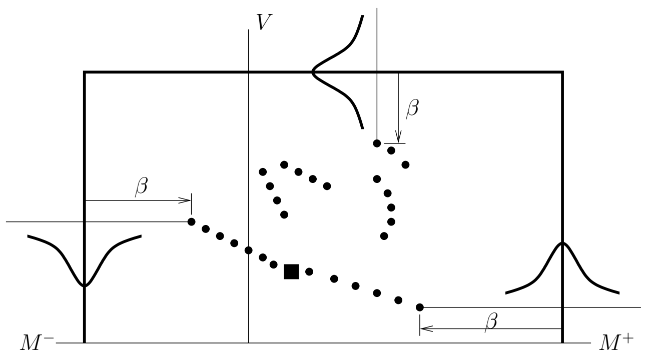
Both these pieces of information can be found through a single analysis [26]. This analysis finds the probability that a bridge will be unable to withstand the weight of a truck at specified points. Furthermore, it can also be used to determine those sections of the bridge which need further examination. This allows bridge engineers to concentrate on specific areas of the physical bridge when performing inspections. While the analysis produces the results being visualized, the analysis is not a contribution of this work.

Fig. 8.1 shows the basic premise behind the evaluation method, which is based on moment-shear interaction of reinforced concrete girders. The scalar field being visualized in the system is β, which is defined as the number of standard deviations the peak moment-shear load effect is from the nominal capacity curve [26]. The capacity curve is calculated based upon a statistical model of combined moment-shear force effects in reinforced concrete girders [65].

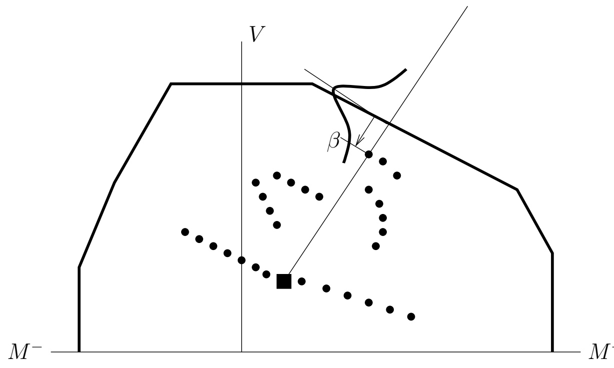
The most important aspect of the bridge analysis is being able to convey to people without an engineering background information on the analysis, such as the location in need of repair, or the regions that would be prone to permanent damage should an overloaded truck cross it. This allows an engineer to be more effective in communicating with a politician or community organizer to get the funds needed to perform necessary repairs.



(a) Load and Resistance Factor Rating (LRFR) for single force effect



(b) Reliability-based approach for single force effect

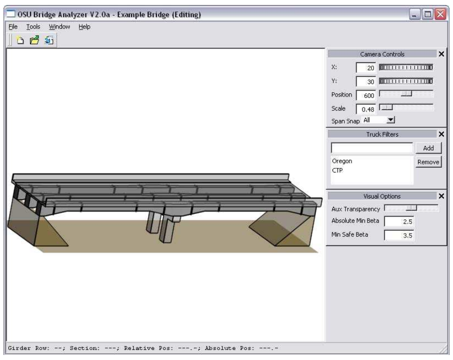


(c) Reliability-based approach for combined force effect

Figure 8.1: Methods for evaluating bridge girder ratings. The curve represents the moment-shear capacity of a cross section. M: moment reaction; V: shear reactions.

8.2 System Overview

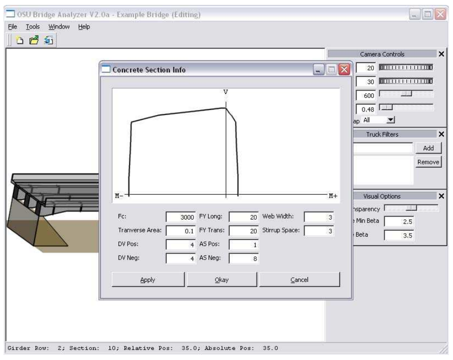
Fig. 8.2 shows a workflow for the bridge analysis system. The bridge analysis system has a three stage process. The first stage loads a bridge from a descriptor file and presents the bridge to the user for a chance to review and edit bridge parameters. The bridge file contains both geometric and material information on the bridge. The second stage of the process is the analysis step. The analysis is non-interactive and is performed without the need of user intervention. Upon completion of the analysis, the data collected during the analysis is presented to the user. These results can be exported and presented.



