Image-based Region Modelling Focusing on Template Generation of Road Networks

BY

Sun Jing

A Thesis Submitted to

The Department of Computing of

The Hong Kong Polytechnic University

in Partial Fulfillment

of the Requirements for

the Degree of Master of Philosophy

Hong Kong, January 2004

Certificate of Originality

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written nor material which has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

Name of Student: Sun Jing

Signature:

Abstract

Urban regions are complex multi-functional dynamic systems which play a significant role in the development of modern societies. They are mirrors of historical, cultural, economic and social aspects of human settlements. Urban synthesis, which is to collect, compile and analyze city data, construct the model and visualize the result, thus plays a very important role in the creation of the Digital Earth. Modeling and visualizing the dynamic growth of cities has become an important area of research in many fields such as urban planning, visibility analysis, environmental control, urban engineering, communication design, shopping and market facilities, cultural centers, museum display and entertainment industries.

One significant purpose of urban synthesis is to design and develop the real city of tomorrow in the optimum way and manner by using virtual cities to plan real cities. Virtual cities may be regarded as working models for the city of the future, for further development of the city and building-up volume. It may also be a totally new model for some undeveloped areas.

In this thesis, we develop an image-based 3D urban model with the main focus on transportation system generation which is a critical element in ensuring the growth and vitality within an urban community. A template-based road model is presented on analyzing several factors that will affect the transportation system. Given input maps such as land and water boundaries, population distribution and elevation, this system is able to generate road maps of various patterns. The model can also adjust itself intelligently to avoid illegal

areas, follow along the direction of elevation and connect the dead-end roads to form a valid and efficient network. In our model, it is easy to incorporate future extensions of new road patterns by adding new road templates into a template library.

The urban terrain is also constructed based on information extracted from maps. The roads and terrains are blended together to form an integrated surface model.

Publications Arising from the Thesis

[1] Jing Sun, George Baciu, Xiaobo Yu, Mark Green. Image-based Template Generation of Road Networks for Virtual Maps. Accepted by *International Journal of Image and Graphics.*

[2] Jing Sun, Xiaobo Yu, George Baciu. Inverse Medial axis Transformation in 3D Road Network Expansion. *Proceedings of 8th International Conferrence on CAD/Graphics'03.* pp 236-241, 2003.

[3] Jing Sun, George Baciu, Xiaobo Yu, Mark Green. Template-Based Generation of Road Networks for Virtual City Modeling. *Proceedings of the ACM Symposium on Virtual Reality Software and Technology.* pp. 33-44, 2002.

Acknowledgements

The MPhil. thesis and defense mark the end of two years of intensive academic learning, which was at times frustrating and yet triumphant.

There are many peoples to thank, friends and colleague alike, who made my journey all the more pleasant and memorable. I begin with the heartfelt thanks to my supervisor, Dr. George Baciu, whose guidance helped me sift out golden principles from the dross of competing ideas, and whose patience gave me room to learn. I thank also the other members of my thesis committee: Professor Stephen Chan, Professor Ze-sheng Tang and Dr. Rynson Lau.

Special thanks to all my friends in GAMA lab. Thanks to Xiaobo Yu who gave me precious support and spent time proof reading my thesis. Thanks to Wingo Sai-Keung Wong whose passion in research encouraged me a lot. Thanks to Jon Sze-Chun Lee, Cliff Kwok-Fung So, Ki-Wan Kwok, Bartholomew Ka-Chun Iu. They are all brilliant people that I am glad to work with and learn from.

Thanks to my parents for their love and support which I can always depend on.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

Over the last few decades, society has amassed an enormous amount of digital information about the Earth and its inhabitants. However, these archives pale in comparison to the flood of data which is about to engulf us. Spurred by developments in computing, communications, remote sensing, and digital media, this new wave of data threatens to overwhelm our ability to process and interpret it. Recognizing this challenge, the concept of a "Digital Earth" was introduced by Dr. Gore in an address at the California Science Center in January 1998 [19]. The Digital Earth would be a new framework for integrating a wide variety of geo-referenced data, including natural, cultural, and historical components.

Urban regions are complex multi-functional dynamic systems in the world which play a significant role in the development of modern societies. They are mirrors of historical, cultural, economic and social aspects of human settlements. Urban synthesis, which is to collect, compile and analyze city data, construct the model and visualize the result, thus is very important in the creation of the Digital Earth. Modeling and visualizing the dynamic growth of cities has become an interesting area of research in many fields such as urban planning, visibility analysis, environmental control, urban engineering, communication design, shopping and market facilities, cultural centers. However, the immediate impact and the highest demand of such generating systems can be found in digital entertainment from the design and generation of background setting in film and

advertising to 3D computer games.

1.2 Motivation

One significant purpose of urban synthesis is to design and develop the real city of tomorrow in the optimum way and manner by using virtual cities to plan real cities. Virtual cities may be regarded as working models for the city of the future, for further development of the city and building-up volume. It may also be a totally new model for some undeveloped areas.

Transportation system planning is a critical element in ensuring the growth and vitality within a community. Historically, improvements to transportation networks have exerted a large influence on urban development by alerting accessibility and development potential.

Geographic Information System has been employed into transportation modeling since early 1990s. With its help, transportation modeling of traffic impacts, calculation of vehicle emissions and consideration of wider transportation planning effects has been carried out effectively [44]. However, it will need more analysis techniques to efficiently design the road networks and make up for the insufficiency of data.

1.3 Contributions

In this thesis, we present new methods for generating an image-based 3D region model with the main focus on the template generation of road network.

1. Given input images such as land and water boundaries, population distribution and elevation, this system is able to generate road maps of various patterns which provides multiple choices for urban planning.
2. The model can also adjust itself intelligently to avoid illegal areas, follow

along the direction of elevation and connect the dead-end roads to form a valid and efficient network.

3. In our model, it is easy to incorporate future extensions of new road patterns by adding new road templates into a template library.
4. 3D terrain structures are also modeled with elevation and land/water partitioning constraints. The roads and terrains are blended perfectly to form an integrated surface model.
5. Our system is able to plan growth of a region incorporated with the current condition of development.

1.4 System Structure

Urban synthesis involves techniques of computer graphics, image processing, database technology, network technology and hardware knowledge. Modeling and visualization needs theories and approaches of computer graphics. Original photographic images need to be processed before the computer can recognize. Recognition of roads and symbol buildings, classification of vegetation and extraction of elevation information use many technologies of image processing. City data is gigantic and diversiform, an effective data engine which manages data acquisition, storage and inquiry is very important. The realization of virtual reality also needs technologies of network and hardware. In our system, these techniques are also required.

The system is composed of five main parts: Modeling Engine, Visualization Engine, Human-Computer Interaction Module (HC Module), Data Engine, and Database (as shown in Figure 1.1).

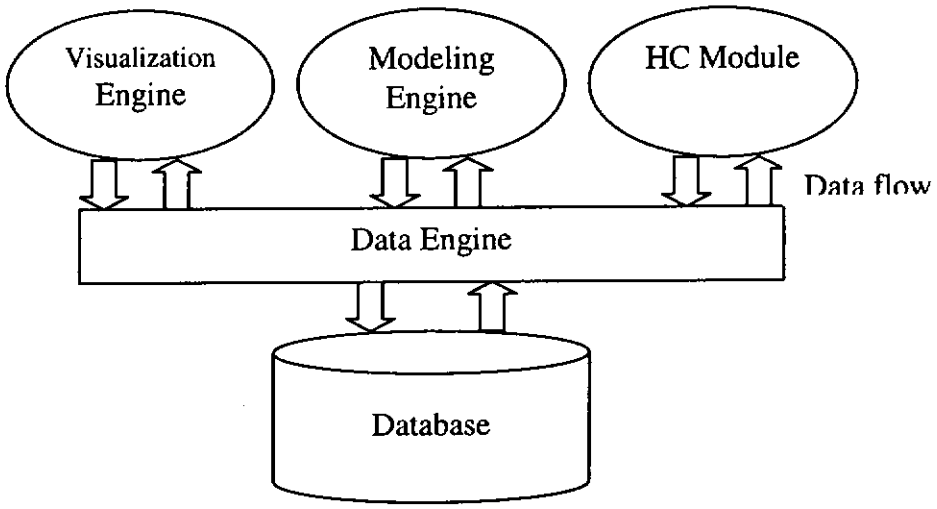


Figure 1.1: System Structure

Modeling Engine is the kernel which analyze the inputs and model them into useful information. Visualization Engine shows the model which can be directly seen by our eyes. Data Engine works as a bridge, handling all the data access, storage operation between Modeling Engine, Visualization Engine, HC Module and the Database. It also preprocesses the maps, extracting information out of the images. HC Module is introduced for user's modification, which provides more designing space for the planners.

1.5 Organization

The organization of this thesis is as follows:

Chapter 2 gives a survey of current work related to urban synthesis, and some basic concepts and techniques which we will employ in our system.

From Chapter 3 on, the flow will be directed by the brief system pipeline shown in Figure 1.2.

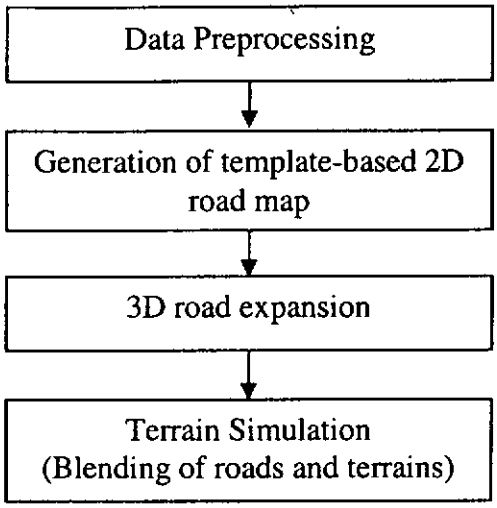


Figure 1.2: System pipeline.

Chapter 3 discusses the data classification and representation, as well as data preprocessing method for extracting information from images. As a preparation for the road generation, in chapter 4, we will introduce the concepts of road patterns and road templates. The approach to the generation of road template is also discussed. Chapter 5 shows the whole procedure of 2D road network generation. In chapter 6, our system expands 2D road segments into 3D space using an inversed medial axis approach. Chapter 7 simulates the terrain surfaces and blends the terrain and roads network together to form an integrated model.

In Chapter 8, we conclude the thesis and present several directions of future work.

2 Literature Survey

This survey includes two parts. In section 2.1, we will introduce some related work done in the urban synthesis area. In section 2.2, some basic concepts and techniques which will be employed in our system is discussed.

2.1 Related Work in Urban synthesis

Urban models are created for very different reasons and also in many different ways as well as a variety of emphases. They are used for commercial, educational, military, industrial and/or municipal purposes. The urban synthesis as a whole may have several phases: 1. Efficiently generating models of current conditions. 2. Modeling and visualizing multiple alternative plans in 3D. 3. Visualizing detailed design in broader context. We will discuss some related research in these three phases respectively in the following parts.

2.1.1 First phase – Reconstruction techniques

Many of the pressing problems often mentioned are associated with the first phase which aims at the automatic reconstruction of photorealistic 3D city models. They range from developing metadata or organizational principles for collating 3D information, (semi-) automated procedures for data acquisition, developing real-time techniques for data access using levels of details, to degree of visual fidelity and geometric accuracy, and navigation techniques. All these issues are significant from implementation and developmental perspectives and much research has been done in these areas.

As we know, city models have grown from basic cartography and traditional maps. These are merely symbolic information browsing tools and often missed important details such as elevation, land features and locality of reference. Most importantly these compilations are static with very little information about the growth pattern and the development of the urban region. With the current tools for digital modeling urban regions can be built using a multitude of sources such as aerial photogrammetry, digital pictures and video, together with symbolic maps. Researchers have developed different methods and data structures to extract and represent the information from these new resources. Wang et al. [47] developed a 3D data structure based on Formal Data Structure [36] and a system (CC-GIS), which included vector and raster as well as attribute data. Zhu [51] developed a surface triangulations method to describe the building by use of a series of surface triangulations and orthophoto used as texture of terrain and building. Li [28] designed a layer combined model for buildings which could accurately describe building structure in different layers and as a basis for future spatial analysis and manipulation.

Approaches to data analysis and automatic modeling are also highlights in the city reconstruction. Kawasaki et al. [26] proposed a new matching method between real-world video data and digital maps containing geometric data. It is an automatic organization method focusing on video objects to describe video data in an efficient way. Jokinen [24] described an approach for the registration and integration of multiple 3D data sets that could be represented as single valued parametric surfaces such as disparity maps, profile maps, or range images. In [22], Henricsson proposed two systems for automated of reconstruction of buildings from aerial images. Novel methods for feature extraction, segment

stereo matching, 2D and 3D grouping, color and object modeling and geometric reasoning were also described in this paper.

A virtual city model was described by Thomas et al. [45]. In this work, autonomous virtual actors were generated and programmed to behave like pedestrians or car drivers within a complex city environment. Cutini [10] and Bin Jiang [23] also discussed pedestrian movement in urban environments. In their work, virtual cities were populated by virtual actors providing the visitor with a real life-like atmosphere. Cremer et al. [9] provided a real-time virtual historical environment to immersive museum visitors in a 3D reconstruction of an old city. Patrons were able to access multimedia content associated with buildings and other historically significant sites. This showed a new application of city reconstruction.

From above related work, we can see that such systems just show the appearance of a real world. They are not amenable to generating new models without sufficient external data.

2.1.2 Second phase – Modelling techniques

The research done for the first phase gives us a good preparation for the planning. Regarding usefulness and implementations of digitally assisted space-related simulation environments, the following requirements are being increasingly demanded [11]:

- "Electronic Sketching": Integration of the computer at earlier stages of the special planning processes or the architectural planning processes;
- "Spatial Interaction": Increased interaction with information and planning assistance systems: e.g. by navigation in digital city models in real time; spatial interaction/ modification-possibilities of the digital model (e.g. by moving,

shifting, rotating, scaling, texturing of objects);

- pnf3VR-simulation environments: New designing of user interfaces, overcoming the "obstacle" monitor (e.g. by utilizing the VR-environment CAVE).

Research in the phase is comparatively scarce. In 1998, Kato et al. [25] proposed a new method that can model original virtual cities that have characteristics of real cities by using artificial life techniques. The method made use of L-systems to generate road networks and the genetic algorithm to generate building layouts. The road networks were composed of two types of roads – linear flow systems which were generated by using the Tree L-system and cellular networks which were generated by using the Map L-system. It might be a simple and first try in virtual city modeling.

Later in 2001, Parish et al. [40] proposed a system using a procedural approach based on L-systems to model virtual cities. For the creation of a city street map, L-systems had been extended with methods that allowed the consideration of global goals and local constraints and reduced the complexity of the production rules. An L-system that generated geometry and a texturing system based on texture elements and procedural methods composed the buildings. Their work attracted interests of many people.

However, using strictly a rule-based approach makes it difficult to create new rules into the production pool to let the road creation mechanism generate realistically looking road networks. Also, their models are built on a horizontal plane without considering the elevation. And the most important defect is that these systems have to generate city models from the beginning stage which means that it can help little to plan growth of a city incorporated with the current

condition of development.

2.1.3 Third phase – Visualization techniques

Visualization, as its name suggests, conveys information directly perceptible to human brains so that detailed spatial relationships are revealed in a way that conventional planar presentations are not capable of. The term "visualisation" has been used extensively in many fields: from scientific and engineering visualisation to entertainment industry, from physics to molecular behaviour, to medical visualisation. Now it has found its way to the fields of urban design, urban planning, and landscape architecture as well.

In the design and geography communities, visualization often refers to the static techniques, such as perspective drawings, photomontage, thematic mapping and graphics, though dynamic sense of visualization is increasingly gaining attention [42][30][31]. Chen [7] presented how CAD-based visual simulation techniques (AutoCAD and Truevision) can be applied to the domain of urban design, based on either existing or hypothesis design problems, to some extent to assist visual impact analysis. Similarly, Levy [27] used ArchiCAD to visualize a water front design alternative. The Cartogram promoted by Dorling [12] was another example of how visualization was perceived by some researchers in the urban and geography communities.

Chen [8] emphasized the dynamic sense of visualization. The focus of this paper was to introduce modern technology that would allow one to construct a data model with which the model would be rendered dynamically as one experienced different portions of the model at a reasonable frame fresh rate.

To solve the highly demanding task for the visualization of large urban complexes on the web, which is that the whole 3D model of a city is mostly too

big to be downloaded in a reasonable time and to be stored in a memory of commonly used computers, Zara et al. [50] presented a scalable approach for subdividing an urban model into smaller parts. A sequence for levels of detail specific to urban modeling was also designed.

2.2 Some Basic Concepts

In this part, let's recall some basic concepts and techniques.

2.2.1 Voronoi diagram

Voronoi diagrams are fundamental geometric structures that arise in several application contexts in disciplines such as computer science, computer aided design and manufacturing, computer graphics, robotics, finite element analysis, geological sciences, and crystallography. They may be defined for a variety of (finite) geometric sets including sets of planar or spatial points, sets of line or curve segments, boundary curves of closed planar domains, and bounding surfaces of three-dimensional volumes.

The concept of the Voronoi diagram of a planar point set was introduced by the Russian mathematician G. M. Voronoi in his treatise on the geometry of numbers. This structure may be intuitively understood as follows. Given a set of N planar points, $\{p_1,\cdots p_N\}$, their Voronoi diagram partitions the plane into N mutually disjoint regions, $VR(p_1),\cdots VR(p_N)$, such that each region, $VR(p_k)$, $1\le k\le N$, contains precisely one points, p_k, from the point set. Points in $VR(p_k)$ have the common property that they are *at least as close* to p_k as they are to *any* other points p_j, $j\neq k$, of the point set. Figure 2.1 illustrates this idea.

The region $VR(p_k)$ associated with the point p_k is called its *Voronoi region*, while the boundary segments of $VR(p_k)$ constitute the *Voronoi polygon* of p_k, denoted by $VP(p_k)$.

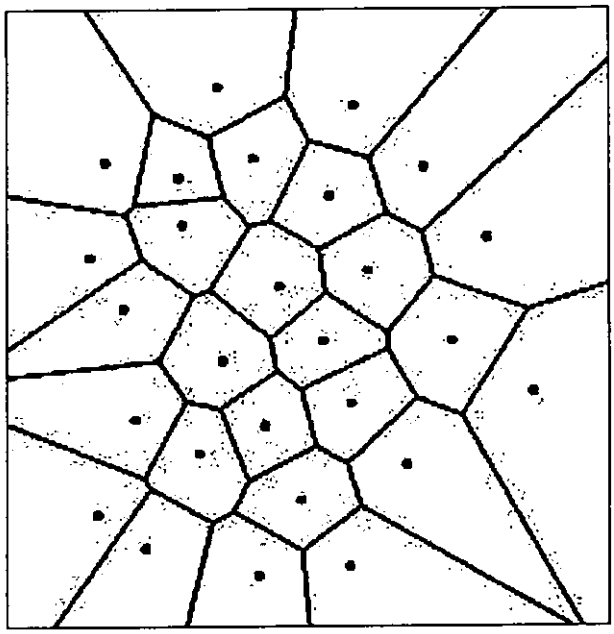


Figure2.1: The Voronoi diagram of a set of planar points.

Definitions of Voronoi Diagram [Melkemi] in a formal way:

1. A Voronoi diagram is defined as a partitioning of the Euclidean space. Let S be a set of seeds $M_1, M_2, \cdots M_n$. Let $d(M_i, M_j)$ be the Euclidean distance between M_i and M_j. The Voronoi polygon associated to M_i is defined by:

   $P(M_i) = \{M \mid d(M, M_i) < d(M, M_j)$ *for every j different from i* $\}$

   When seeds are points, we speak about ponctual Voronoi diagram.

2. The Voronoi diagram of a domain is composed of boundaries polygons *P(M)*, the boundaries are called Voronoi edges.

3. A graph environment is used to define the data structure. It associates to each polygon its seed and pointers to adjacent polygons. Each polygon is described by a list of top points and a list of sides.