X: 20
Y: 30
Position 600
Scale 0.48
Span Snap All
Oregon
CTP
Aux Transparency
Absolute Min Beta 2.5
Min Safe Beta 3.5

Girder Row: --; Section: ---; Relative Pos: ---.-; Absolute Pos: ---.-

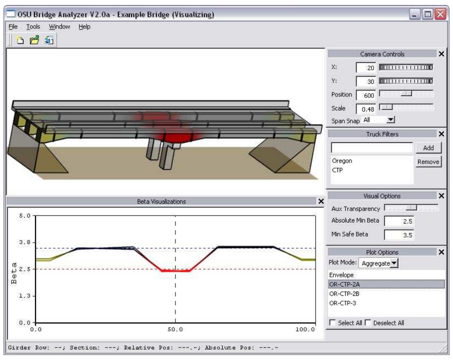
(a) Input bridge properties



Fc: 3000
FY Long: 20
Web Width: 3
Tranverse Area: 0.1
FY Trans: 20
Stirrup Space: 3
DV Pos: 4
AS Pos: 1
DV Neg: 4
AS Neg: 8

Girder Row: 2; Section: 10; Relative Pos: 35.0; Absolute Pos: 35.0

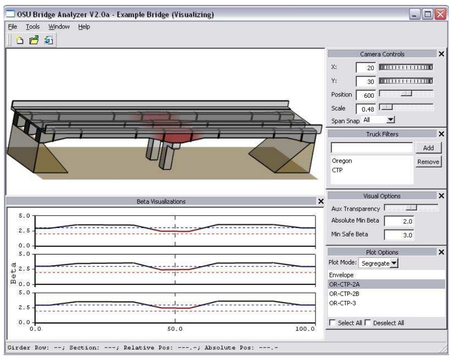
(b) Modify section properties



X: 20
Y: 30
Position 600
Scale 0.48
Span Snap All
Oregon
CTP
Aux Transparency
Absolute Min Beta 2.5
Min Safe Beta 3.5
Plot Mode: Aggregate
Envelope
OR-CTP-2A
OR-CTP-2B
OR-CTP-3

Girder Row: --; Section: ---; Relative Pos: ---.-; Absolute Pos: ---.-

(c) Visualize rating (aggregate)



X: 20
Y: 30
Position 600
Scale 0.48
Span Snap All
Oregon
CTP
Aux Transparency
Absolute Min Beta 2.0
Min Safe Beta 3.0
Plot Mode: Segregate
Envelope
OR-CTP-2A
OR-CTP-2B
OR-CTP-3

Girder Row: --; Section: ---; Relative Pos: ---.-; Absolute Pos: ---.-

(d) Visualize rating (segregate)

Figure 8.2: An example user workflow: (a) the bridge is loaded into the system; (b) modification can be made to critical sections; (c) and (d) the ratings are visualized.

Chapter 9 – Bridge Visualization

The first step in the bridge analysis system is the ability to visualize and review the bridge, which is covered in this. The visualization of the bridge also occurs throughout the analysis system and is vital to give the user an intuitive sense of where the analysis has found an area that an engineer should concentrate. This chapter discusses how the bridge mesh is generated, and the techniques used to visualize the bridge.

9.1 Bridge Mesh Generation

The base of the bridge visualization system is the mesh of the bridge. The mesh is generated procedurally from a user generated input file. A short example file is shown in Appendix A. The file specifies the dimensions of various components, from the deck to the bents. The system uses such information to first create the girders, then the deck and bents. While the deck and bents are less important for the analysis of the bridge, they provide an important visual cue to the user for the position of the span separations and the orientation of the bridge. As such, it is important that they represent a real world bridge.

Dimensional information on the girders includes the length and the cross-section dimensions, width and height, of the girder at various points along its length.

The width and height are stored in dimensional section. The girders are created independently of the girder elements surrounding it. The first step is to sort the dimension sections based upon the distance to the beginning of the girder element. An end cap is placed at the beginning of the girder element which extends the dimensional information of the girder to the beginning of the element. Rectangular tubes are then created for each pair of dimension sections. On the other end, an end cap is created which extends the dimensions of the last tube over the remaining length of the element.

Information on the deck is mostly derived from the values of the girder. The deck is assumed to cover the entire length and width of the bridge. Additional information provided by the user is the distance from the center of the outside girders to the curb and railing, along with the thickness of the deck. Some of the elements of the deck that are unimportant, or redundant, are assumed to be a constant value. Two examples are the height of the curb and the thickness of the railing, both of which are set to four inches. The dimensional information is used to create a flat deck with sidewalks and solid rails.

The final structural component of the bridge is the bents. The bents are the structural components that carry load from the girders to the foundation. They are assumed to be a two column bent, with the distance between the columns being a quarter of the width of the bent. Two column bents are common in many bridges of this type, although they typically take a different appearance than that represented in the system.

Fig. 9.1 shows the underlaying mesh of a bridge which has been generated in the bridge analysis system.

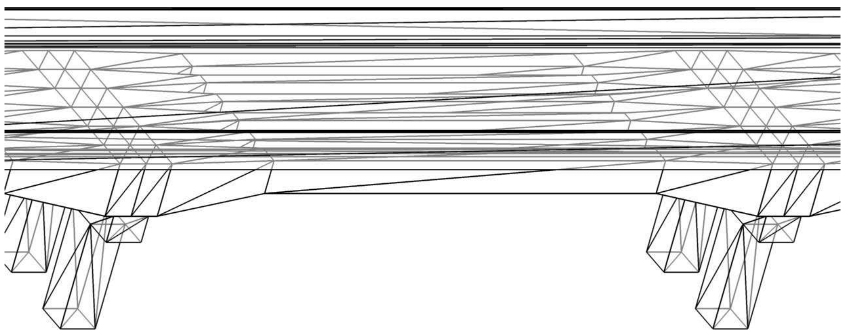


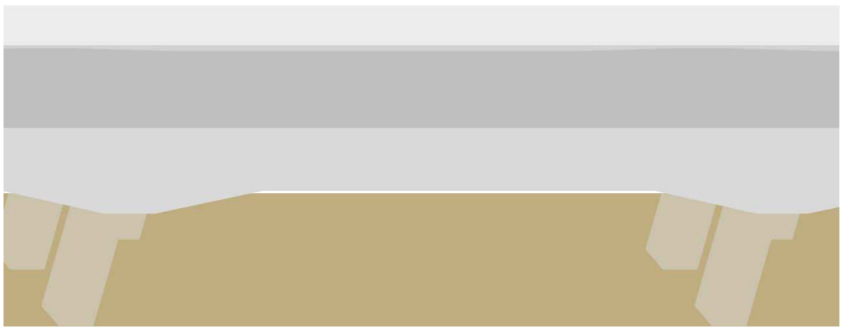
Figure 9.1: The underlaying triangular mesh of the bridge.

9.2 Lighting

Lighting provides important cues to the viewer on both the orientation and the geometric variations in the shape object. Without lighting, the bridge looks flat, as shown in Fig. 9.2(a). Fig. 9.2(b) shows how lighting can be used to enhance the visual appearance of a bridge. Without lighting, changes in the orientation of the face normals would be difficult to distinguish. Also, lighting makes such features as the girder spacings and other geometric variations more obvious. Multiple lights can be used to highlight the orientation of the bridge.

9.2.1 Lights

The system uses two lights, one that is tied to the orientation of the camera, and the other that is tied to the orientation of the bridge. The light tied to the bridge


(a) Without lights looks flat

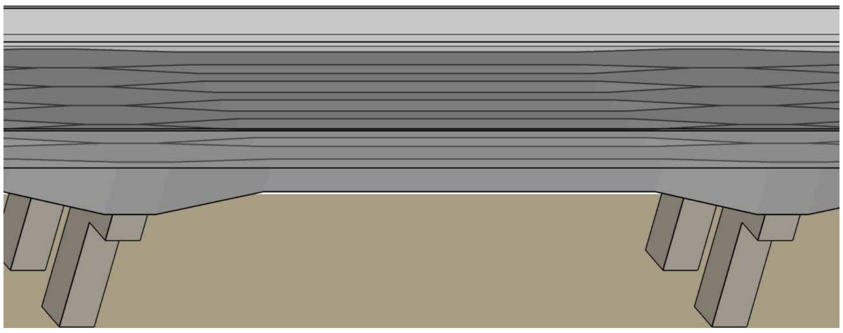

(b) Lighting and highlighting exposes features

Figure 9.2: The selection of the right rendering techniques can vastly improve the visual appearance of a bridge.

gives the user a more intuitive sense on the orientation of the bridge and highlights the tapers and haunches of the bridge more clearly. The light tied to the camera gives the user a better sense of shape for the bridge, and also allows the user to view specific points that may otherwise be hidden in shadow. Both lights are directional lights that are set to a shade of gray. It is important that the light be gray to ensure that the color of the light does not cause oversaturate the output image.

It is important to note that this lighting scheme produces similar results as toon shading [34]. This is intentional as toon shading can be used to give an illustrated appearance. The use of flat lighting without mapping to a shade can be made since the surfaces are prismatic.

9.2.2 Materials

An important aspect giving the user the ability to understand the structure of the bridge is the color of individual structural components. In the system, gray was used to represent parts of the bridge that are concrete, such as the deck, bents, and girders. The parts of the bridge visualization that are supposed to represent the ground, such as the end embankments and the ground beneath the bridge are colored brown. This was chosen as the structures under the bridge are unknown and can be a number of things, including water, another road, or vegetation. Finally, the sections which the user marks to be analyzed are marked blue, to appear less natural and stand out to the user.

The specular effects of the bridge components are disabled since all the components of the bridge have flat surfaces and made from materials that do not have specular effects. If enabled, specular effects would cause surfaces to flash as the bridge is rotated.

9.3 Feature Edge Highlighting

While lighting provides more detail to the user, feature edge highlighting can be used to further convey shape information. Many of the edges on the bridge are sharp. Highlighting the sharp edges gives an indication to the user on the shape and structure of the bridge. Another important feature is to highlight the edges of the bridge image, which are also known as the silhouette edges. The methods used to find both sets of edges are discussed in the next sections.

9.3.1 Sharp Edges

To determine which edges are considered sharp, one must first determine what it means to be sharp. Here, a sharp edge is a place on the model where the curvature changes drastically. This follows from the detection of feature edges for smooth faces as discussed in [46]. An observation for prismatic shapes leads to an optimization. The curvature over a prismatic surface is constant at zero except at the edges, where the curvature experiences an instantaneous change. The edges which can be considered sharp are defined as those edge where the angle between the two neighboring faces normals, κ are greater than some angle.