Voronoi diagrams have been used in many applications such as fire observation towers, nearest neighbour clustering, facility location and path planning, etc. [39].

2.2.2 L-system

An L-system is a parallel rewriting system introduced by Lindenmayer for the purpose of describing the growth of living organisms [32].

Simulation begins with an initial string called the axiom, and proceeds in a sequence of discrete derivation steps. In each step, rewriting rules or productions replace all modules in the predecessor string by successor modules. The applicability of a production depends on a predecessor's context, values of parameter, and on random factors [41].

In the most extensive case, a production has the format:

id : lc < pred > rc : cond -> succ : prob

where *id* is the production identifier (label), *lc*, *pred*, and *rc* are the left context, the strict predecessor, and the right context, *cond* is the condition, *succ* is the successor, and *prob* is the probability of production application. The strict predecessor and the successor are the only mandatory fields.

Table 2.1 shows most of the characters used in L-system and their interpretations.

Character	Meaning
F	Move forward by line length drawing a line
f	Move forward by line length without drawing a line
+	Turn left by turning angle
-	Turn right by turning angle
|	Reverse direction (ie: turn by 180 degrees)

[	Push current drawing state onto stack
]	Pop current drawing state from the stack
#	Increment the line width by line width increment
!	Decrement the line width by line width increment
@	Draw a dot with line width radius
{	Open a polygon
}	Close a polygon and fill it with fill colour
>	Multiply the line length by the line length scale factor
<	Divide the line length by the line length scale factor
&	Swap the meaning of + and -
(	Decrement turning angle by turning angle increment
)	Increment turning angle by turning angle increment

Table 2.1: Characters used in L-system and interpretations.

Figure 2.2 shows the result of the following example:

axiom: X

p1: F --> FF

p2: X --> F-[[X]+X]+F[+FX]-X

ø = 22.5

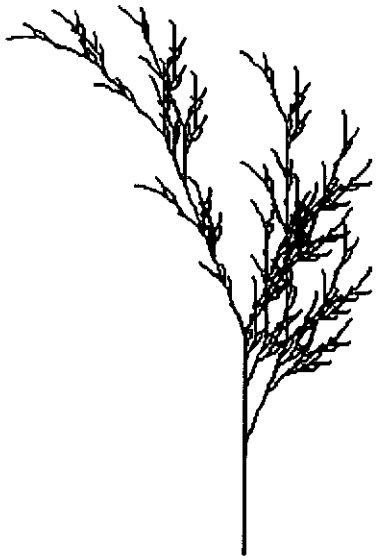


Figure2.2: Example of L-system.

2.2.3 Medial axis

Shape representation is considered a difficult and challenging problem in computer vision and computer graphics. The *Medial Axis Transformation (MAT)* was first introduced by Blum [6] as a shape descriptor used in thinning a shape to find its skeleton consisting of a points that are equidistant from its envelope [5][39]. Since its introduction, the MAT has become a powerful tool for path planning, compact shape description and other applications [18][38][49].

The medial axis has several properties which neither Boundary Representation nor Constructive Solid Geometry directly provide. First, because it elicits important *symmetries* of an object, it facilitates interrogation of symmetrical objects. Second, the MAT exhibits *dimensional reduction*; for example, it transforms a 3D solid region into a connected set of points, curves, and surfaces, along with an associated radius function. Third, once a region is expressed with the MAT, the skeleton and radius function themselves may be manipulated, and the boundary will deform in a natural way, suggesting

applications in computer animation. Fourth, the skeleton may be used to facilitate the creation of coarse and fine finite element meshes of the region. Fifth, the MAT determines constrictions and other global shape characteristics that are important in mesh generation, performance analysis, manufacturing simulation and path planning. Finally, the MAT can be used in document encoding and other image processing applications.

Figure 2.3 shows an image and its medial axis in 2D plane.

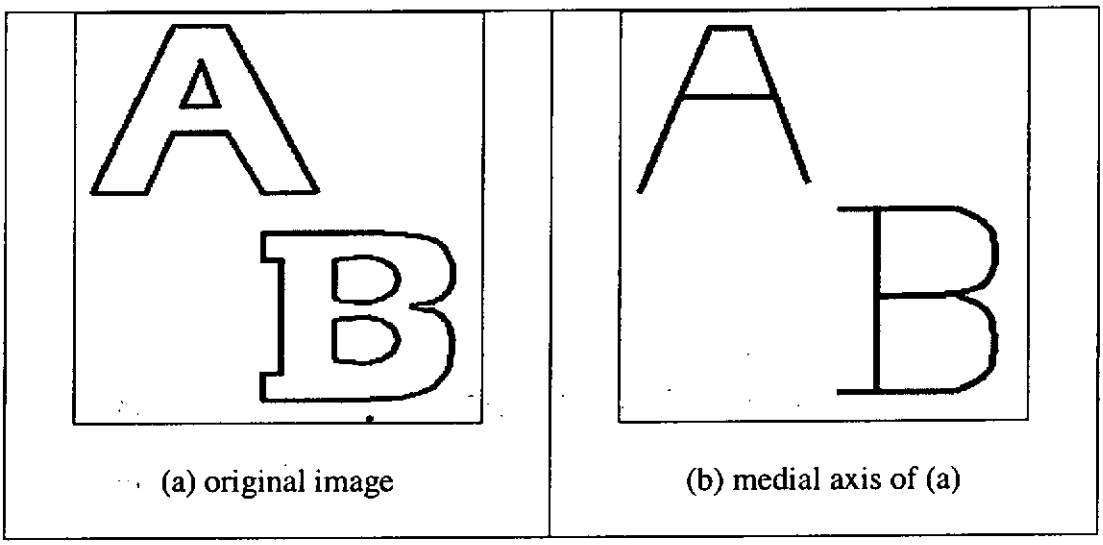


Figure 2.3: An example image and its medial axis.

2.2.4 Delaunay Triangulation

Of the many methods to triangulate arbitrary planar domains, the Delaunay triangulation is perhaps the most widely known. It can be defined in this way:

- A triangulation is a subdivision of an area (volume) into triangles (tetrahedrons).
- The Delaunay triangulation has the property that the circumcircle (circumsphere) of every triangle (tetrahedron) does not contain any points of the triangulation.
- The Delaunay triangulation is the dual structure of the Voronoi diagram.

One of the most important advantage of Delaunay triangulation is that it avoids the creation of long thin triangles automatically. Lawson has proved that the Delaunay triangulation is *local equiangular*. This means that for every convex quadrilateral formed by two adjacent triangles, the minimum of the six angles in the two triangles is greater than it would have been if the alternative diagonal had been drawn and the other pair of triangles chosen. Because of this property, Delaunay triangulation is a natural choice for mesh generation in finite element analysis and contour plotting applications.

Figure 2.4 shows an example to show the relationship between the Delaunay triangulation and Voronoi diagram.

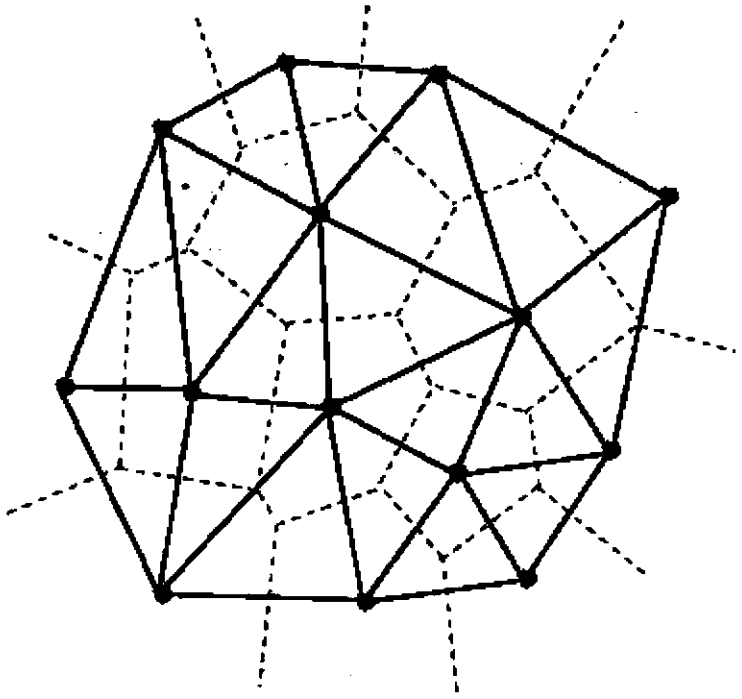


Figure 2.4: Example of Delaunay triangulation.

2.2.5 Polygon Clipping

Clipping 2D-polygons is a fundamental operation in image synthesis. For example, it can be used to render 3D images through hidden surface removal, or

to distribute the objects of a scene to appropriate processors in a multiprocessor ray tracing system.

An algorithm that clips a polygon is rather complex. Each edge of the polygon must be tested against each edge of the clipping window, usually a rectangle. As a result, new edges may be added, and existing edges may be discarded, retained, or divided. Multiple polygons may result from clipping a single polygon.

Figure 2.5 illustrates a simple case of the results of clipping a polygon to a clipping window.

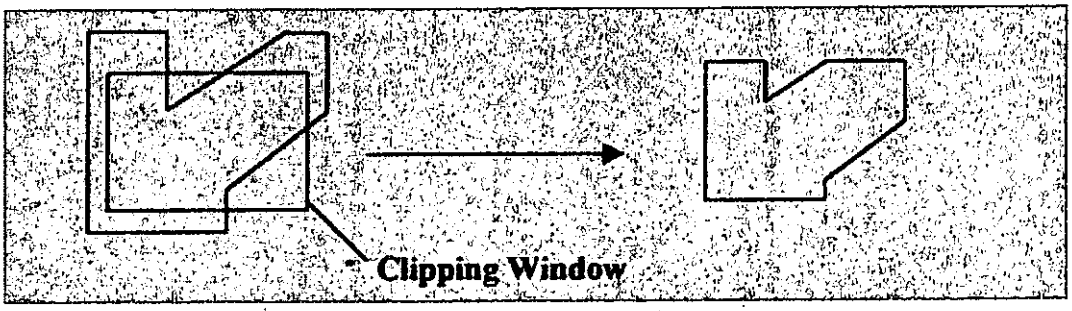


Figure 2.5: A case of polygon clipping.

Several very efficient algorithms are available for special cases:

1. Sutherland and Hodgeman's algorithm [43] is based on a divide-and-conquer strategy that solves a series of simple and identical problems that, when combined, solve the overall problem. The simple problem is to clip a polygon against a single infinite clipping edge. This process outputs the series of vertices that define the clipped polygon. Four clipping edges, each defining one boundary of the clipping window, are used to successively to fully clip the polygon. This algorithm is limited to convex clip polygons.
2. Weiler and Atherton algorithm [48] is a more general method which can be used to compute the intersections between two polygons. Although the

algorithm is powerful, it is not well adapted to the 2D case, with a rectangular clipping region, because of its complexity.

3. The Liang and Barsky algorithm [29] provides a polygon clipping method based on the parametric equation of the polygon edges. One edge can give four scalar parameters. By establishing a classification based on the contribution of each parameter, this algorithm produces the needed turning points. This algorithm requires that the clipping polygon be rectangular.

More general algorithms are presented by Foley [15], Maillot [33] Andereev [2], Montani and Re [37], Vatti [46] and Greiner-Hormann [20]. All these algorithms are quite complicated.

3 Data Manipulation

Data are the objects that modeling engine, visualization engine and HC module are operating on. City modelling involves large amount of and various kinds of data. To effectively organize and manage these data, logical data structures and efficient data processing methods are required.

In our system, data include *model data* and *control data*.

3.1 Model data

Model data, which are often stored as geometric primitives, save the whole urban model. At the beginning, they are users' primary design stage of a city like symbolic roads and buildings, or current condition of a city extracted from maps or telemetric sources, which are characteristics of a certain city comparing to a general pattern and blend into the whole structure of the road network system. The model data are updated in the modelling procedure until the whole urban structure is generated.

We design a hierarchical storage structure for model data. Vertices are primary elements (the first level). Other complicated data structures (the second level) such as lines, polygons can be represented by the correlations of vertices. Also, there are other more advanced data structures based on the second level. A higher-level data structure may be represented by several low-level structures while a lower-level data structure may have relationship with many high-level structures. If this many-to-many relationship is not taken into consideration, there will be waste of storage space and chaotic in logic. Considering this, we design a

hierarchical model which place different level of data structures into different files where there are links in between. The model has three levels (files). The first file contains the coordinates of all the vertices. The second contains basic geometric primitives like isolated points, lines and polygons. The third contains more complicated structures described as bounding box. The specification of the three files will be discussed in detail below.

This hierarchical model can represent data more clearly and accelerate the access speed. Give a simple example to show this advantage. Let there be m geometric primitives and totally n points. If we use a single file to save all the data, the time needed to find a particular primitive and its vertices may be $O((m+n)*(m+n)^x)$ while it is only $O(m*n^x)$ if we use the 3-level model (x means this primitive has x vertices).

File Specification:

(i) Vertices:

VID	*X*	*Y*	*Z*

VID of every vertex should be unique and it is the key word for search. This file saves the coordinates of all the vertices.

(ii) Geometric primitives:

GID	*Type*	*VID1*	*VID2*	...	*VIDn*	*Attr1*	*Attr2*	...	*Attrm*

GID: ID of each geometric primitive.

Type: Type of geometric primitive: point, line, polygon, etc.

VIDi: Pointer of corresponding vertices.

Attri: Attributes of the geometric primitive, e.g. line width.

(iii) Bounding box:

BID	*MinX*	*MaxX*	*MinY*	*MaxY*	*MinZ*	*MaxZ*

We use bounding box to describe a more complicated structure like a block in a city. The bounding box is represented by its boundaries in three dimensions.

3.2 Control data

Control data are often represented as 2D color images which can be easily generated either by drawing or by scanning from statistical and geographical maps [14]. They work as parameters and control the action of modelling engine.

The maps usually involved are location maps which provides city boundaries as well as the distribution of land, water and vegetation; population density, rainfall, etc. These different maps decide the general distribution of people; elevation map which modifies the extension of roads and district map which defines different city patterns.

Most of the image data should be preprocessed in order to get rid of noise, correct color bias or enhance important features. Gaussian noise, and salt and pepper noise in elevation image or population density image are the frequent sources that will reduce the image quality and may cause error when used as control data. *Medium filter* or *low pass filter* is efficient in dealing with these noises. The transfer function of an ideal 2D *low pass filter* is

$$H_{(u,v)} = \begin{cases} 1 & D_{(u,v)} \le D_0 \\ 0 & D_{(u,v)} > D_0 \end{cases} \qquad (3\text{-}1),$$

where D_0 is the cut-off frequency of an ideal *low pass filter*.

Boundary is another kind of important control information which is used in validity control and breakpoint classification. Boundary partition the image into different parts which have different attribute and are presented by different colors

or gray values. We use *Isotropic Sobel filter* to extract the boundary. The *Isotropic Sobel filter* consists of two kernels: horizontal operator $\begin{bmatrix} -1 & -\sqrt{2} & 1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix}$ and verticle operator $\begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix}$, which detect horizontal and vertical changes respectively in an image. If both are applied to an image, the results can by used for boundary extraction.

In order to acquire the information of illegal areas for road construction such as outside the city boundary or a lake and inside parks, we need to define a standard to represent them and do some modifications on the map. For example, in a location map, we use a correspond value of 0 for the pixel intensity (RGB(0,0,0)) as the land boundary and blue (RGB(0,0,255)) as the water boundaries, ocean and lakes. When the city engine reads the pixel information, it can compute the land feature they imply.

For maps, we often use colors to distinguish areas with different properties (Figure 3.1 shows a population density map).

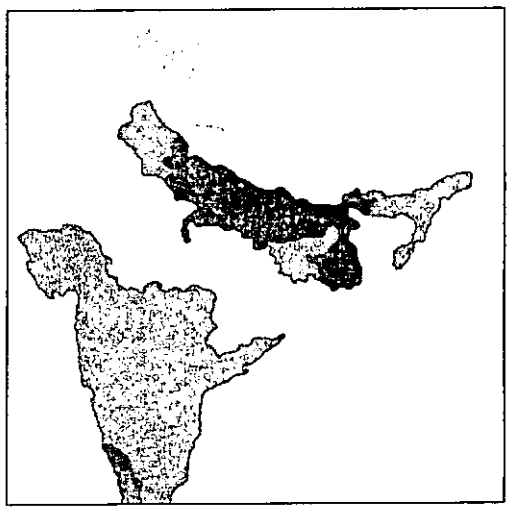


Figure 3.1: A map.

In cartography, a specific data set is associated with a specific intensity value of a color. Often, gray value of a certain color is referred when drawing a map. That is to say, we often use a darker color to indicate an area with a denser population or a higher altitude. The color values, which are represented by RGB (*0-255, 0-255, 0-255*) values, cannot be sorted directly. They should be transformed to gray values first. The transform function is

$$G_{(x,y)} = 0.3 \times R_{(x,y)} + 0.59 \times G_{(x,y)} + 0.11 \times B_{(x,y)} \qquad (3\text{-}2)\,.$$

Two similar colors have close gray values after transform, which means they will lie in same gray band. Slight color bias and blur could be corrected by gray transform.

The Figure 3.2 illustrates the relationship between the color extracted from Figure 3.1 and its gray value.

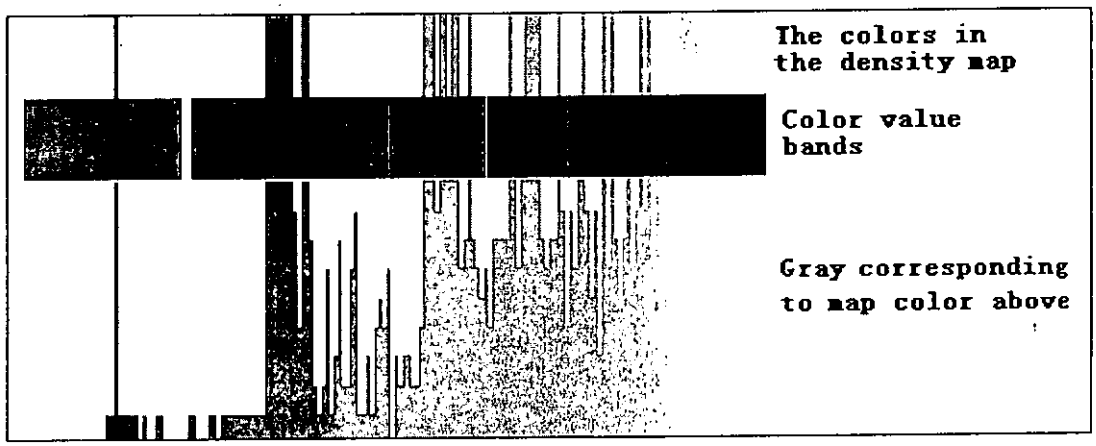


Figure 3.2: The color and gray value.

In some cases, the assumption that a darker color indicates an area with a denser population or a higher altitude is not accurate. We need to be able to find out what information that colors denote and assign property value to a certain color. Figure 3.3 shows an interface for user to assign values to the colors and

eliminate the disturbance of noise by changing the threshold on the left slide through calculating the actual area size of the color occupied.

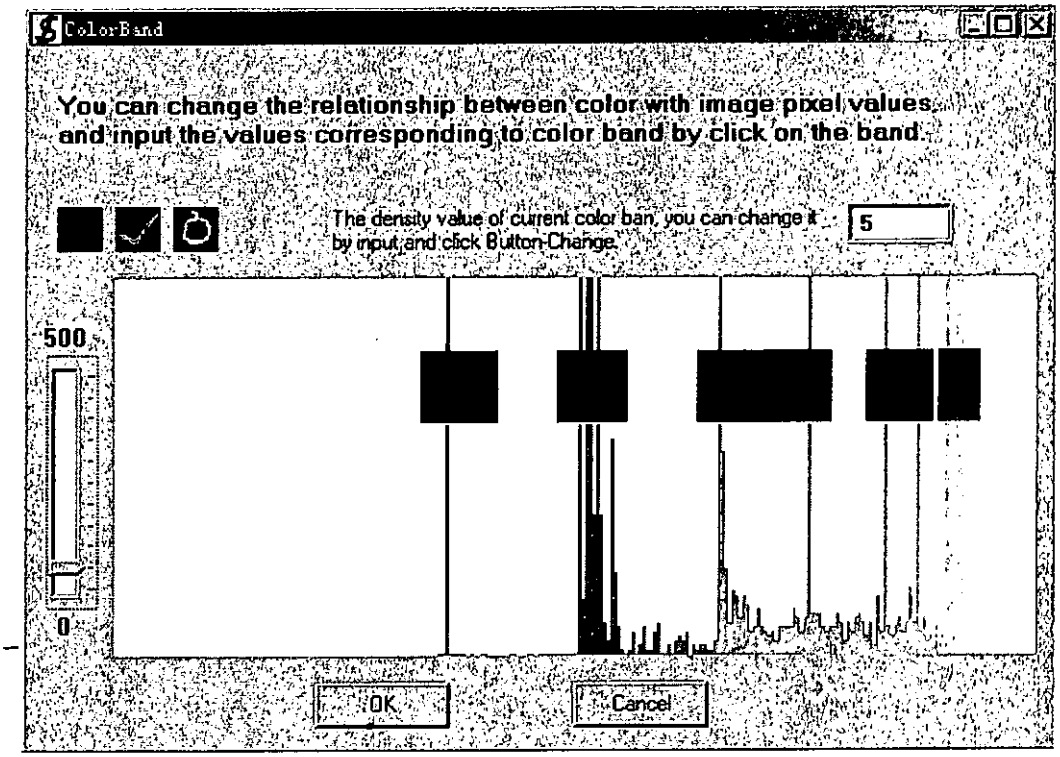


Figure 3.3: An interface for user to assign values.

According to the characteristics of image, the map will provide data value in regular grid format (raster format). The density of the grid is associated with the resolution of the image. The value associated with certain color which we can define by Figure 3.3 is for each vertex of the grid (pixel). If there is a point lies inside a grid (shown in Figure 3.4), the value of that point should be computed through interpolation.

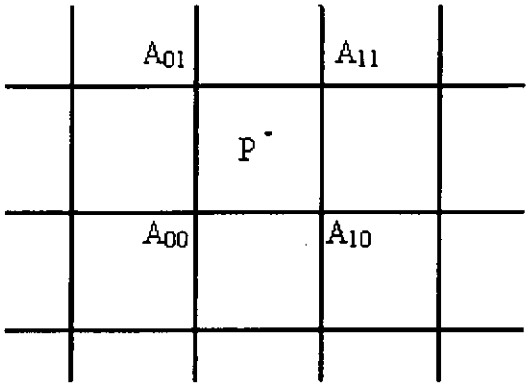


Figure 3.4: Raster format.

Point P lies in grid $A_{00}\,A_{10}\,A_{11}\,A_{01}$, then

$$P = (1-X_0)\times(1-Y_0)\times A_{00} + X_0\times(1-Y_0)\times A_{10} + (1-X_0)\times Y_0\times A_{01} + X_0\times Y_0\times A_{11} \quad (3\text{-}3)$$

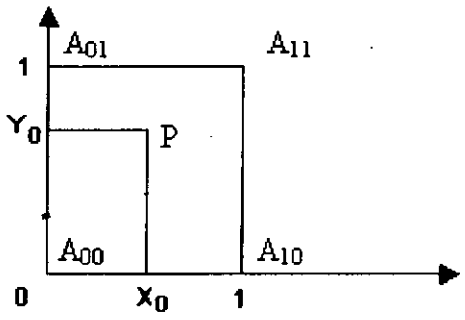


Figure 3.5: Four-point interpolation.

4 Road Patterns and Road Templates

As preparation for road network generation, we will talk about pattern and template in this chapter.

The traffic network of a city is very complicated and seems difficult to be described at the first glance. There are many factors that will affect the traffic network, e.g., geographical information, sociostatistical information, government policies, historical influences, etc., which endow the network its own characteristics. However, disciplines can be summarized after analyzing a large number of the transportation networks, which are called road patterns in this thesis.

4.1 Road Patterns

Urban roads are "artificial products" often generated by the need to connect resource centers. A pattern is just the internal regularity underlying the diversity of appearances. A pattern is conceptual and global, not considering details and exceptions. In [1], Alexander et al. describe a pattern language, which consists of over 250 relevant patterns for the successful construction of cities, roads and buildings. In Figure 4.1, several frequent road patterns are listed:

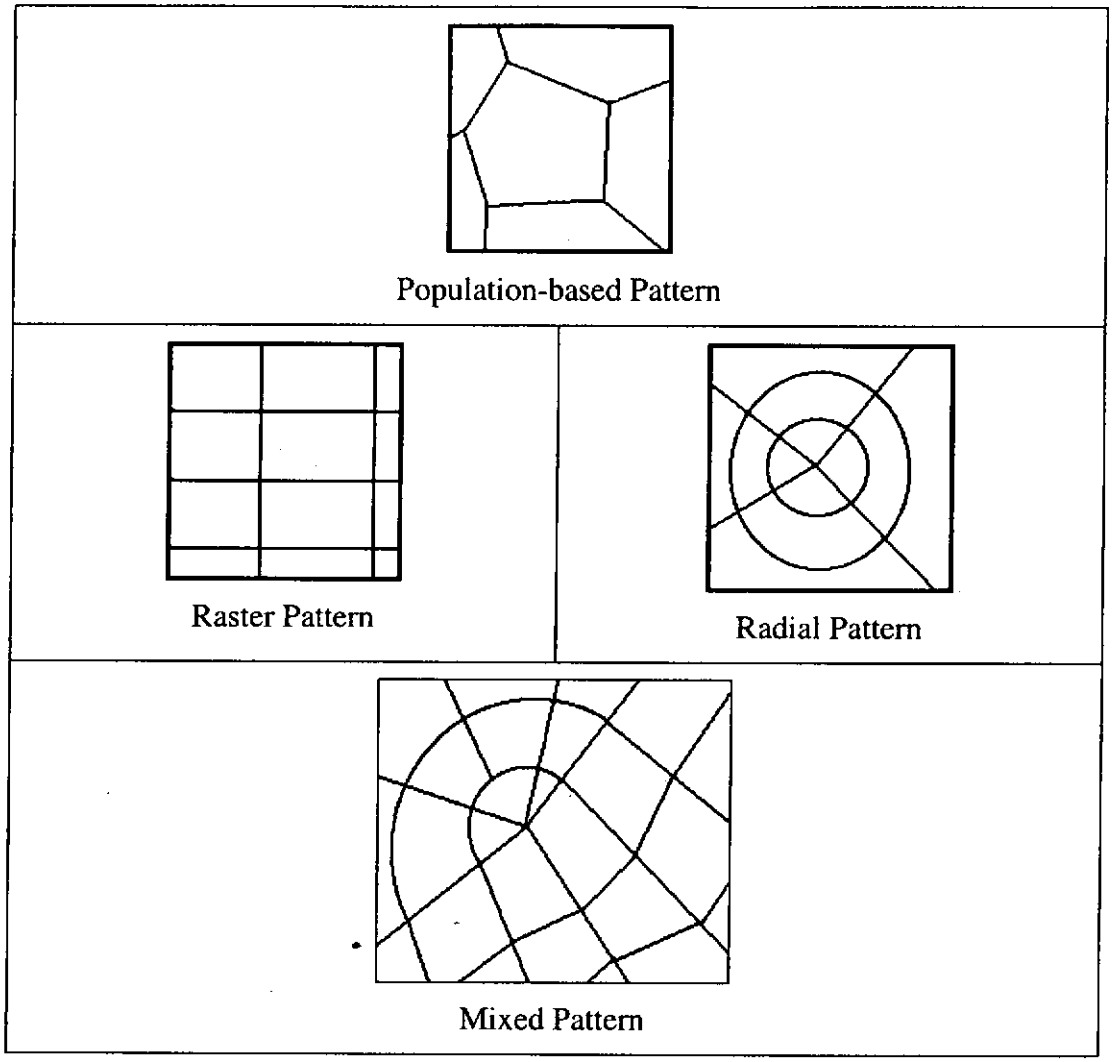


Figure 4.1: Several frequent road patterns.

Often, older cities show a population-based pattern because it is a natural way of transportation network growth. In polycentric cities, raster patterns prevail, whereas in monocentric cities, radial patterns dominate [16]. Many real cities display more than one single pattern due to historical growth and different phases of growth, so mixed patterns are introduced. These mixed patterns are actually a combination of simpler patterns, as illustrated in Figure 4.1. In the next section, we describe how templates can be constructed and used for complex traffic network pattern generation.

4.2 Road Templates

For each pattern, a corresponding template is designed. Template is defined by *rules* and *parameters*. Rules define the overall road structure while parameters provide different variation in the pattern specification.

4.2.1 Population-based Template

A transportation system serves many functions, but among the most important are the mobility of people and goods and the access to the main facilities. It is often found that roads in higher density regions are also denser in order to alleviate the traffic flow burden and people everywhere should have easy access to the road. Taking these two aspects into consideration, we use the Voronoi diagram as the generation method for population-based template.

Imagining that if we concentrate the population of an area into one point (the density point as it is called here), the polygon enclosing the density point can be seemed as a city block deriving from the population density. The boundaries of polygon are thus the roads which carry people all around. The approach to the construction of population-based template comes out: Using the density points extracted from maps as seeds for generating a Voronoi diagram, the Voronoi edges we get are the result roads.

A dichotomy method with threshold T is proposed for the area subdivision. Besides the main parameter: population density, the area size should also be considered because a large area with less or no people inhabited should also have some roads in case people will pass by. Therefore, two parameters are considered in this subdivision: the population density and the area size. They work together

and interactionally. Their influence to the division are reflected by their weight w_p, w_s and $w_p + w_s = 1$. The whole city is divided iteratively into small areas until for each area i, $T_i \leq T$. The T value for area i T_i is given by:

$$T_i = w_p * P_i / P + w_s * S_i / S ; \qquad (4\text{-}1)$$

where P: total population; P_i: population of area i; S: total area size; S_i: size of area i; w_p, w_s: weight of population and size. The computation of S_i and P_i depends on the information from boundary map and population density map which we will discuss later. The density of roads can be controlled by the threshold T while T is predefined by the city planners. A smaller T will leads to denser roads.

The dichotomy method is given as follows in Table 4.1:

Step0: $i = 1$;

Step1: Calculate T_i using equation (4-1).

if do not exist i that $T_i > T$

then Goto Step3;

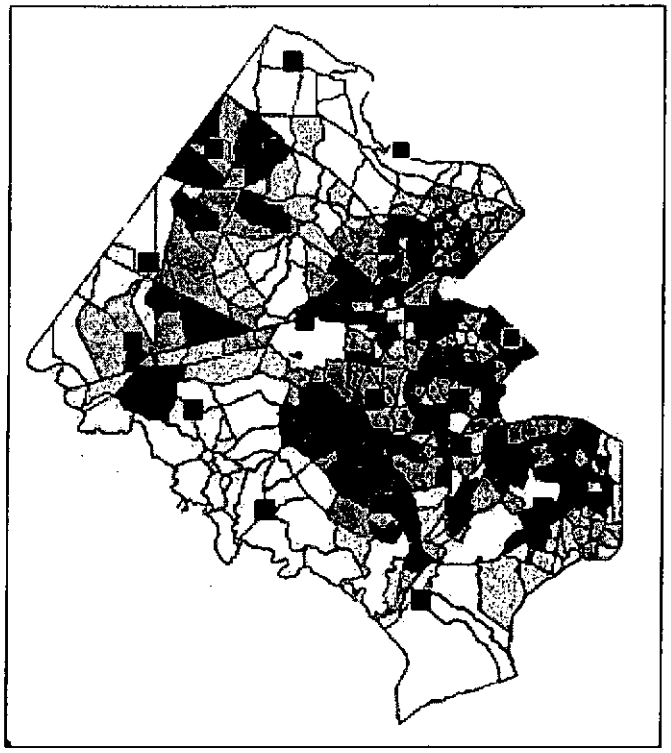
Step2: For all the subareas i that $T_i > T$, divide the area into half on its longer edge.

Goto Step1;

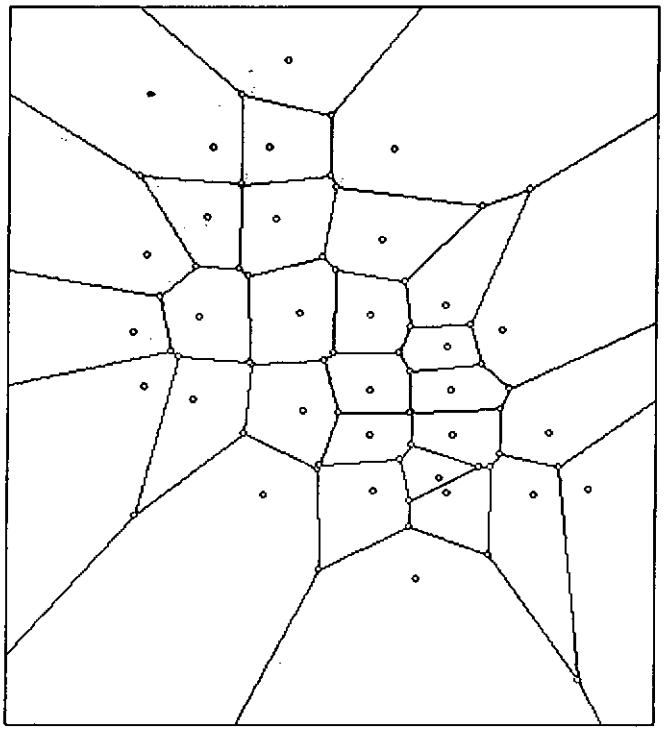
Step3: Density points are computed from the center of gravity of each subarea.

Table 4.1: Dichotomy method for density point extraction

Figure 4.2(a) shows the extraction result of density points and Figure 4.2(b) shows the corresponding Voronoi diagram.



(a): Extraction result of density points.



(b): Corresponding Voronoi Diagram.

Figure 4.2: Example of population-based template.

4.2.2 Raster and Radial Template

This template serves as a mechanism for growing patterns. It is similar in format to an L-system but simpler in the generative rules.

This template starts with a point and generates vertices iteratively until it reaches the bounding box of the map. Pattern decides the way of expansion and parameters decide the density, slope, etc. After generating all the vertices, we link them in pairs according to their location, and a connected road network is synthesized.

Template = {Rules, Parameters}

Rules = {

Generation Rule: Starting Point $\xrightarrow{pattern,density,slope,step}$ {Branch Points} + {Edges}

Branch Point $\xrightarrow{pattern,density,slope,step}$ {Branch Points}+{Edges}

Stop Rule: Branch Point $\xrightarrow{BoundingBox}$ {Leaf Point}

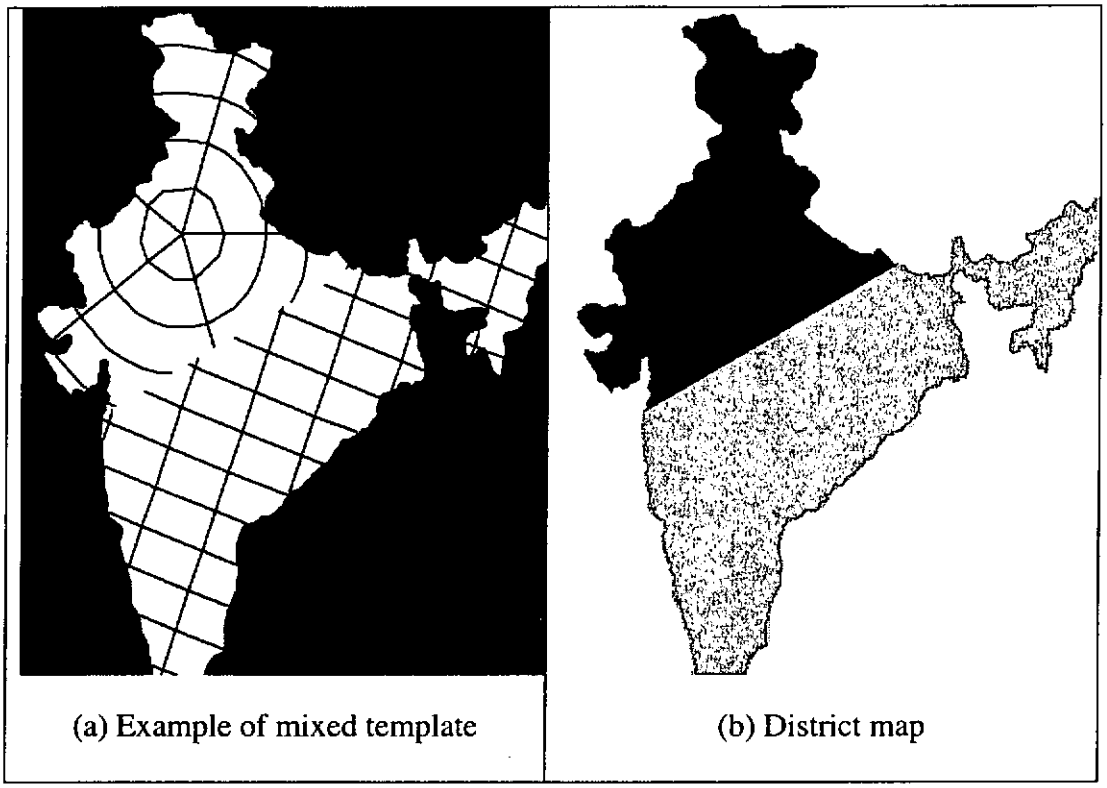
Connection Rule: {Branch Points}+{Leaf Points} $\xrightarrow{pattern}$ {Edges}

}

Parameters = {density, slope, step, bounding box}

4.2.3 Mixed Template

As the template of mixed pattern, the mixed template is also a combination of templates of simple patterns. We apply the corresponding templates above into different regions to produce the whole road structure as shown in Figure 4.3. The region extraction has been discussed in Chapter 3 and the connection of the division of different patterns will be discussed in the next Chapter.



(a) Example of mixed template

(b) District map

Figure 4.3: Example of Mixed Template.

5 Generation of 2D Road Map

In [1], Alexander et al. emphasize that cities should be subdivided into local transport areas by highways and these transport areas themselves should be further subdivided into communities and neighbourhoods by streets. In other words, highways are like main arteries in a body while streets are like capillary vessels. The difference between highway and street is not only in the different width or length, in fact, it most lies in the function in the city transportation and representation of the city's features, such as road pattern. The generation of a 2D road map takes two main steps in which the highway map and the street map are generated respectively. Figure 5.1 shows the pipeline of the generation of 2D road map.