9.3.2 Silhouette Edges

The silhouette edges are defined as the edges that form the border of the image formed by projecting the bridge mesh onto the view plane. Furthermore, this can be determined by finding the place where the normal of the surface crosses from forward facing to back facing. In a prismatic object, this can only happen on the edges. A face is considered front facing if the following inequality holds

$$\overrightarrow{e} \cdot \hat{n} \leq 0 \tag{9.1}$$

where e is the eye vector and n is the normal vector of the face.

To determine which edges are considered the silhouette edge, the system runs over all the faces in the mesh and labels each face as front or back facing according to equation 9.1. Once all the faces have been labeled, the system runs through all the edges, marking those edges whose neighboring faces are labeled differently as silhouette edges.

9.3.3 Drawing Highlights

The edges found by the sharp edge and silhouette detection are the same in many cases. For this reason, the algorithms are executed simultaneously to prevent from drawing the same edge multiple times. To draw the edge highlights, the system

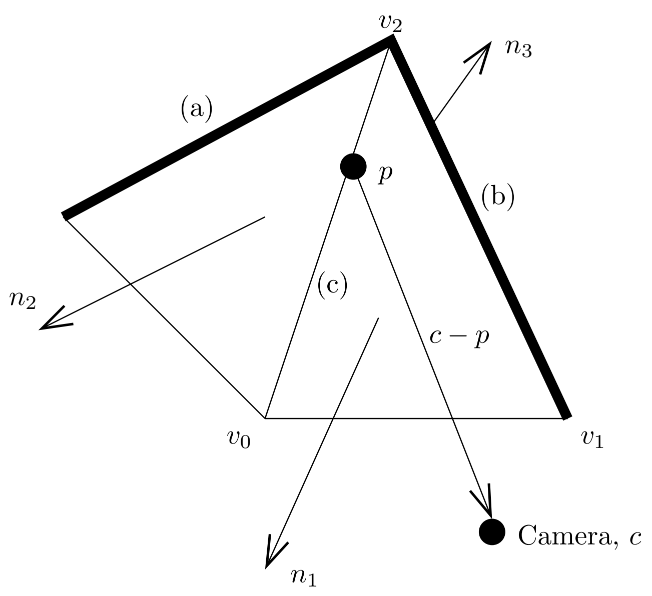


Figure 9.3: A silhouette edge is an edge that is shared by front and back facing polygons, such as edges (a) and (b). Edge (c) is not classified as a silhouette edge as it is shared by two front facing polygons.

first labels the faces as front or back, then cycles through the edges to determine if they are either a sharp edge or a silhouette edge. If they are found to be either a sharp or silhouette edge, they are drawn to the screen. In this way, the edges are cycled over a single time.

9.4 Auxiliary Components

An option given to the user is the ability to adjust the opacity of auxiliary components, such as the deck and the bents; however, the opacity does not affect the feature highlighting. This allows the user to deemphasize the parts of the bridge that are not important for the analysis of the bridge. Keeping the highlights ensures that some visual indication to the overall shape and orientation of the bridge remains intact.

Chapter 10 – Beta Visualization

While the visualization of the bridge gives the user an impression of the shape of the actual bridge, providing a means for the user to view the results of the analysis is the primary function of the system. Upon completion of the bridge analysis, the user must be able to view the results in a way that is both intuitive and meaningful. This chapter discusses the visualization of β values over the length of the bridge.

10.1 Mapping β Values

While a typical scalar visualization system would visualize values over the entire girder using a color encoding to indicate intensity, bridge engineers have indicated they are not interested in specific intensities of the scalar field, but instead prefer highlighting places where the scalar value exceeds a certain value. An optional feature of the visualization system uses color to indicate the mode of failure. In this thesis, the optional features are enabled since the basic version is a simplification where all modes of failure are assigned the same color.

When the user desires to see the mode of failure, the color corresponds the mode of girder failure related to the peak moment-shear demands found during the analysis. As shown in Fig. 10.1, there are five modes of failure. Magenta and cyan respectively relate to failure due to positive and negative bending moments, while yellow indicates failure due to shear. Red and green indicate failure due to a combined shear and negative or positive moment.

10.2 User Thresholds

Fig. 10.1 also shows three color regions for each mode of failure. These three regions relate to thresholds set by the user. The user is able to indicate a value they would consider *safe* and a value they consider *critical*. When the value of β is above the *safe* value, the girder cross section is said to satisfy the inventory rating condition of performing safely indefinitely [1]. This section is indicated by the gray portion of the color wheel. The solid color portion of the graph indicates a β where the girder cross-section does not satisfy the rating condition [1]. These areas indicate a position where the bridge may need immediate repair or weight restriction postings. The sections between the solid color and gray are attained through a linear blending and indicate a region that neither can perform safely indefinitely, nor is in need of immediate attention.

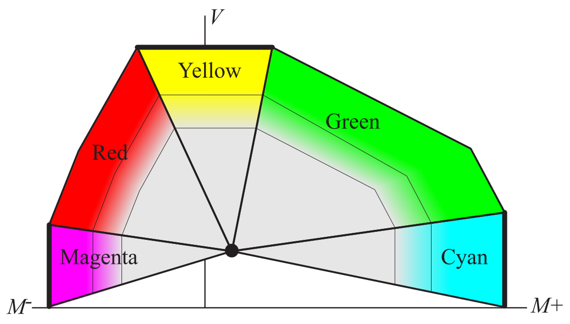


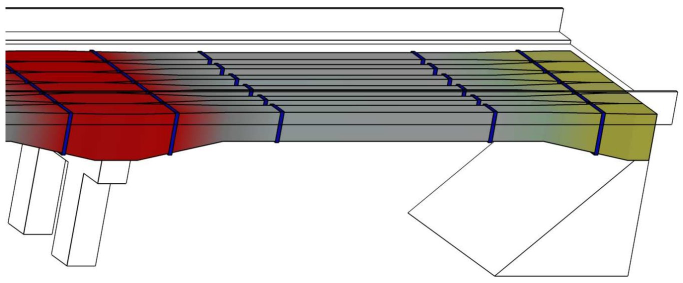
Figure 10.1: The color wheel for the bridge rating. The color is picked based on the mode of failure based on the peak moment-shear reactionary force.

The colors are applied on the bridge in sections. First, the results are retrieved from the analysis at the sections of the girder marked for rating. Since the β plot is global over the entire bridge, values must be transferred from one girder to the next to ensure that the visualization is smooth. The visualization is then processed in a way that is similar to the generation of the mesh for the girders, with one marked difference. The girders are split along the boundaries of the regions to allow for the smooth interpolation of β values in the transition region.

Chapter 11 – Bridge Visualization Results

The bridge visualization system has been developed in close collaboration with bridge engineers at ODOT. They confirmed that the visualization techniques put forth will add value to their current methods of analysis. The mapping of β values onto the bridge was specifically mentioned as an technique to convey bridge health and a need for repairs to lawmakers. Below are shown several screen captures from the bridge analysis software to give the reader an impression of what views are possible. They show how the engineer may view a specific section of the bridge and how the tool may be used to view the bridge at an angle optimal for viewing the visualization.

Beyond the orthogonal views shown, with minor modifications, the visualization system is capable of alternative views as well. Fig. 11.2 shows two views where the user is looking at the bridge in a perspective view and rotated along a third axis. These images also show one possible future extension to the system, augmented reality [31]. This was not one of the goals of the system and was not pursued further.


(a) Hiding auxiliary bridge features of an end span.

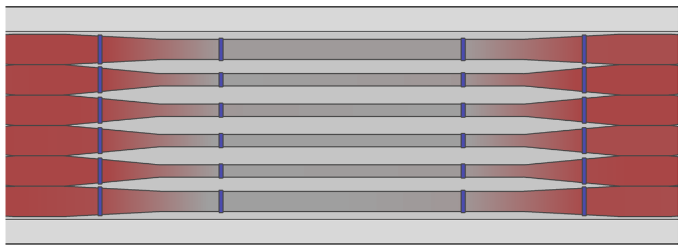
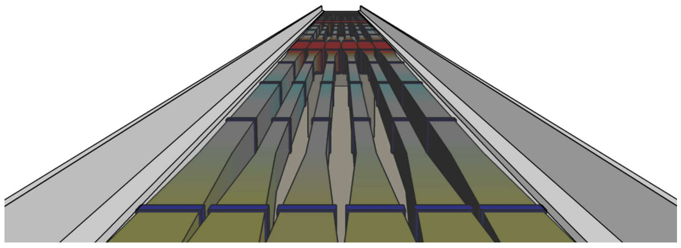
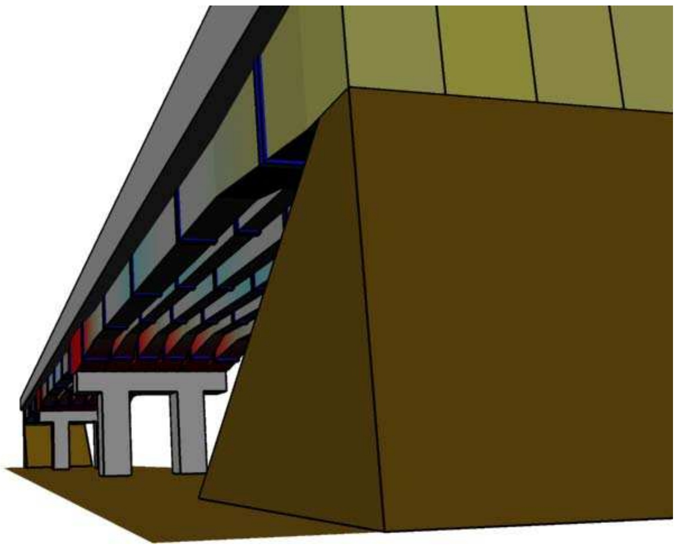

(b) Looking from above the bridge at a center span.

Figure 11.1: Views of a bridge with the rating visualization enabled.



(a) Bridge rating visualization looking down a bridge.



(b) Looking up from the ground at the underside of a bridge

Figure 11.2: Bridge visualization with advanced viewing options enabled.