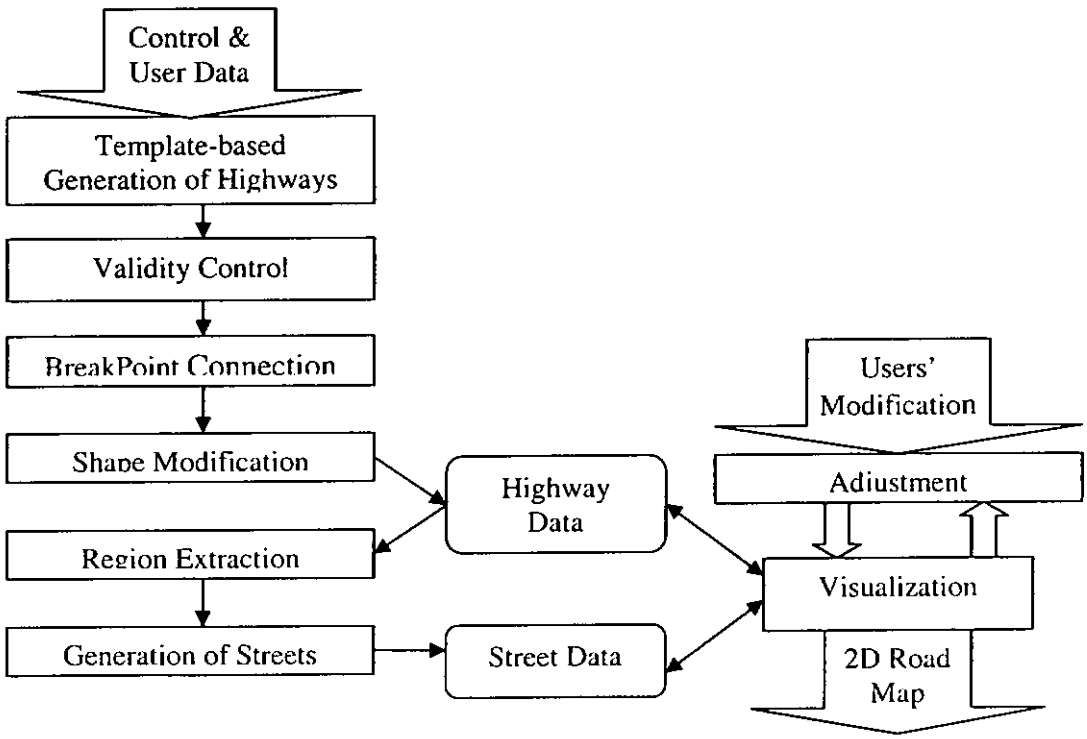


Figure 5.1: Pipeline of 2D road generation.

5.1 Generation of Highways

For the framework of traffic networks, highways cover the city, bear the main transportation flow and exhibit certain patterns. After applying the templates described in Chapter 4, we can obtain a rough highway model. The parameters, which are used in the template, are obtained from the control data such as population density map. However, in the initial phase of template application, boundary information is not yet represented and illegal areas inside the city are not yet taken into consideration. In order to get approximately practical highways, three steps should be implemented on the original result: 1. validity control; 2. breakpoint connection; 3. shape modification. These steps should be done in order.

5.1.1 Validity control

Highways may cross areas such as rivers, lakes, symbolic buildings, etc. which may not allow the construction of smaller ground roads and is so-called as *illegal area* here. In such cases, the system needs to do automatic adjustments so that the local environment constraints are satisfied. The road synthesis algorithm attempts to find all the illegal roads and then handles them according to their specific circumstances. In the following paragraphs we show how these constraints are handled based on three general cases.

Let d_i be the *direct distance* of road i between the illegal areas, let b_i be the *bypass distance* which means the shortest distance along the boundary between legal and illegal area. Figure 5.2 illustrates the definition of direct distance and bypass distance. a and b are two points in the legal area (gray area shown in

Figure 5.2). Dashed line shows the direct distance and the solid lines show the bypass distance.

Figure 5.2: Illustration of direct distance and bypass distance.

There are three solutions to maintain validity of roads due to different circumstances (shown in Figure 5.3):

1. cross directly: if $d_i \leq D_{\max}$;
2. discard: if $(d_i > D_{\max})$ and $(b_i / d_i > R_{\max})$;
3. bypass: if $(d_i > D_{\max})$ and $(b_i / d_i \leq R_{\max})$

$D_{\max}, R_{\max}$ are predefined parameters which indicate the specified maximal directed distance and ratio of bypass distance to direct distance. For example, the ratio of cost of building a bridge and that of building a road will affect $R_{\max}$. They will be decided by the funding and the government policy eventually.

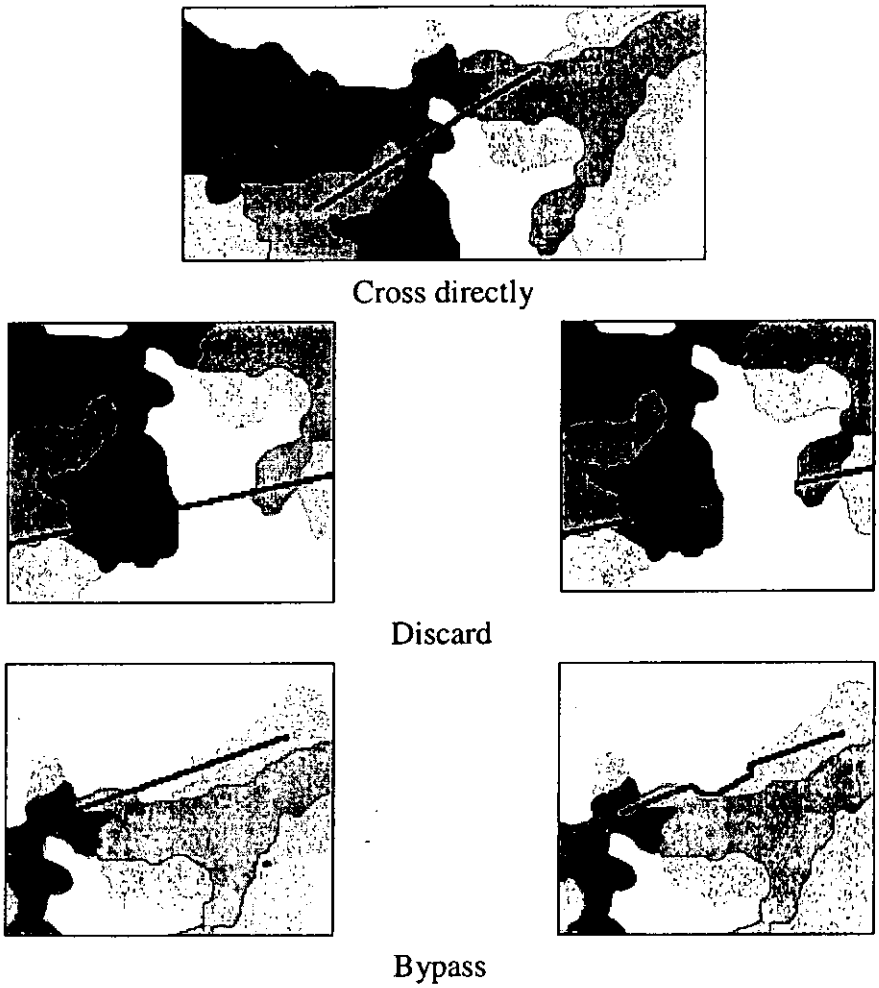


Figure 5.3: Three circumstances of illegal control.

5.1.2 Breakpoints Connection

To maintain the connectivity of a traffic network, each node in this graph should connect at least two different nodes and dead ends should be avoided. There are two kinds of breakpoints: interior and boundary. Interior breakpoints can be classified as *implicit interior breakpoints* which are produced by the block of illegal areas inside a city, described in section 5.1.1 as the "discard" case and *explicit interior breakpoints* which are due to the user's disconnected input or exist between the divisions of mixed patterns. Boundary breakpoints can be further divided as *water boundary breakpoints* and *land boundary breakpoints*.

Water boundary breakpoints are those ending at the city boundary where there is illegal area like an ocean on the other side. Land boundary breakpoints are those that end at the land boundary. Figure 5.4 shows the different cases of breakpoints.

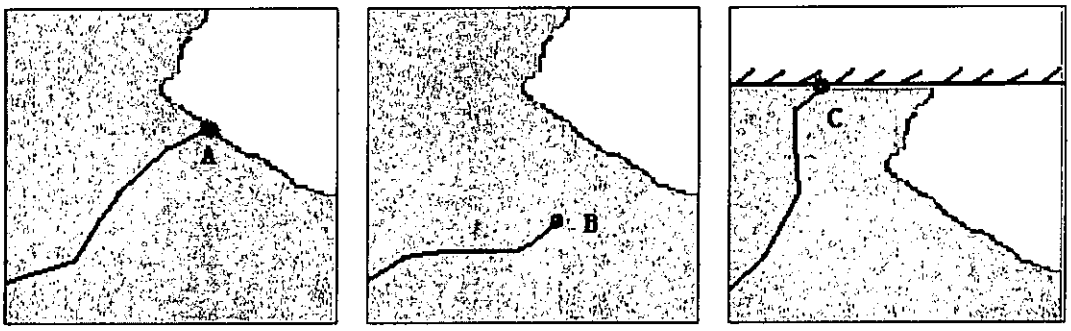
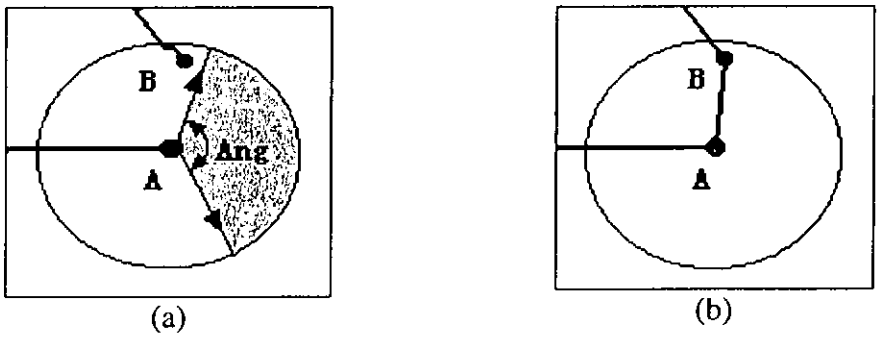


Figure 5.4: Example of breakpoints.

(A: water boundary breakpoint; B: interior breakpoint; C: land boundary breakpoint)

For the implicit interior breakpoints, the road should go along the area of the illegal boundary in order to meet the nearest valid road. For the explicit breakpoints, we search by gradually increasing the angle expanding from the original road direction and radius of circle centered by the breakpoint (as shown in Figure 5.5(a)) in order to find the appropriate connection (shown in Figure 5.5 (b)).



(a) (b)

Figure 5.5: Connection of explicit interior breakpoints.

The land boundary breakpoints are left unmodified. However, the water boundary one, because it is not reasonable to build many harbors along the

coastline, we let the breakpoint go along the boundary until it meets another one. Figure 5.6 (a) shows the original water boundary breakpoint and (b) indicates how it grows.

(a) (b)

Figure 5.6: Connection of water boundary breakpoints.

The algorithm to connect a water boundary point can be described in four steps (shown in Table 5.1):

STEP 0: Set $step\_length = A$, and $cv = a$.

STEP 1: With *cv* as center and *step_length* as radius, find if there is a breakpoint inside the circle.

If there is one *bv*, then connect *cv* and *bv* and go to **STEP 3**.

STEP 2: Connect *cv* with another water boundary point *cv'*. Length of *cvcv'* should less than *step_length*, and *cvcv'* does not cross illegal area. Set $cv = cv'$ and go to **STEP 1**.

STEP 3: Return the path.

Table 5.1: Algorithm of water boundary connection.

Figure 5.7 shows the result path (using algorithm in Table 5.1) from a water boundary breakpoint *a* to another breakpoint b, which could be any kind of breakpoints.

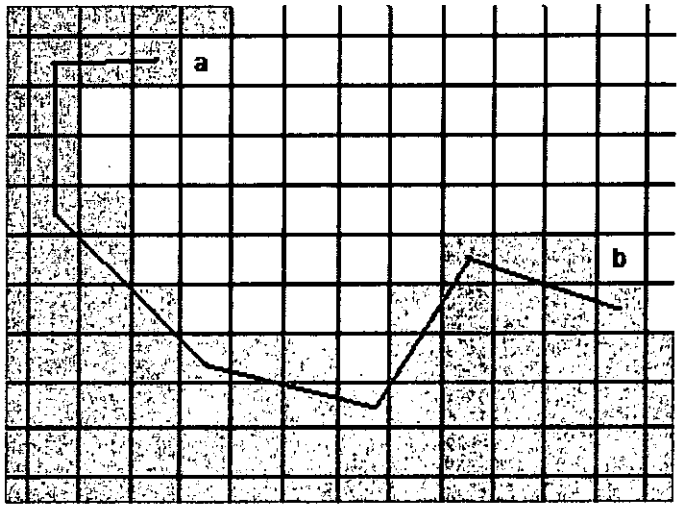


Figure 5.7: The way to connect two water boundary breakpoints: a and b.

5.1.3 Shape Modification

Roads are not always straight. They are often curved around large gradients or obstructions due to different altitudes. At the same time, the destination points act like strong attractors that dictate the growing road pattern with the objective to minimize deviation. This kind of attraction factor decides the freedom of the growth of roads. Therefore, the shapes of roads are decided by two aspects: *elevation* and *freedom*. These two factors work together while restricts each other in the mean time. For example, a bigger *freedom* makes the growth of roads rely more on the variation of *elevation*. We need to find a balance between them. There is no general criterion to choose a balance point of these two things. It can be defined by users or decided by the features, history and custom of the city.

This module tries to find a more realistic road by segmenting the original road and moving every time towards a reasonable direction from the current starting point at a specified step.

Let *OD* be the original road direction, *DD* be the direction from current starting point to the destination. To define the best elevation direction *ED*, the system emits many fixed-length radials centered by the current point, gets the elevation variation along each radial and adopts the direction of the smallest variation as the best elevation direction (shown in Figure 5.8). If there are several radials which have the same variation, the system chooses the one whose angle with *OD* is the smallest.

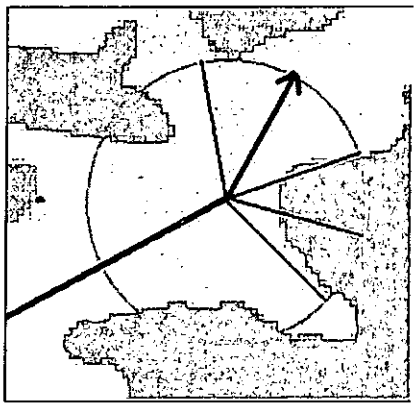


Figure 5.8: Search for the best elevation direction ED.

The actual direction of the road AD is then:

$$AD = ED * \alpha + DD * (1 - \alpha); \qquad (5\text{-}1)$$

where $\alpha = (1 - Ang/90) * F$ (*Ang* is the acute angle between OD and DD, *F* is the freedom parameter between 0 and 1).

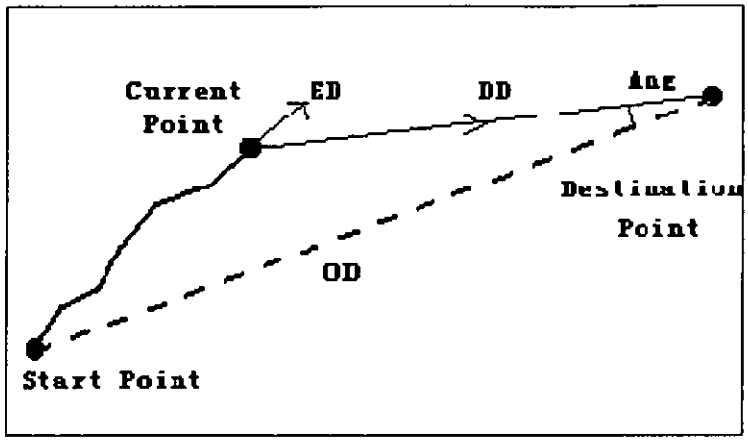


Figure 5.9: Computation of AD.

5.2 Generation of Streets

Streets cover the local transport areas enclosed by highways and boundaries. As the preparation for streets generation, we need to extract all the local transport areas.

5.2.1 Region Extraction

In Figure 5.10, D1 is a blank image of the same size as a city map. Mapping the location map onto D1, we get D2 which retains the boundary and legal area information. D3 comes from mapping all highways onto D2. After we connect all the illegal breakpoints, D3 becomes a connected graph. By extracting the regions separated by highways and marking them respectively, the local transport areas appear.

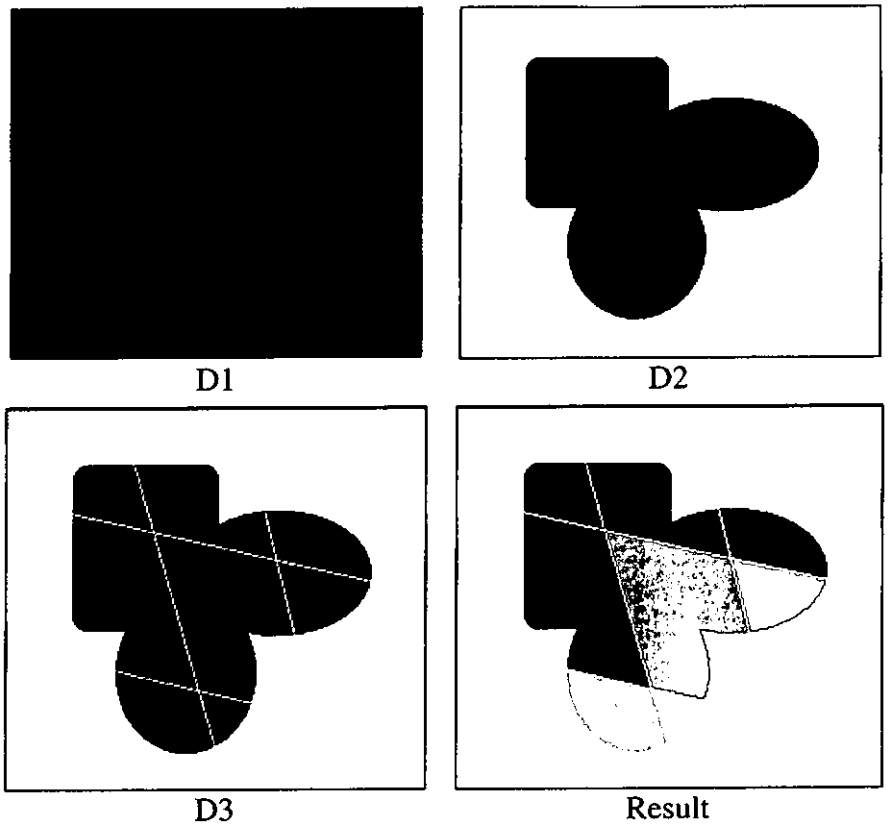


Figure 5.10: Region Extraction.

Table 5.2 shows the region extraction algorithm (D3->Result). It finds each connected area isolated by highways and marks them. If one point of certain area is found, it will find the adjacent points recursively until all points in the area are all marked.

```
AreaCount=0;                    // Counter of Regions

for ( Y = 0; Y < D3->Height; Y++)
{
    for ( X = 0; X < D3->Width; X++)
    // Scan each point of matrix D3
    {
      if (D3[Y][X] is a legal point && have not been marked)
      {
        AreaCount++;

        mark D3[Y][X];
```

```
            CurrentPoint= D3[Y][X];
            push D3[Y][X] into stack S (count);
            while (S != Null)
            {
              scan the four adjacent points D3[Y'][X'] of
              CurrentPoint in clockwise order;
              if (there is a D3[Y'][X'] that is a legal point && have
                 not been marked)
              {
                 mark D3[Y'][X'];
                 push D3[Y'][X'] into stack S;
                 CurrentPoint= D3[Y'][X'];
              }
              else
                 CurrentPoint= S (count-1);
            } //End while
          } //End if
        } //End X
} //End Y
```

Table 5.2: Connected region extraction algorithm.

5.2.2 Generation and validity control

Streets divide local transport areas into relatively even blocks and provide access to the highways nearby. In most of the cases, streets appear in raster or raster-related forms. For each extracted region, we can define a bounding box.

Except for current region, other area in the bounding box will be regarded as illegal area. The raster template (or other template, if needed), is applied in this bounding box to generate streets. Any street crossing an illegal area will be deleted. There may be three kinds of street ends including the breakpoints that appear after validity control, as shown in Figure 5.11.

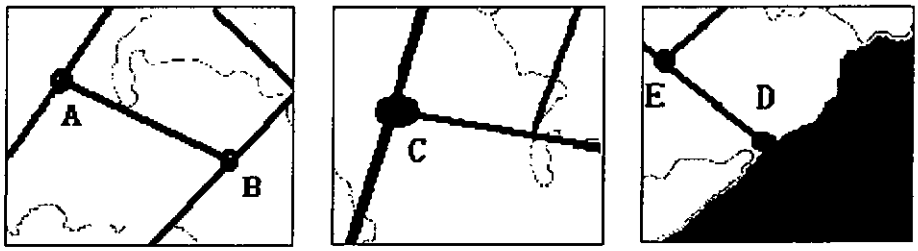
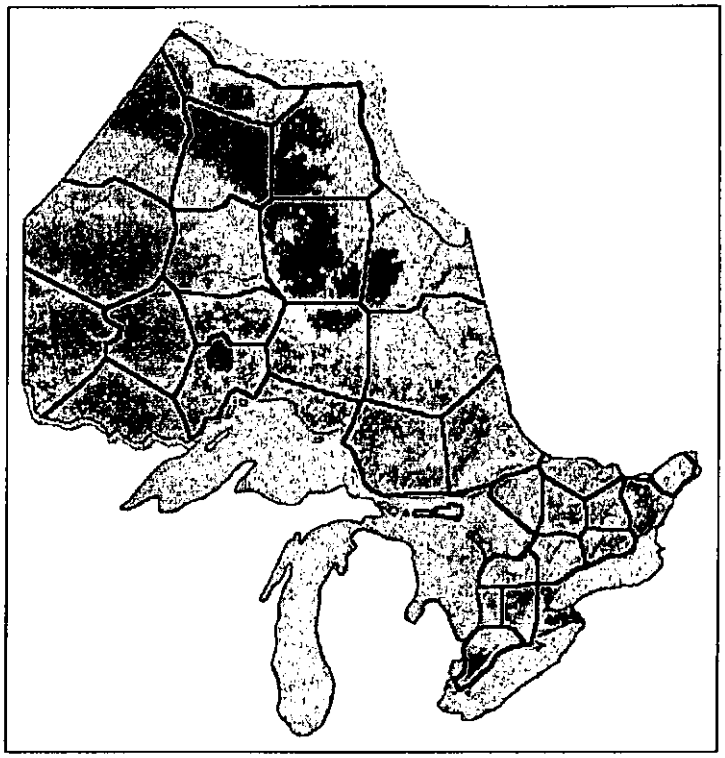


Figure 5.11: Three kinds of street ends.

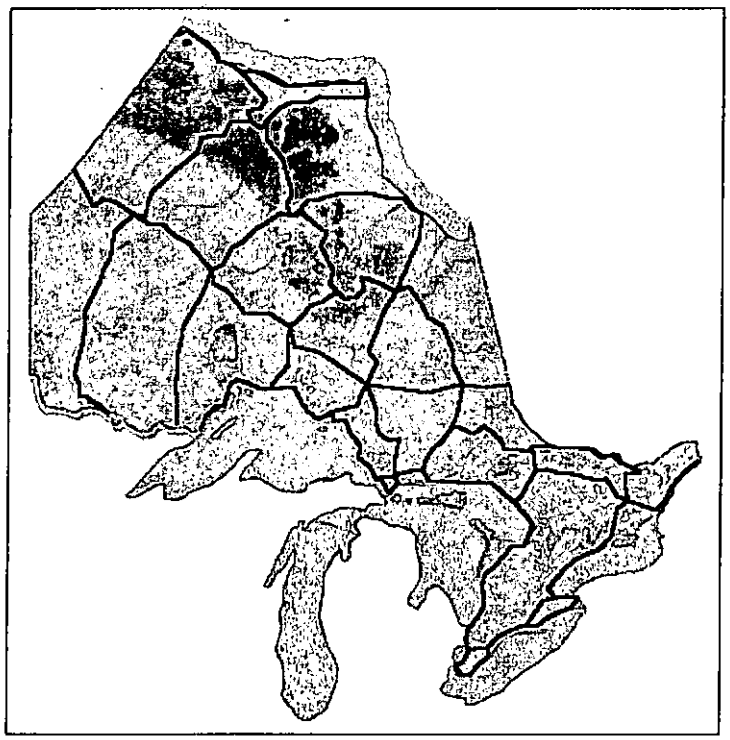
A and B are road ends that intersect other streets, C is the point joining the highway, and D ends at an illegal area. A, B, C are valid. As for D, if it end at land boundary, we simply delete this street segment DE.

5.3 Results of 2D road map

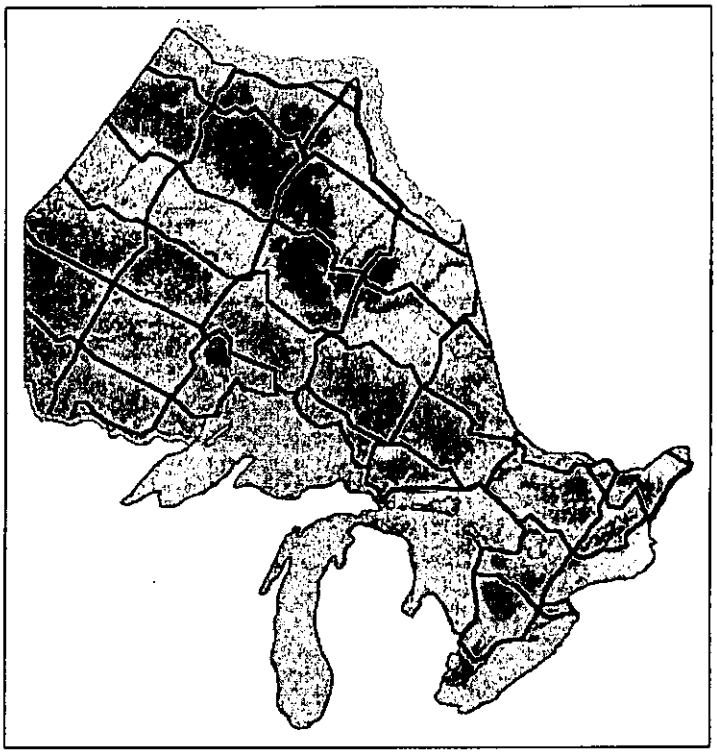
In this section, we represent a series of road networks generated by the 2D road map generation system and illustrate the effectiveness of our method by direct comparisons with actual patterns of road networks from real maps. Figure 6.12 shows four highway maps using the same input of boundary map, population map and elevation map of Ontario.



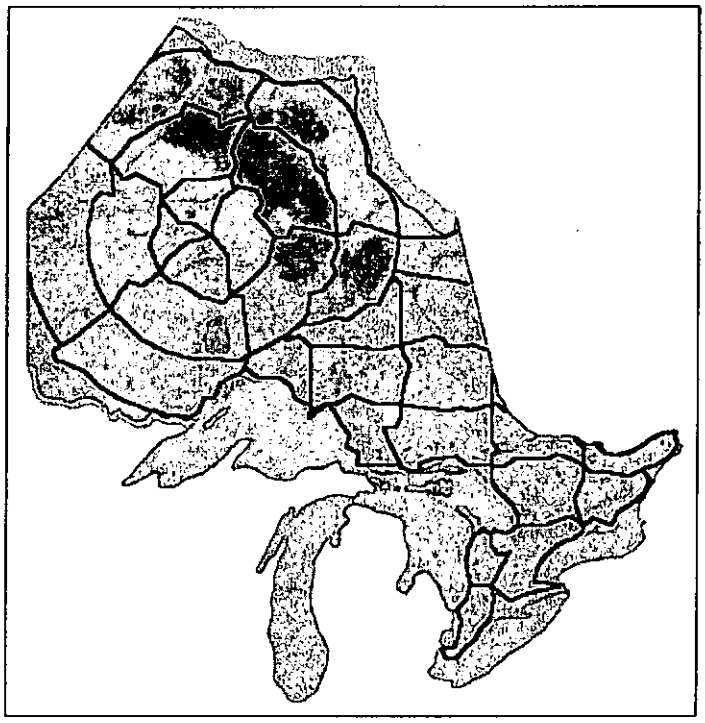
Population-based pattern



Radial pattern



Raster pattern

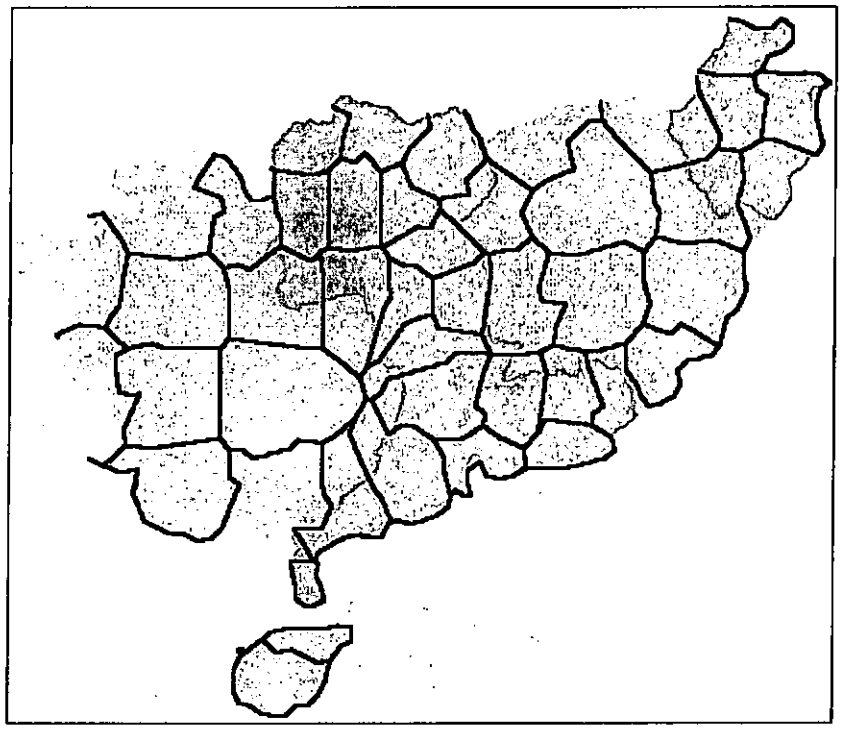


Mixed pattern

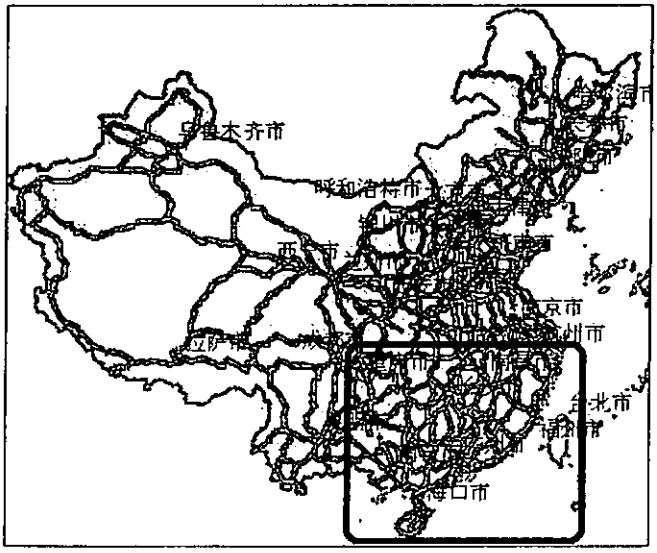
Figure 5.12: Four patterns of highway map of same input.

Figure 5.13 (a) shows a population-based highway map of several provinces in the southern part of China generated by our system and (b) is the

actual one of China. It can be seen that the two are very much similar in pattern that the roads in the middle part where the population is higher are denser. This shows that our system can take into account other factors and attributes that affect the dynamic growth of road networks, such as demographical data.



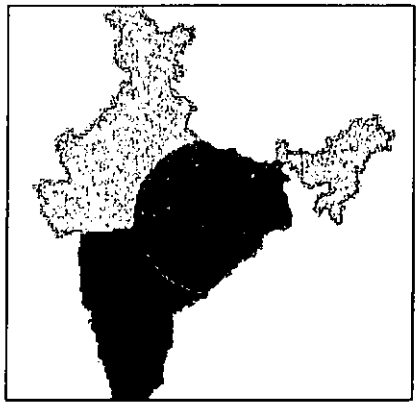
(a) Highway map of population-based pattern



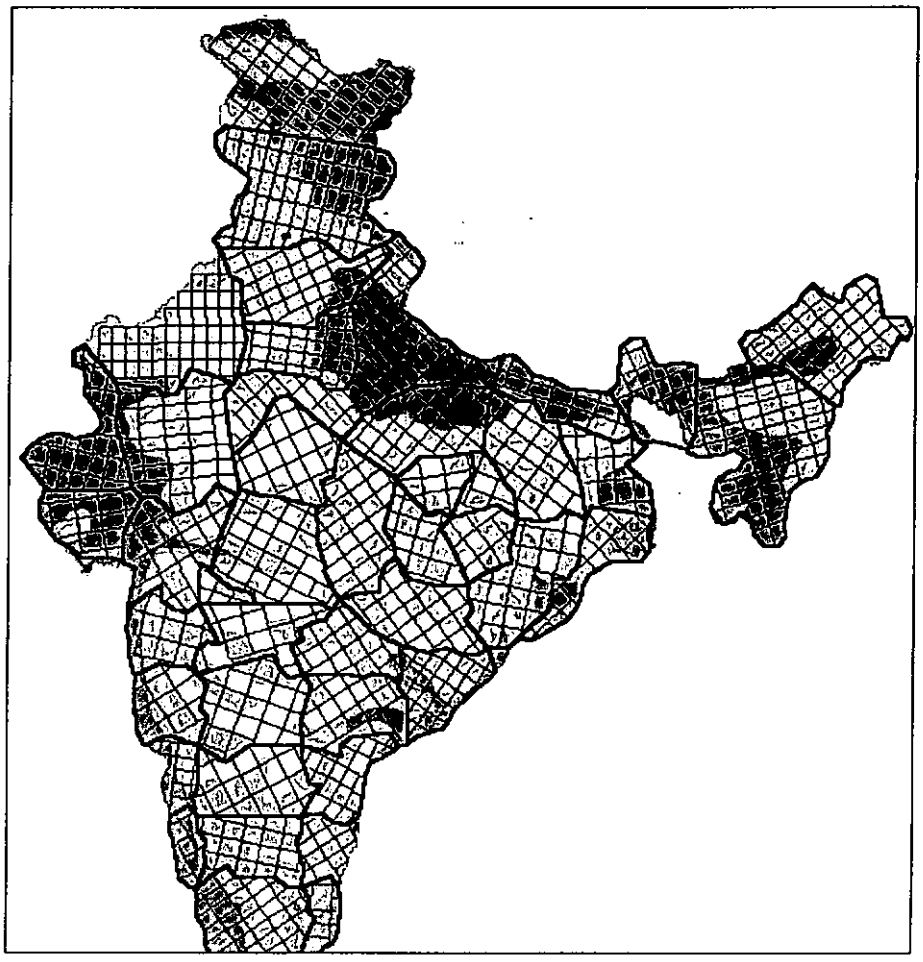
(b) Actual highway map of China

Figure 5.13 Comparison of highway map generated by system and actual one.

Figure 5.14 (b) is a road map of mixed patterns - the top part is population-based, the middle right part is radial and the bottom part is raster. The division of the three parts is given by the district map (a).



(a) District map



(b) Road map

Figure 5.14: Road map (highways and streets) of mixed pattern.

6 Road Expansion

The MAT provides us with a mechanism for finding the centers of the set of maximal balls representing a certain shape. It is best used in *thinning* a shape, or finding its *skeleton*. The topology of traffic networks forms the *skeleton* of road areas. The problem of generating road areas from a network of connected segments is the same as the problem of reversing the procedure for finding the medial axis of a point set to finding a shape from a given medial axis. The inverse problem has fewer constraints than the forward problem because the boundary points of the road shape are not known in advance. The following steps are employed for the inverse medial axis transformation problem:

Step1. Road sampling;

Step2. Road expansion;

Step3. Junction synthesis.

The first two steps operate on individual road segments and the third step handles the junctions formed at the intersections of road areas.

6.1 Road Sampling

One road segment is defined by the starting point (x1, y1, z1) and the end point (x2, y2, z2). In order to simulate the rugged road due to elevation variation, each road is sampled separately at a given rate (shown in Figure 6.1). The sampling rate decides the detail level of roads.

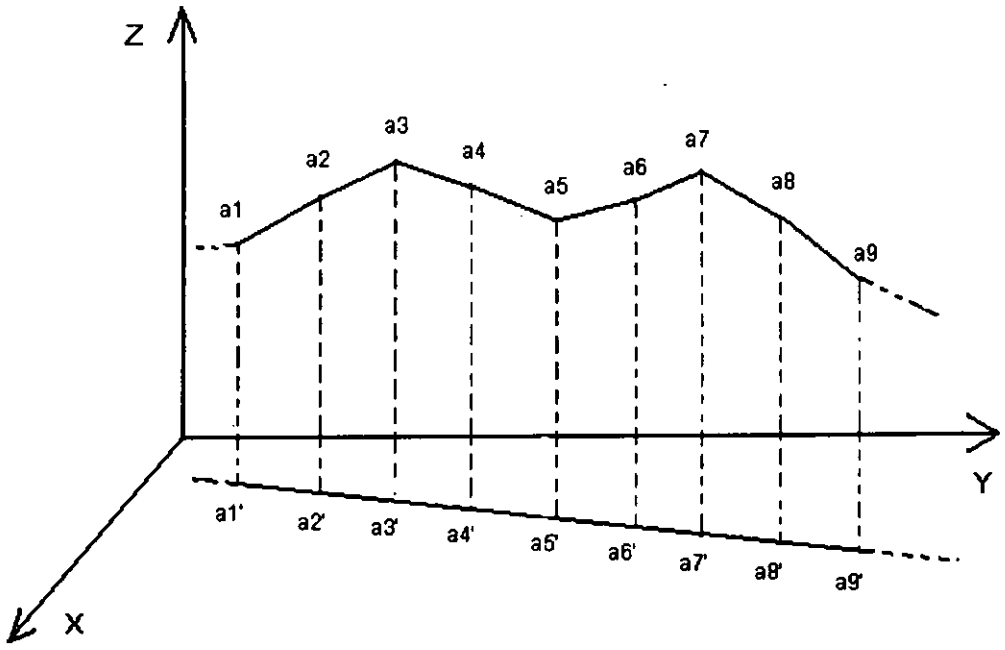


Figure 6.1: Sampling a road structure.

The x and y coordinates of these sampling points are determined by the starting and end point of the road while the z coordinates, which represent the elevation information at given points, is collected from the map. The method of acquiring information from maps has been discussed in Chapter 3.

6.2 Road Expansion

Using sampling points as centers of maximal balls, we expand each road on both sides according to the width (the radius of the maximal balls) given by type (highway, street, etc). Figure 6.2 shows one example of road expansion. Each road is composed of consecutive road sub-planes. These sub-planes are defined by sets of parallel lines which are parallel to the horizontal plane and vertical to both road sides.

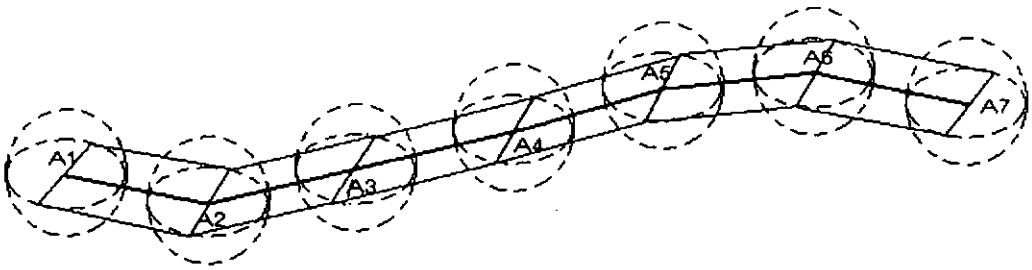


Figure 6.2: Example of road expansion.