Chapter 12 – Conclusion

Systems were presented for both the design and analysis of road infrastructure. Preliminary discussions indicate they give the user an intuitive understanding of the problem they are presented to solve. The street network modeling system gives the user a system which can be used to design large road networks and the bridge analysis visualization system allows bridge engineers to more effectively communicate with less technically inclined people.

12.1 Future Work

While the systems presented here both achieved their stated goals, some extensions to the systems can allow them to perform tasks beyond their original purposes.

The street modeling system can be extended to include more road patterns. Currently, the system is able to handle regular grids quite well, but other less regular patterns, such as those seen in European cities, are much more difficult to attain. One way to attain these patterns would be to extend the tracing algorithm for minor roads to break up the regularity. While the system is more intuitive than previous methods, further user studies should be preformed to determine ways the system could be made more intuitive.

The bridge design system can also be extended. The current system is intended for use by those who are analyzing a bridge using information stored in databases. These databases could be updated in real time and the analysis can be performed on the updated data to give the user interactive visualization of the impacts of a specific truck traveling over a bridge. The system can also be used to show an augmented view of the bridge where the visualization is applied over the bridge when viewed through specialized goggles. Furthermore, due to the simplicity of the visualization techniques used, the visualization system can be placed onto mobile devices to give the user access to the information in the field.

12.2 Final Words

The systems presented in this thesis attained their respective goals of allowing the intuitive design and visualization of road infrastructure. Through the use of tensor field design and advanced hyperstreamline tracing, the street network modeling system was able to provide an intuitive design system while remaining powerful and flexible. The bridge visualization system gives the user a system that can convey analysis results directly on a representation of the bridge being analyzed.

Road infrastructure is at the foundation of both these problems. The importance of roads to the survival and character of cities will ensure research in this area will continue into the future.

Bibliography

[1] AASHTO. *Manual for Condition Evaluation and Load and Resistance Factor Rating (LRFR) of Highway Bridges.* American Association of State Highway and Transportation Officials, Washington, D.C., 2003.

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

[3] F. Akgül and D. M. Frangopol. Bridge rating and reliability correlation: Comprehensive study for different bridge types. *Journal of Structural Engineering*, 130(7):1063–1074, 2004.

[4] Christopher Alexander, Sara Ishikawa, and Murray Silverstein. *A Pattern Language: Towns, Buildings, Construction.* Oxford University Press, New York, 1977.

[5] Daniel Aliaga, Carlos Vanegas, and Bedřich Beneš. Interactive example-based urban layout synthesis. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 Papers*, volume 27, New York, NY, USA, 2008. ACM.

[6] Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Lévy, and Mathieu Desbrun. Anisotropic polygonal remeshing. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 485–493, New York, NY, USA, 2003. ACM.

[7] Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Lévy, and Mathieu Desbrun. Anisotropic polygonal remeshing. *ACM Transactions on Graphics*, 22(3):485–493, 2003.

[8] Edward Angel. *Interactive computer graphics: a top-down approach with OpenGL.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1996.

[9] Julien Basch, Leonidas J. Guibas, and G. D. Ramkumar. Reporting red-blue intersections between two sets of connected line segments. In *ESA '96: Proceedings of the Fourth Annual European Symposium on Algorithms*, pages 302–319, London, UK, 1996. Springer-Verlag.

[10] Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry*. Springer-Verlag, 2000.

[11] Transportation Research Board. *Highway Capacity Manual; U.S. Customary Version*. Transportation Research Board, 2000.

[12] Eric Bruneton and Fabrice Neyret. Real-time rendering and editing of vector-based terrains. *Computer Graphics Forum*, 27(2):311–320, April 2008.

[13] J. R. Cash and A. H. Karp. A variable order Runge-Kutta method for initial value problems with rapidly varying right-hand sides. *ACM Transactions on Mathematical Software*, 16:201–222, 1990.

[14] Guoning Chen, Gregory Esch, Peter Wonka, Pascal Müller, and Eugene Zhang. Interactive procedural street modeling. *ACM Trans. Graph.*, 27(3):1–10, 2008.

[15] Guoning Chen, Konstantin Mischaikow, Robert S. Laramee, Pawel Pilarczyk, and Eugene Zhang. Vector field editing and periodic orbit extraction using morse decomposition. *IEEE Transaction on Visualization and Computer Graphics*, 13(1):769–785, 2007.

[16] T. Delmarcelle and L. Hesselink. The Topology of Symmetric, Second-Order Tensor Fields. In *Proceedings IEEE Visualization '94*, pages 140–147, 1994.

[17] S. Dong, S. Kircher, and M. Garland. Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Comput. Aided Geom. Des.*, 22(5):392–423, 2005.

[18] G. Esch, M. Scott, and E. Zhang. Graphical 3d visualization of highway bridge ratings. *Journal of Computing in Civil Engineering*, under review.

[19] Greg Esch, Peter Wonka, Pascal Müller, and Eugene Zhang. Interactive procedural street modeling. In *SIGGRAPH '07: ACM SIGGRAPH 2007 sketches*, page 35, New York, NY, USA, 2007. ACM.