6.3 Junction Synthesis

At the junction of different roads, the road sub-planes are no longer consecutive because, as shown in Figure 6.3, a_1a_2 is not parallel to b_1b_2 due to the angle between their projection lines on the horizontal plane.

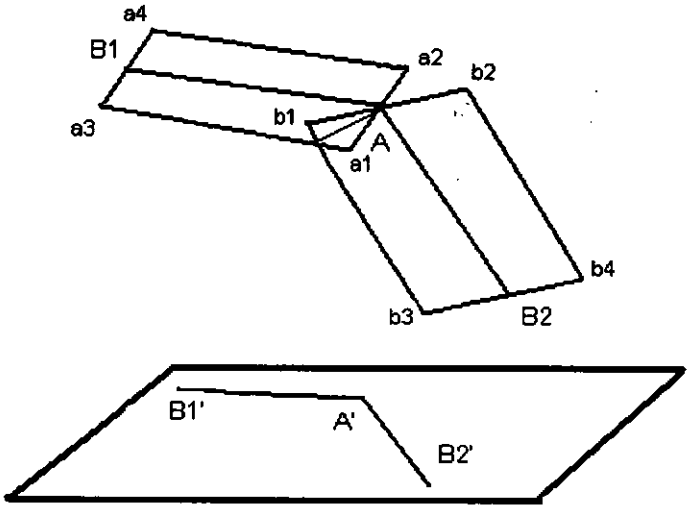
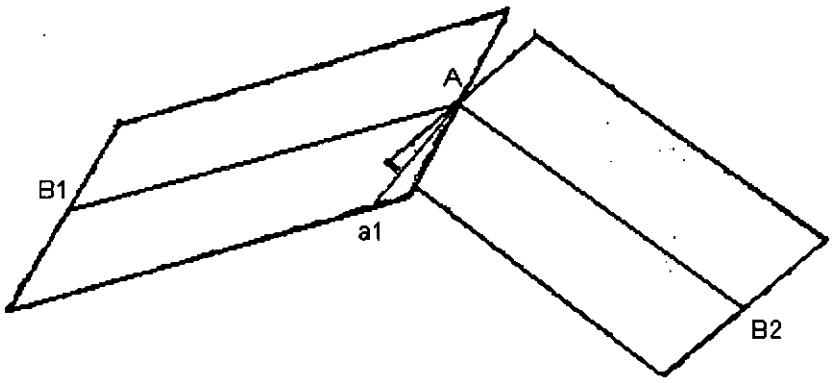


Figure 6.3: Misalignment of road sub-planes at the junction.

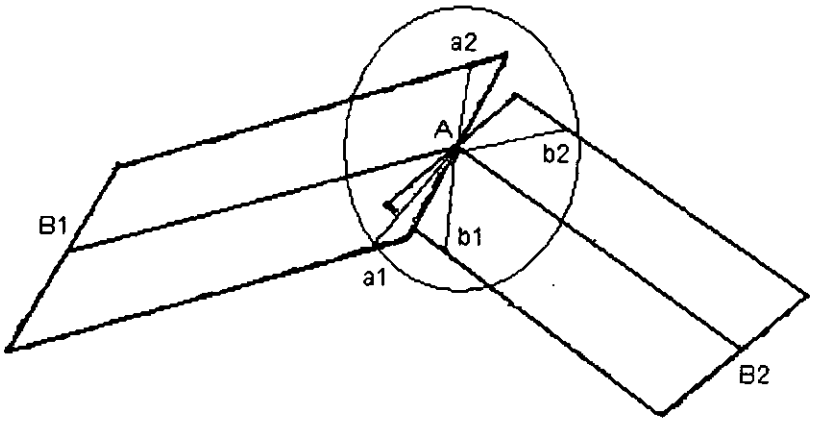
6.3.1 The generalized method

For junctions, we operate on the *nearest sub-planes* adjacent to the different roads. In Figure 6.4, we use a two-way junction to illustrate the process of junction manipulation. Firstly, we compute the intersection line between plane AB_1 and AB_2 which will intersect with both planes on the side border

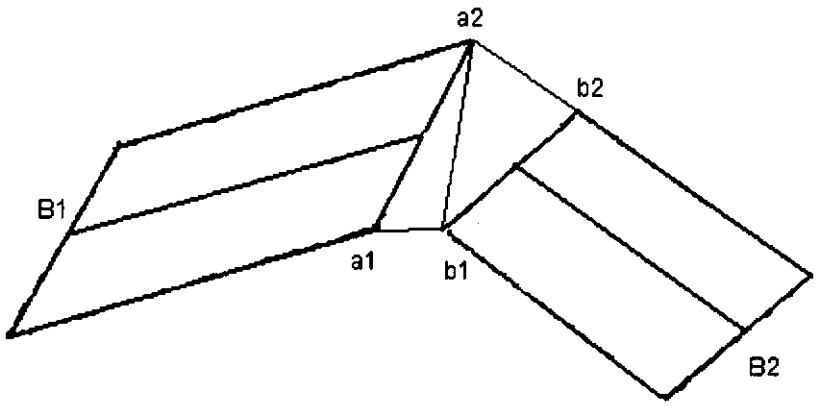
(Figure 6.4(a)). Secondly, we take the longer distance between the two borders as the radius and original junction point as the center to draw a ball. The ball will intersect both sides of both planes at four points a_1, a_2, b_1, b_2. The segments a_1a_2 and b_1b_2 are still vertical to their original plane sides and parallel to the horizontal plane (Figure 6.4(b)). The junction area consists of two planes decided by $a_1b_1a_2$ and $b_1b_2a_2$. The aim of this connection is to maintain the consistency and parallelism of lines on road plane (Figure 6.4(c)). As A is the center of the ball, the distances from A to a_1, a_2, b_1, b_2 are equal. A is still the center of this junction area.



(a)



(b)



(c)

Figure 6.4: Process of two-way junction.

The process of multi-way junction is similar to the two-way junction (Figure 6.5). First, we need to sort all the sub-planes in counter-clockwise order. Then, compute the candidate radius for the ball in pairs and get the biggest one. The ball will intersect each plane at two points.

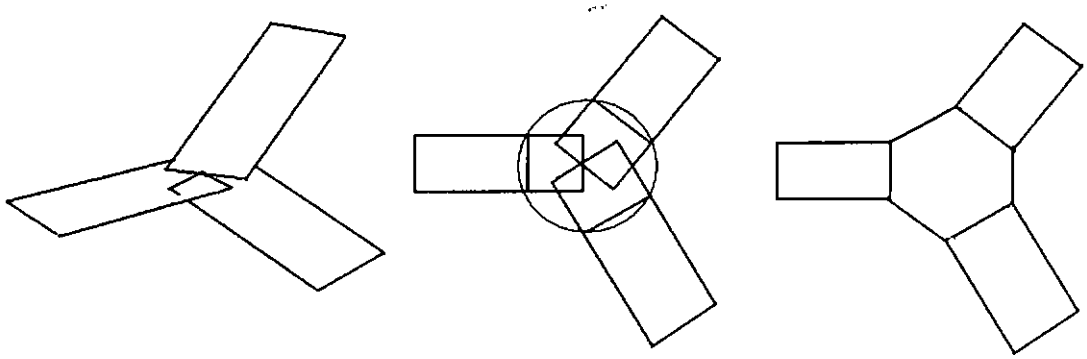


Figure 6.5: Process of multi-way junction

The connection of the junction area can be obtained by the following algorithm shown in Table 6.1 (n is the number of vertices in the junction area):

```
for(i = 0; i < ceil(ln(n))-1 ; i++)
    for(j = 0; j < n ;)
    {
        a = j;
        b = j + pow(2,i);
        if( b >= n)
          break;
        c = j + pow(2, i+1);
        Triangle (a , b, c mod n) ;
        j = c;
    }
```

Table 6.1: Junction Connection Algorithm.

For $A = \{a_0, a_1, a_2, a_3, a_4, a_5\}$ in Figure 6.6, we can get the triangle set $T = \{\{a_0, a_1, a_2\}, \{a_2, a_3, a_4\}, \{a_4, a_5, a_0\}, \{a_0, a_2, a_4\}\}$ in sequence according to above method.

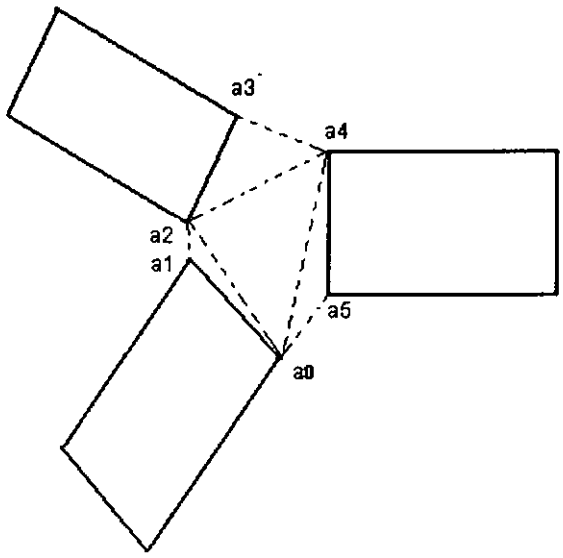


Figure 6.6 Connection of Multi-way junction area.

6.3.2 A modified method

In some cases, the radius of the ball becomes large. This leads to a large junction area that is not very common in real road intersection patterns. In this section we describe a modified method based on thresholding the ball radius for the IMAT.

Let the threshold be $\sqrt{2}/2$ * (the largest width among all the roads connected to the junction). When the radius of the ball is bigger than the threshold, we set it equal to the threshold.

Analyzing the road in 3D space, we can find that when the angle between the projection of two roads on the XOY plane is an obtuse angle (AB_1 and AB_{22} shown in Figure 6.7), four correct intersection points can always be got by intersecting with the ball which has the radius no bigger than the threshold. If the angle between is acute (AB_1 and AB_{21} shown in Figure 6.7), then an exception may appear.

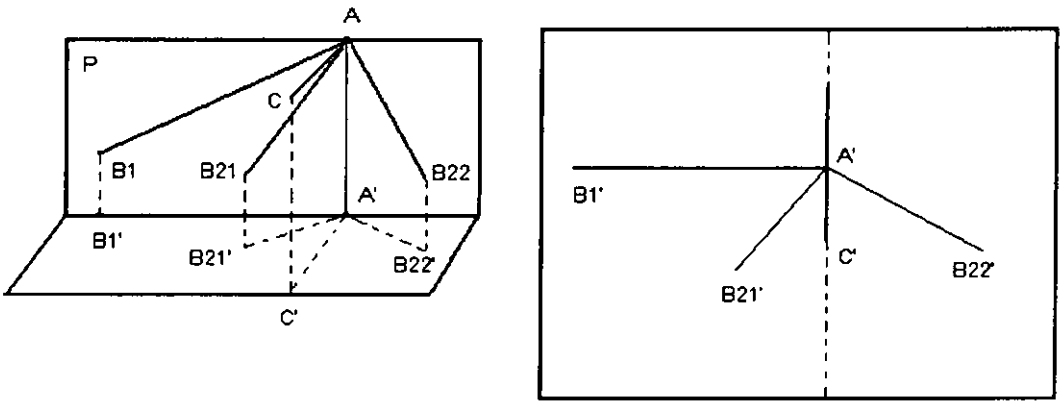


Figure 6.7: Analysis of Road in 3D space.

6.3.3 Exception handling

Let C_1, C_2 be points on the borders of the two planes whose projection on the XOY plane is the same point C (Figure 6.8 (a)). If the two line segments a_1a_2 and b_1b_2 coming from projection of the four points intersected by the ball do not intersect on the project plane, which means the junction area is not twisted (Figure 6.8(b)), link the junction area in the same way introduced above. Otherwise, compute the midpoint C' on the segment C_1C_2, lower or higher the two sub-planes to let them meet at C'. We link the junction area as shown in Figure 6.8(c).

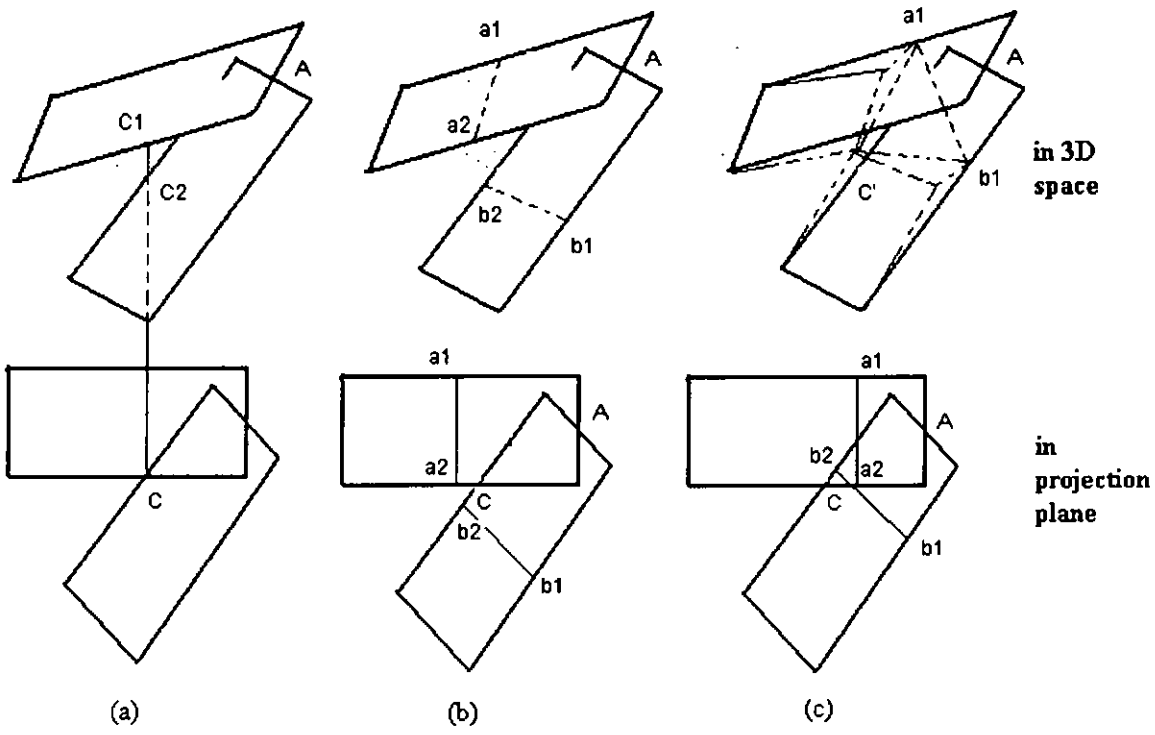


Figure 6.8: Cases for roads having an acute angle.

6.3.4 Junction smoothness

In real world, the junction area is always smoothed by human. Take Figure 6.4(c) for an example, using the average value of z_a, z_b (the original z value at $a_1, a_2; b_1, b_2$ respectively) to replace the original ones, which means that we lower or heighten the ground to make the junction area a plane parallel to the horizontal plane. This approach can also be generalized to the multi-junction areas.

6.4 Results of Road Expansion

Figure 6.9 and Figure 6.10 show two expanded road networks elevated on 3D terrain maps based on our two models generated in Chapter 5. Since the maps we used are not precise enough, the sampling rate is comparatively low. Elevation values on the sampling points due to noise in the map are computed from neighboring points, our system is still robust enough in dealing with crude maps.

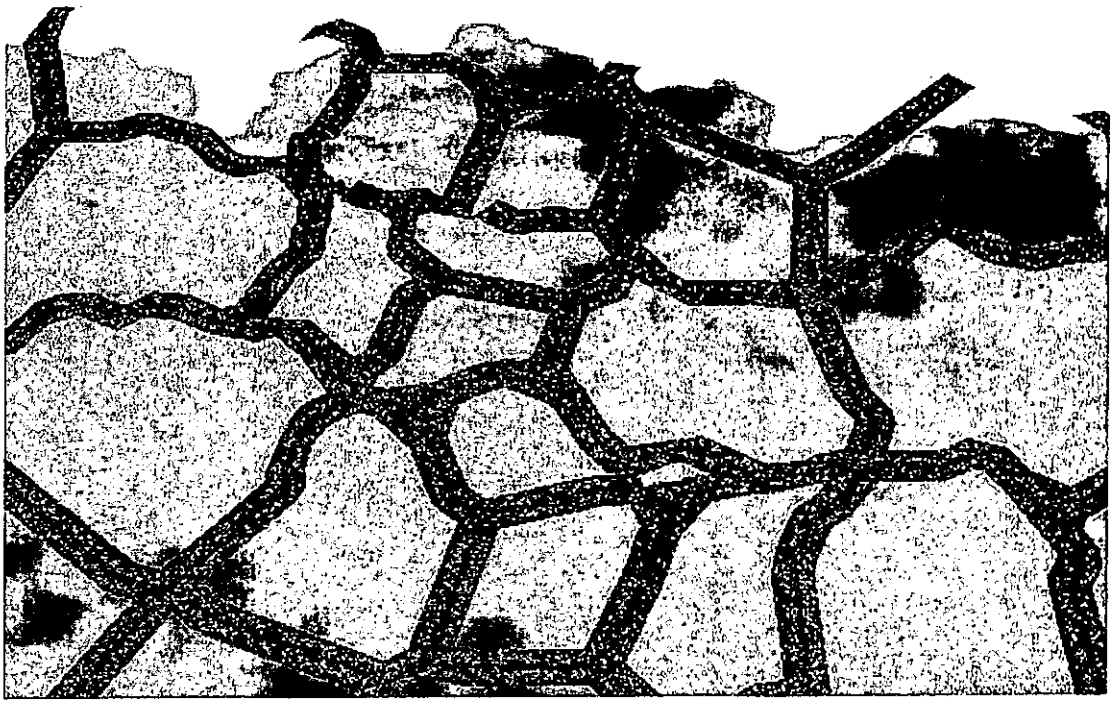


Figure 6.9: Expanded population-based road network.

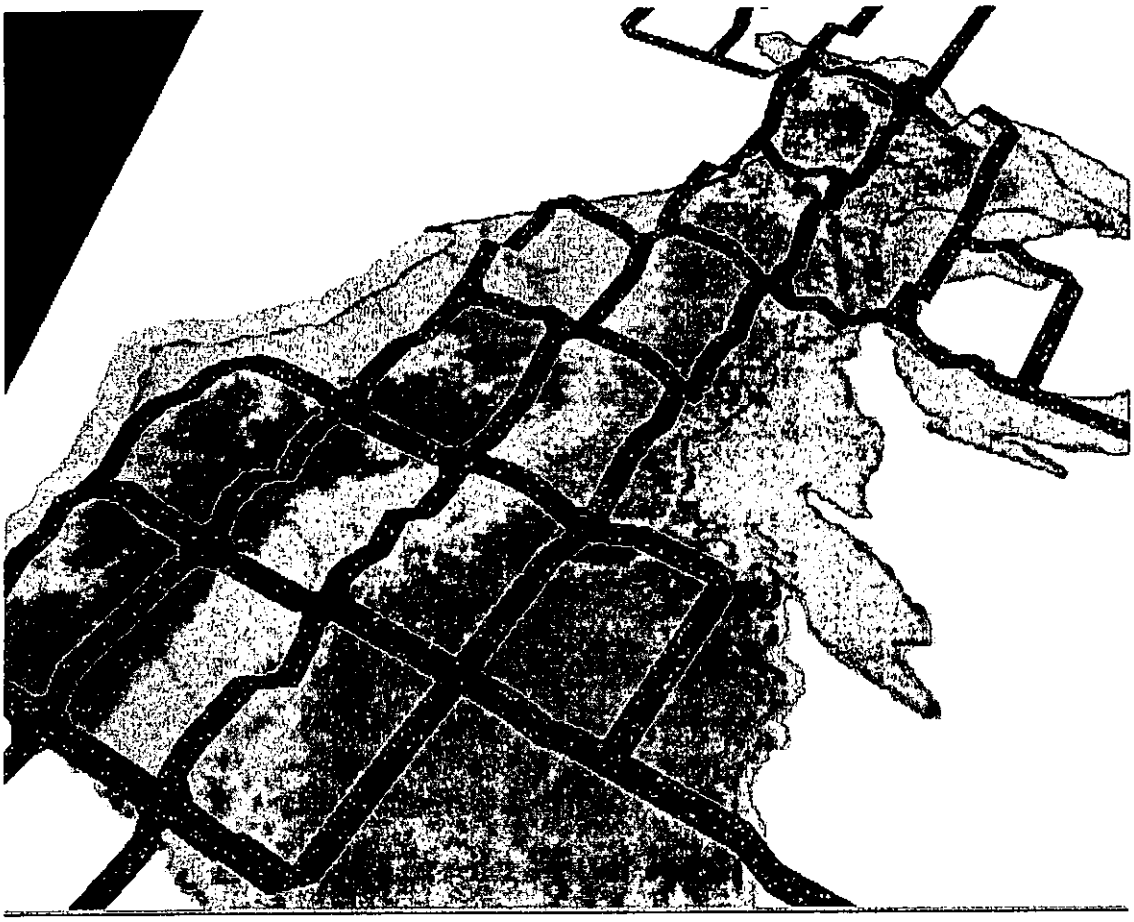


Figure 6.10: Expanded raster-like road network.

7 Terrain Simulation

After generation of 3D road network system, which is a part of city terrain, the other surfaces are still blank. In this chapter, the terrain of the city will be simulated to achieve a complete surface city model. There are several problems that should be solved in the terrain simulation:

1. Interpolation of terrain elevation data;

2. Collision detection between terrain surface and road plane;

3. Triangulation of terrain polygons.

7.1 Interpolation of Terrain Elevation Data

Elevation map is part of control data, which plays an important role in the generation of transportation system. In the elevation map, different colors or gray scales are used to represent different elevation values of areas. Most of elevation maps are usually 8-color or 16-color images with standard color scales and corresponding interpretation information of the scales. In the previous section, we have introduced an interface used to establish the relationship between color scales and data values. However, it can not provide sufficient information for the terrain simulation.

For crude maps, the elevation is represented by only several big color blocks which results in the great vertical variation in elevation in the boundary of different blocks. To make up for this defect, an effective method for interpolation of elevation data is very important.

A triangular interpolation method is designed to generate approximate city elevation data. This method is feature-point-based. Feature points, which can also be called as seeds, are extracted from available maps or assigned by users, and can be classified as two kinds: elevation points and zero points. Elevation points are those that will affect the whole structure of the terrain, such as the crest or trough points of the elevation variation. While zero points give the boundary and the illegal areas. The elevation of all the points outside the city or in the illegal area is regarded as zero.

There are two assumptions. First, we assume that it is a gradual variation in elevation from one surface point to another, which means the curve between two points on the tangent of surface are continuous. Second, we assume that the elevation value of a point is related and restricted by some points adjacent to it, and there is no great change or holes on the surface of earth. Because the distribution of feature points is irregular, we propose a triangular interpolation method for the generation of elevation data. If three most close points are used to estimate the elevation value of point, it is a triangulation-based generation method and the feature points will be the input points of triangulation algorithm, described in the following steps:

Step 1: Extract the elevation points from the elevation maps; Get zero points from the location maps.

Step 2: Using the elevation points as input, do Delaunay triangulation in 2D plane.

Step 3: For each point without known elevation value, estimate its elevation value using interpolation

algorithm below.
Step 4: Save all the interpolated elevation value as images.

Table 7.1: Triangular interpolation processes.

According to the relationship of point *P*, the Delaunay triangles decided by elevation points, and zero points, there are four possible cases when doing triangular interpolation.

(1) Case 1.

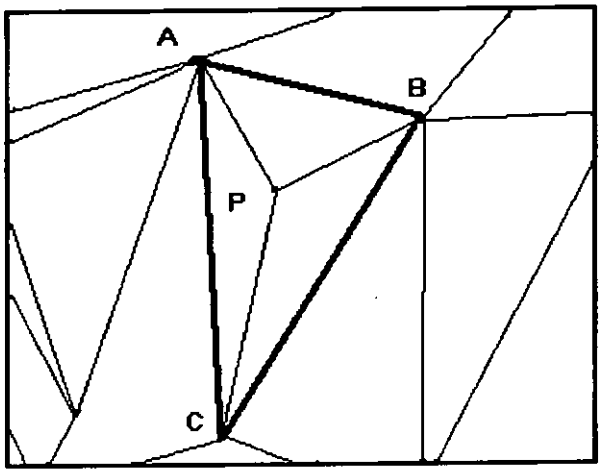


Figure 7.1: Case 1 of triangular interpolation.

Point *P* lies in a Delaunay triangle *ABC*, and there is no zero point in it (shown in Figure 7.1). Only the three vertices of *A, B, C* will define the elevation value of *P*. Then

$$P_{ele} = w_a \times A_{ele} + w_b \times B_{ele} + w_c \times C_{ele} \qquad (7\text{-}1)$$

where $w_a = \frac{area\ \ of\ \ sub-triangle\ \ a}{total\ \ area}$, $w_b = \frac{area\ \ of\ \ sub-triangle\ \ b}{total\ \ area}$,

$w_c = \frac{area\ \ of\ \ sub-triangle\ \ c}{total\ \ area}$. *sub-triangle a, b, c* are ΔPBC , ΔPCA , ΔPAB which faces *A, B, C* respectively.

(2) Case 2.

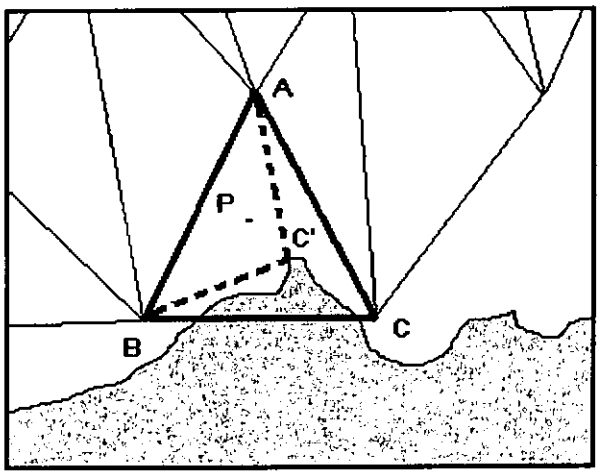


Figure 7.2: Case 2 of triangular interpolation.

Point *P* lies in a Delaunay triangle *ABC*. *C'* also lies in *ABC* and is the nearest zero point to *P* (shown in Figure 7.2). *A*, *B* and *C'* make a new triangle in which point *P* lies in. The elevation value of point *P* will be affected by three vertices of *ABC'*. The computation of P_{ele} is the similar to case 1. Because C_{ele}' equals 0, it can be simplified as

$$P_{ele} = w_a \times A_{ele} + w_b \times B_{ele} \qquad (7\text{-}2)$$

$$w_a = \frac{area\ \ of\ \ sub-triangle\ \ \Delta PBC'}{total\ \ area},\ w_b = \frac{area\ \ of\ \ sub-triangle\ \ \Delta PC'A}{total\ \ area}.$$

(3) Case 3.

Point *P* lies outside the convex hull which means it does not lie in any Delaunay triangles (shown in Figure 7.3). Edge *AB* is the nearest Delaunay edge to point *P*, *C* is the nearest zero point to point *P*, and *P* lies in triangle *ABC*. The elevation value of point *P* will be affected by three vertices of *ABC*. Then:

$$P_{ele} = w_a \times A_{ele} + w_b \times B_{ele} \qquad (7\text{-}3)$$

$$w_a = \frac{area\ \ of\ \ sub-triangle\ \ \Delta PBC}{total\ \ area},\ w_b = \frac{area\ \ of\ \ sub-triangle\ \ \Delta PCA}{total\ \ area}.$$

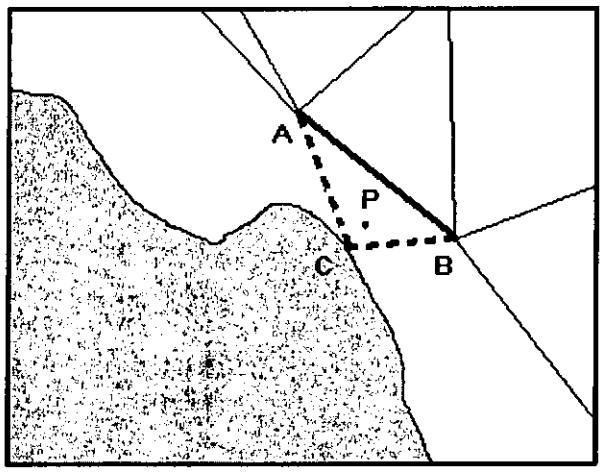


Figure 7.3: Case 3 of triangular interpolation.

(4) Case 4.

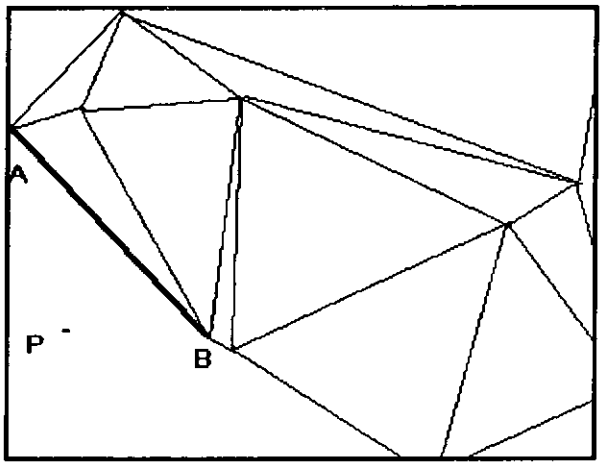


Figure 7.4: Case 4 of triangular interpolation.

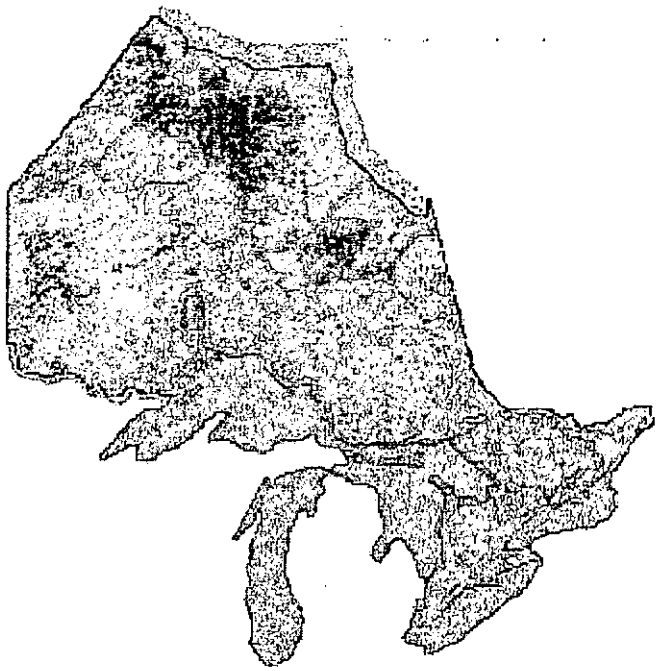
Point *P* lies outside the convex hull, and there is no zero point that can make a triangle with a Delaunay edge to surround point *P* (shown in Figure 7.4). Edge *AB* is the nearest Delaunay edge to point *P*, and the elevation value of point *P* will be affected point *A* and *B*. Then

$$P_{ele} = w_a \times A_{ele} + w_b \times B_{ele} \quad (7\text{-}4)$$

where $w_a = \frac{PB}{PA+PB}$, and $w_b = \frac{PA}{PA+PB}$. PA /PB is the distance between P and A /B.

In order to keep the data consistency of the system, all these interpolated elevation values should also be represented by images as control data. The Data Engine will save the elevation data as an image which has the same resolution as the other images of the city.

Figure 7.5 (b) shows the saved interpolated elevation image computed by our method. It is gray value based, brighter area means higher altitude. (a) is the original elevation map.



(a) Original elevation map.

(b) Interpolated elevation image.

Figure 7.5: Interpolated elevation image and original map.

7.2 Collision Detection between Terrain surfaces and Road Planes

7.2.1 Collision on 2D plane

Polygon is a common structure in terrain modelling. In this section, a regular mesh is used to partition the terrain into small rectangles in 2D plane.

3D road network and the terrain are generated separately. Therefore, collisions may occur in their overlapped areas. When handling collision, the priority of the road network should be kept, which means that the collision operation will not change the layout of roads. The collision should be handled by slight modification of the terrain.

The 3D road network is represented by polygons that can be divided into two classes. One is road sub-plane whose projection on the ground is a rectangle,

and the other is junction area whose projection on the ground is convex polygon. On the 2D plane, the terrain grids, which intersect the road polygon, are cut into one or more new polygons (shown in Figure 7.6). The collision operation becomes an arbitrary polygon clipping problem:

Road polygon -> Clipping window;

Terrain grid -> Polygon waiting for clipping.

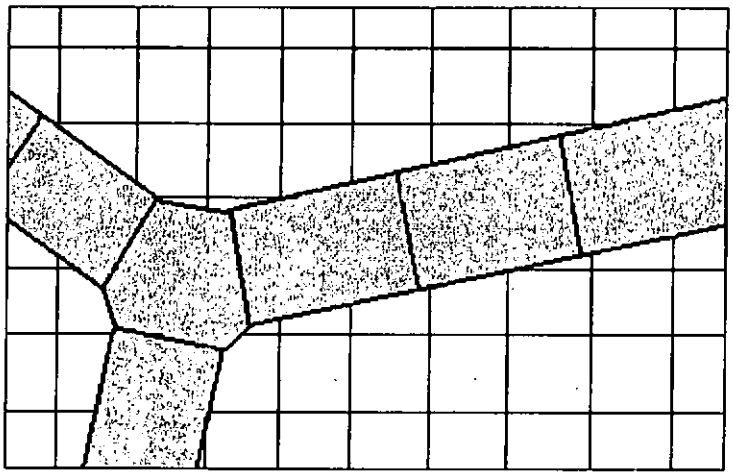


Figure 7.6: The projections of road segments on the 2D terrain grid.

Our collision problem has its own characteristics:

1. The clipping window may not be rectangular, but it is convex.
2. After clipping, the terrain polygon may be partitioned into more than one polygon; and what we needed are the polygons outside the clipping window rather than the inside part.
3. All elevation values of the vertices of the new polygons should be calculated and recorded during the processing.
4. The edges of clipping window and the terrain polygon may be overlapped.
5. Each edge of a clipping window will intersect with at most two edges of a terrain polygon.

6. There is no hole in terrain polygon.

7. All edges, which belong to the old terrain polygon and will make a new terrain polygon, will intersect the edges of clipping window only two times.

These characteristics may be seemed as constraints and may help to simplify the structure and flow of famous polygon clipping algorithms we have discussed in chapter 2. A new method derived from the traditional algorithms is designed for our own problem in order to improve the efficiency. In section 7.2.2, 7.2.3, we will talk about the data structure and edge type classification as preparation. The method will be discussed in detail in section 7.2.4.

7.2.2 Data structure in collision operation

Figure 7.7 shows the data structure – a bidirectional linked list, which is designed to represent *Polygon*: both the terrain rectangles and the clipping windows (the projection of road segments).

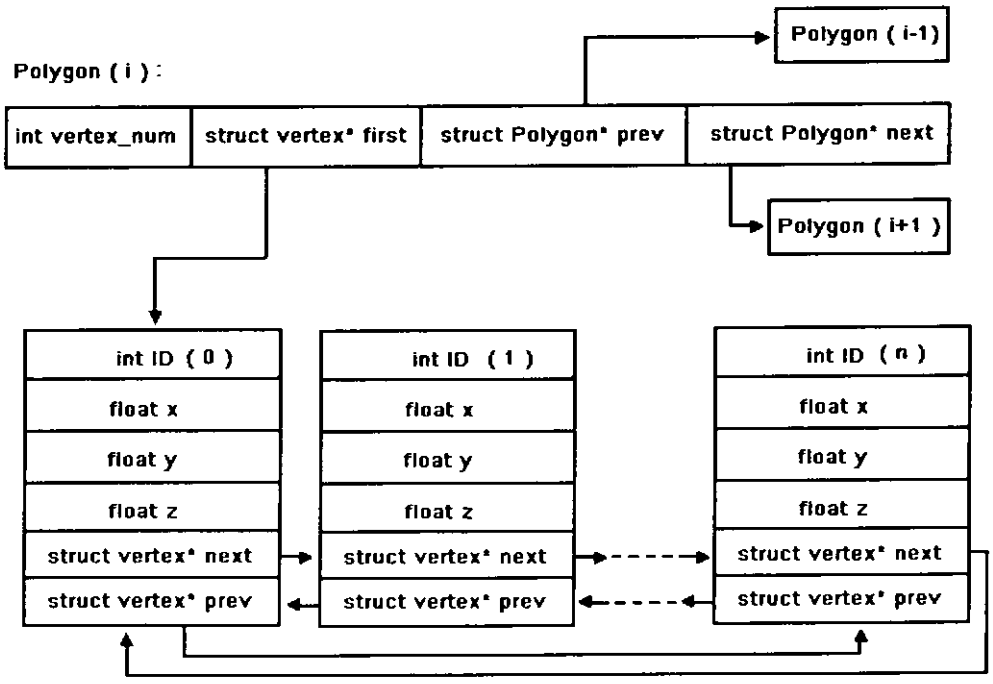


Figure 7.7: Data structure: Polygon.

All vertices of an arbitrary polygon are sorted in a counter-clockwise order beforehand, which means that the left side of each edge is inside. Figure 7.9 is the illustration of the data structures of clipping window A and terrain polygon B in Figure 7.8, in which c_1, c_2, c_3 and c_4 are intersection points of corresponding edges of A and B. The result polygons - $c_2 b_4 c_3$, $c_1 a_6 c_4 b_1$ are composed of edges which connect the intersection points, e.g., $c_3 c_2$ or edges out of clipping window, e.g., $c_2 b_4$.

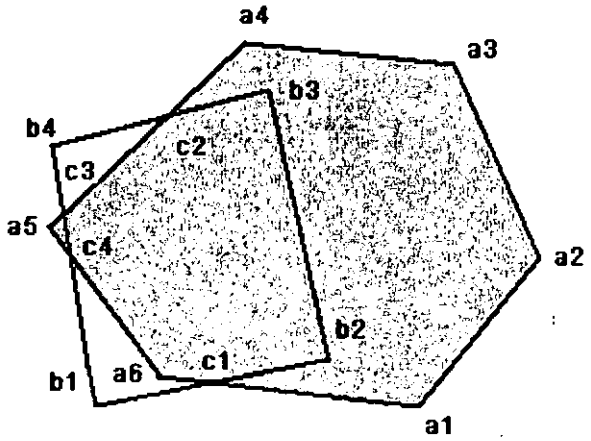


Figure 7.8: An example of clipping window A and terrain polygon B.

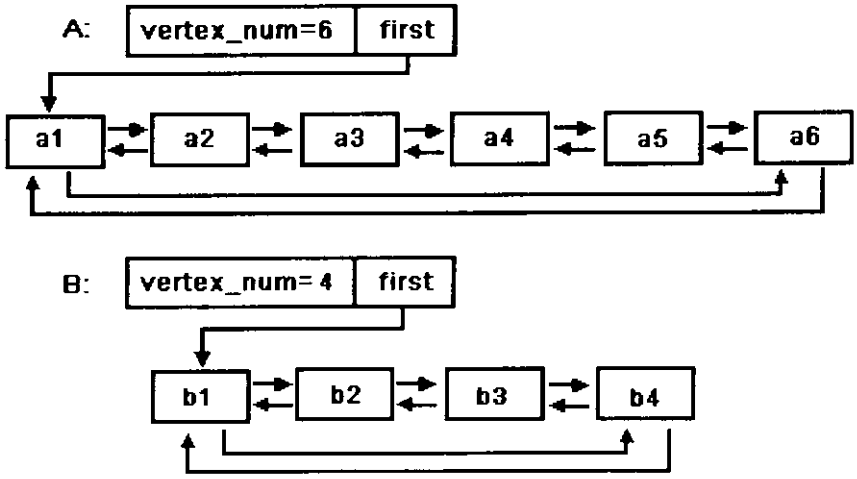


Figure 7.9: Data structure of clipping window A and terrain polygon B.