[20] Matthew Fisher, Peter Schröder, Mathieu Desbrun, and Hugues Hoppe. Design of tangent vector fields. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 56, New York, NY, USA, 2007. ACM.

[21] R. Gingroz, R. Robinson, D. K. Carter, Barry J. Long Jr., and Paul Ostergaard. *The Architectural Pattern Book: A Tool for Building Great Neighborhoods.* W. W. Norton & Company, 2004.

[22] Kevin R. Glass, Chantelle Morkel, and Shaun D. Bangay. 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*, pages 161–169. ACM Press, 2006.

[23] R. M. Haralick, S. R. Sternberg, and X. Zhuang. Image analysis using mathematical morphology. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(4):532–550, 1987.

[24] Alejo Hausner. Simulating decorative mosaics. In *SIGGRAPH Proceedings*, pages 573–580, 2001.

[25] Aaron Hertzmann and Dennis Zorin. Illustrating smooth surfaces. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH 2000)*, pages 517–526, August 2000.

[26] C. Higgins, T. K. Daniels, D. V. Rosowsky, T. H. Miller, and S. C. Yim. Assessment and risk ranking of conventionally reinforced concrete bridges for shear. *Journal of the Transportation Research Board*, 1928:110–117, 2005.

[27] C. Higgins, T. H. Miller, D. V. Rosowsky, S. C. Yim, T. Potisuk, T. K. Daniels, B. S. Nicholas, M. J. Robelo, A.-Y. Lee, and R. W. Forrest. Reliability based assessment methodology for diagonally cracked conventionally reinforced concrete deck girder bridges: An integrated approach. Technical Report SPR 350, Oregon Department of Transportation, Salem, OR, 2004.

[28] B. Hillier. Cities as movement economies. In *Urban Design International*, pages 41–60, 1996.

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

[30] B. Jobard and W. Lefer. Creating evenly-spaced streamlines of arbitrary density. *Proc. Eighth Eurographics Workshop on Visualization in Scientific Computing*, pages 45–55, 1997.

[31] V. R. Kamat and S. El-Tawil. Rapid post-disaster evaluation of building damage using augmented reality. *Proceedings of the 2005 Construction Research Congress*, 2005.

[32] V. R. Kamat and J. C. Martinez. Visualizing simulated construction operations in 3D. *Journal of Computing in Civil Engineering*, 15(4):329–337, 2001.

[33] Junhwan Kim and Fabio Pellacini. Jigsaw image mosaics. In John Hughes, editor, *SIGGRAPH 2002 Conference Proceedings*, Annual Conference Series, pages 657–664. ACM Press/ACM SIGGRAPH, 2002.

[34] Adam Lake, Carl Marshall, Mark Harris, and Marc Blackstein. Stylized rendering techniques for scalable real-time 3d animation. In *NPAR '00: Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*, pages 13–20, New York, NY, USA, 2000. ACM.

[35] Justin Legakis, Julie Dorsey, and Steven J. Gortler. Feature-based cellular texturing for architectural models. In Eugene Fiume, editor, *Proceedings of ACM SIGGRAPH 2001*, pages 309–316. ACM Press, 2001.

[36] Markus Lipp, Peter Wonka, and Michael Wimmer. Interactive visual editing of grammars for procedural architecture, August 2008. Article No. 102.

[37] Fred L. Mannering, Walter P. Kilareski, and Scott S. Washburn. *Principles of Highway Engineering and Traffic Analysis*. John Wiley & Son, 2005.

[38] Martin Marinov and Leif Kobbelt. Direct anisotropic quad-dominant remeshing. In *PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*, pages 207–216, Washington, DC, USA, 2004. IEEE Computer Society.

[39] Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. Fluid control using the adjoint method. *ACM Trans. Graph.*, 23(3):449–456, 2004.

[40] Radomír Měch and Przemyslaw Prusinkiewicz. Visual models of plants interacting with their environment. In Holly Rushmeier, editor, *Proceedings of ACM SIGGRAPH 96*, pages 397–410. ACM Press, August 1996.

[41] C. Minervino, B. Sivakumar, F. Moses, D. Mertz, and W. Edberg. New AASHTO guide manual for load and resistance factor rating of highway bridges. *Journal of Bridge Engineering*, 9(1):43–54, 2004.

[42] David Mould. Image-guided fracture. In *GI '05: Proceedings of the 2005 conference on Graphics interface*, pages 219–226. Canadian Human-Computer Communications Society, 2005.

[43] Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. Procedural Modeling of Buildings. In *Proceedings of ACM SIGGRAPH 2006 / ACM Transactions on Graphics*, 2006.

[44] Xinlai Ni, Michael Garland, and John C. Hart. Fair morse functions for extracting the topological structure of a surface mesh. *ACM Transactions on Graphics (SIGGRAPH 2004)*, 23(3):613–622, August 2004.

[45] ODOT. Oregon's bridges 2002. Technical report, Oregon Department of Transportation, Salem, OR, 2002. http://www.odot.state.or.us/tsbbridgepub/PDFs/Senate_05_16_02.pdf.

[46] Yutaka Ohtake and Er Belyaev Hans-peter Seidel. Ridge-valley lines on meshes via implicit surface fitting. *ACM Trans. Graph*, 23:609–612, 2004.

[47] Jonathan Palacios and Eugene Zhang. Rotational symmetry field design on surfaces. *ACM Trans. Graph.*, 26(3):55, 2007.

[48] Yoav I. H. Parish and Pascal Müller. Procedural modeling of cities. In Eugene Fiume, editor, *Proceedings of ACM SIGGRAPH 2001*, pages 301–308, New York, NY, USA, 2001. ACM Press.

[49] Ken Perlin. An image synthesizer. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 287–296, 1985.

[50] Matt Pharr and Greg Humphreys. *Physically Based Rendering : From Theory to Implementation.* Morgan Kaufmann, 2004.

[51] Emil Praun, Adam Finkelstein, and Hugues Hoppe. Lapped textures. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 465–470, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[52] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, New York, NY, USA, 1992.

[53] PROCEDURAL. CityEngine, 2008. http://www.procedural.com.

[54] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants.* Springer Verlag, 1991.

[55] P. Prusinkiewicz, P. Mündermann, R. Karwowski, and B. Lane. The use of positional information in the modeling of plants. In Eugene Fiume, editor, *Proceedings of ACM SIGGRAPH 2001*, pages 289–300. ACM Press, 2001.

[56] Przemyslaw Prusinkiewicz, Mark James, and Radomír Měch. Synthetic topiary. In Andrew Glassner, editor, *Proceedings of ACM SIGGRAPH 94*, pages 351–358. ACM Press, July 1994.

[57] J. Punter. *Design Guidelines in American Cities.* Liverpool University Press, 1999.

[58] Adam Runions, Martin Fuhrer, Brendan Lane, Pavol Federl, Anne-Gaëlle Rolland-Lagan, and Przemyslaw Prusinkiewicz. Modeling and visualization of leaf venation patterns. *ACM Transactions on Graphics*, 24(3):702–711, 2005.

[59] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. In *Proc. Nat. Acad. Sci*, pages 1591–1595, 1996.

[60] Ken Shirriff. Generating fractals from Voronoi diagrams. *Computers and Graphics*, 17(2):165–167, 1993.

[61] Jos Stam. Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 121–128, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[62] M. G. Stewart, D. V. Rosowsky, and D. Val. Reliability-based bridge assessment using risk-ranking decision analysis. *Structural Safety*, 23:397–405, 2001.

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

[64] Jing Sun, Xiaobo Yu, George Baciu, and Mark Green. Template-based generation of road networks for virtual city modeling. In *VRST '02: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 33–40, New York, NY, USA, 2002. ACM Press.

[65] O. Tugrul Turan, Christopher Higgins, and David V. Rosowsky. Statistical modeling of coupled shear-moment resistance for rc bridge girders. *Journal of Bridge Engineering*, 13(4):351–361, 2008.

[66] G. Turk. Texture synthesis on surfaces. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH 2001)*, pages 347–354, 2001.

[67] Jarke J. van Wijk. Image based flow visualization. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 745–754, New York, NY, USA, 2002. ACM.

[68] Li-Yi Wei and Marc Levoy. Texture synthesis over arbitrary manifold surfaces. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 355–360, New York, NY, USA, 2001. ACM.

[69] Andrew Wilson and Rebecca Brannon. Exploring 2d tensor fields using stress nets. *IEEE Visualization Proceeding*, pages 11–18, 2005.

[70] T. A. Witten and L. M. Sander. Diffusion-limited aggregation, a kinetic critical phenomenon. *Phys. Rev. Lett.*, 47:1400–1403, 1981.

[71] Peter Wonka, Michael Wimmer, F. Sillion, and William Ribarsky. Instant architecture. *ACM Transactions on Graphics*, 22(3):669–677, 2003.

[72] Steven Worley. A cellular texture basis function. In *Proceedings of ACM SIGGRAPH 96*, pages 291–294, New York, NY, USA, 1996. ACM Press.

[73] Brian Wyvill, Kees van Overveld, and Sheelagh Carpendale. Creating Cracks for Batik Renderings. *NPAR 2004 Proceedings of the third international symposium on Non-photorealistic animation and rendering*, pages 61–70, 2004.

[74] Eugene Zhang, James Hays, and Greg Turk. Interactive tensor field design and visualization on surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 13(1):94–107, 2007.

[75] Eugene Zhang, Konstantin Mischaikow, and Greg Turk. Vector field design on surfaces. *ACM Transactions on Graphics*, 25(4):1294–1326, 2006.

[76] Xiaoqiang Zheng and Alex Pang. 2d asymmetric tensor analysis. In *IEEE Visualization*, pages 1–8, 2005.

APPENDICES

Appendix A – Bridge File Example

```
deck

thickness 6

dcurb 1.5

drail 3.5

spans 1

pinned 1 1

girders 2

spacing 3.5


span 1

length 30


section 1

position 15

capacity 3000 20 3 0.1 20 3 4 1 4 8


dimSection

position 15.0

dimension 20 32
```