Besides the *Polygon*, another data structure *temPolygon* (shown in Figure 7.10) is designed to save temporary terrain segments cut by clipping window during the clipping procedure. A valid *temPolygon* has only one *head* and one *tail* which are Ids of vertices of a terrain polygon (There may be two heads and two tails in the clipping procedure but only one will be reserved after completion). Vertices from *head* to *tail* (also in counter-clockwise order), together with vertices saved in temPolygon which begin from the *first*, will make a new terrain polygon. Some of these temporary terrain segments are new terrain polygons and some should be discarded after clipping because they are invalid or incomplete to make a new polygon.

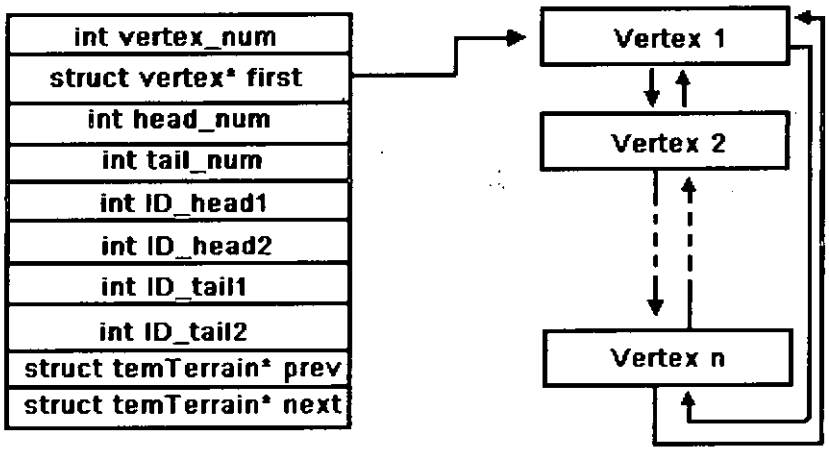


Figure 7.10: Data structure: temPolygon.

7.2.3 Classification of edge type

Vertices of a Polygon are in a counter-clockwise order, and each edge has its direction. Assuming Edge A_iA_{i+1} of polygon A intersects an edge of another polygon B, it may come into, go out of or goes through it. In order to classify the relationship of edge A_iA_{i+1} and polygon B, first, we need to determine if this edge intersects with any edge of polygon B. If intersects, then judge if the begin and end points of the edge are inside polygon B. The horizontal scan line algorithm is

used here to determine if a point inside an arbitrary polygon. After the two-step judgment, all edges of *A* can be divided into five general types, as shown in Table 7.2 which illustrates Figure 7.11.

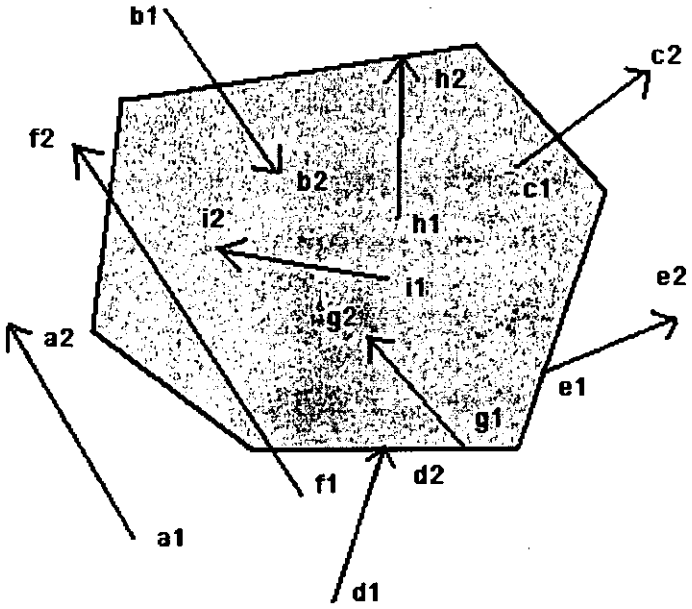


Figure 7.11: Illustration of relationship between clipping edges and polygon.
(Points *d2*, *e1*, *g1* and *h2* are on the edges of polygon)

Type	*Name*	*Intersects or Not*	*Begin Point*	*End Point*	*Example*
A	Outside-edge	Not	Outside	Outside	*a1a2*
		Intersects	Outside	On Edge	*d1d2*
		Intersects	On Edge	Outside	*e1e2*
B	In-edge	Intersects	Outside	Inside	*b1b2*
		Intersects	On Edge	Inside	*g1g2*
C	Inside-edge	Not	Inside	Inside	*i1i2*
D	Out-edge	Intersects	Inside	Outside	*c1c2*
		Intersects	Inside	On Edge	*h1h2*
E	Through-edge	Intersects	Outside	Outside	*f1f2*

Table 7.2: Types of clipping edges and examples.

7.2.4 Arbitrary window clipping method

For each clipping window, a searching area (bounding box) can be determined according to the positions of its vertices. Assume that terrain polygon T lies in the bounding box of clipping window C. It is not a sufficient condition that all vertices of T are outside the C to judge that T does not intersect with C. We have to check all clipping edges one by one. From the first edge c_0c_1, examine the type of each edge of C in counter-clockwise order. For type **A** to **E**, there is a controller leading to different sub-flows which is shown as follow (Figure 7.12).

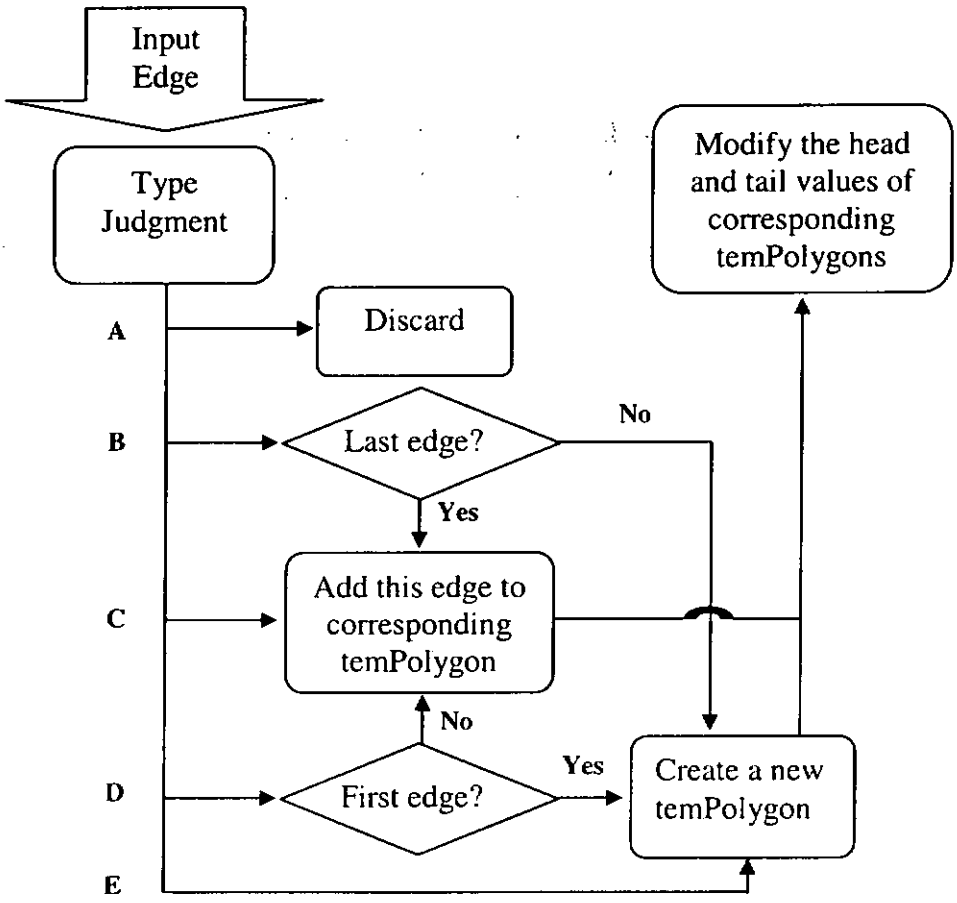


Figure 7.12: Flow chart of processing of different edge type.

Once a *temPolygon* is created, its two *heads* will be defined by end-points of the edge which intersects a type **B** clipping edge, also, the tail will be defined by end-points of the edge which intersects a type **D** clipping edge. A type **E** clipping edge intersect two edges of a terrain polygon, distances from the intersect points to the beginning point of clipping edge determine which end point is the *head* or *tail*.

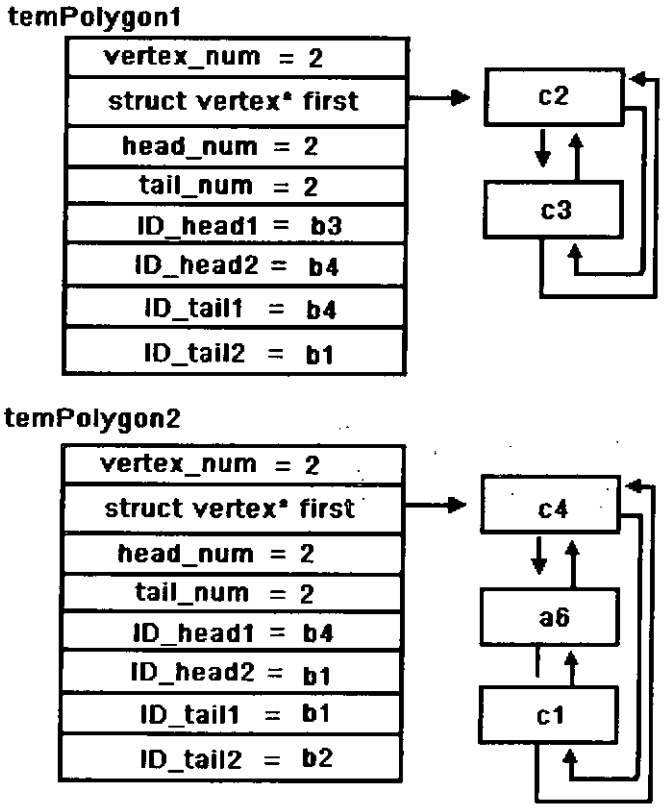


Figure 7.13: Result of the example described in Figure 7.8 after clipping.

A valid temPolygon has only one *head* and one *tail*. The redundant *head* or *tail* can be got rid of under two conditions: 1. A *head* or *tail* should be outside clipping window; 2. One *head* or *tail* is *tail* or *head* of another *temPolygon*. Figure 7.13 shows the result of the example described in Figure 7.8 after clipping. There are two *temPolygon* and each of them has two *heads* and two *tails*. b_2 and b_3 are inside clipping window, so they can be deleted. b_1 and b_4 are *tails* of

temPolygon1 which are the same as *heads* of *temPolygon2*. By comparing the distances from these two candidate *tails* to the intersect points c_3 and c_4, b_4 which has the shorter distance to c_3 is reserved as the *tail* of *temPolygon1* and *temPolygon2* has only one *head* b_1.

7.2.5 Special cases Handling

During the processing of collision operation, some special cases may trigger errors. We need to handle them differently.

(1) A vertex of terrain polygon lies on a non A-type clipping edge

Clipping edge c_1c_2 intersects two edges of a terrain polygon, t_1t_2 and t_2t_3, at vertex t_2 (shown in Figure 7.14). t_1t_3 will be regarded as a "curve" edge. When creating a new *temPolygon* structure or adding this clipping edge into a *temPolygon* structure, the vertices, t_1 and t_3, will be set as *heads* or *tails* of corresponding *temPolygon* structure.

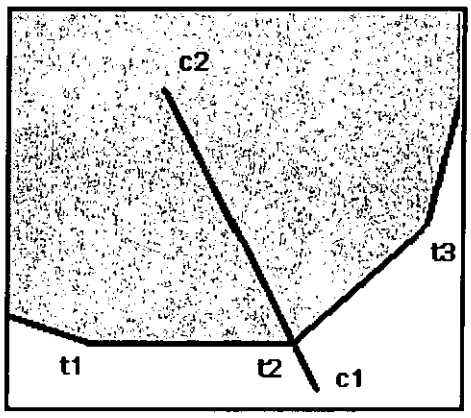
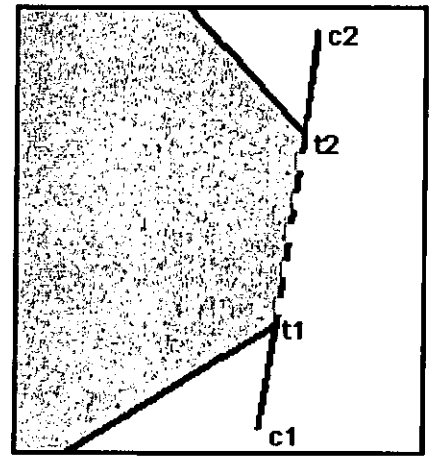


Figure 7.14: Example of A vertex of terrain polygon lies on a non A-type clipping edge.

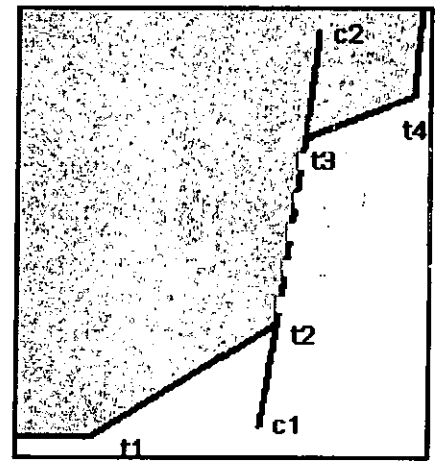
(2) Clipping edge and edge of terrain polygon overlap each other

There are three kinds of overlapping between clipping edge and edge of terrain polygon shown in Figure 7.15. For kind 1, the clipping edge will be regarded as a type A edge and be discarded. The overlapped segment of kind 2

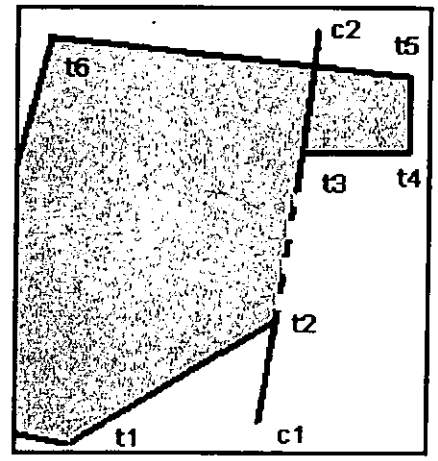
and kind 3, are assumed to shrink to one point t_2. Vertices t_2 and t_3 are superposition, and the vertices connected to them t_1, t_4 will be set as *head*s or *tail*s of corresponding *temPolygon* structure.



Kind 1



Kind 2



Kind 3

Figure 7.15: Clipping edge and edge of terrain polygon overlap each other.
(The gray areas are terrain areas, and the dash lines are overlapped segments.)

(3) Terrain polygon is totally hidden by clipping window

If a terrain polygon is totally hidden by clipping window, it should be deleted. The conditions to determine the totally hiding are:

a. There is no intersection between the edges of the terrain polygon and clipping edges.

b. At least one vertex is inside clipping window.

(4) Clipping window "touches" the terrain polygon

If the clipping window "touches" the terrain polygon like demonstrated in Figure 7.16, the corresponding terrain plane will be changed even there is no clipping because of the value of touch point may different due to one coming out of the map for the clipping window while the other coming out of interpolation for the terrain polygon. If the touch point is vertex c_1 of clipping window, the terrain polygon should insert one vertex whose elevation equals to that of corresponding position on road segment which is related to c_1. If the touch point is vertex t_1 of terrain polygon, the elevation value of t_1 should be adjusted to that of corresponding position on road segment related to touch point.

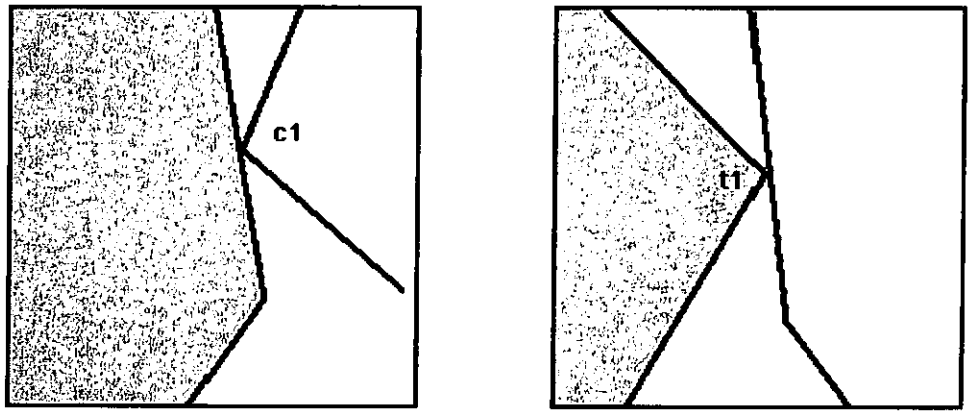


Figure 7.16: The clipping window "touches" the terrain.

Left: the touch point is a vertex of clipping window; right: the touch point is a vertex of terrain polygon.

7.3 Triangulation of Terrain Polygon

The projection on 2D plane being a polygon does not assure that in 3D these vertices is still on a plane. In order to render the terrain correctly, the polygons should be partitioned into triangles which will not overlap each other. The sequence of points should be observed to maintain the boundary information. This triangulation method could be done in the following steps:

Step 0: if (*vertex_num* of input terrain polygon t is less than three)

go to **Step 3**;

else set $t_0 = t \rightarrow first$.

$t_1 = t_0$;

Step 1: set $t_2 = t_1 \rightarrow next$;

$t_3 = t_2 \rightarrow next$

if (*vertex_num* equals to three)

create_triangle(t_1, t_2, t_3);

go to **Step 3**;

Step 2: if ($\angle t_1 t_2 t_3 < 180^\circ$) && (there is no other vertex of t lies in $\Delta t_1 t_2 t_3$)

create_triangle(t_1, t_2, t_3); *delete*(t_2);

set $t_0 = t_3$; $vertex\_num--$;

else

set $t_0 = t_2$.

Go to **Step 1.**

Step 3: Return all created triangles.

Table 7.3: Triangulation of Terrain Polygon.

According to the triangulation method, the triangles that are created orderly in the example demonstrated in Figure 7.17 should be $\Delta t_2 t_3 t_4$, $\Delta t_5 t_6 t_7$, $\Delta t_7 t_8 t_1$, $\Delta t_2 t_4 t_5$, $\Delta t_5 t_7 t_1$, and $\Delta t_1 t_2 t_5$. Note that the contour of $t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8$ is still maintained.

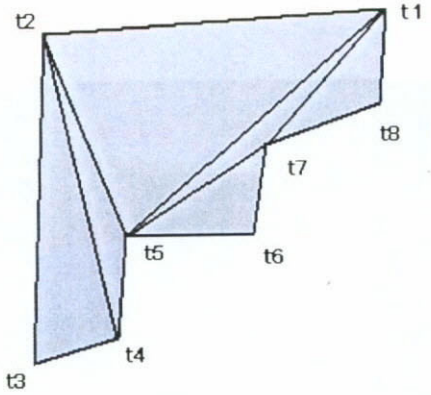


Figure 7.17: Example of terrain polygon triangulation.

7.4 Results of Terrain Simulation

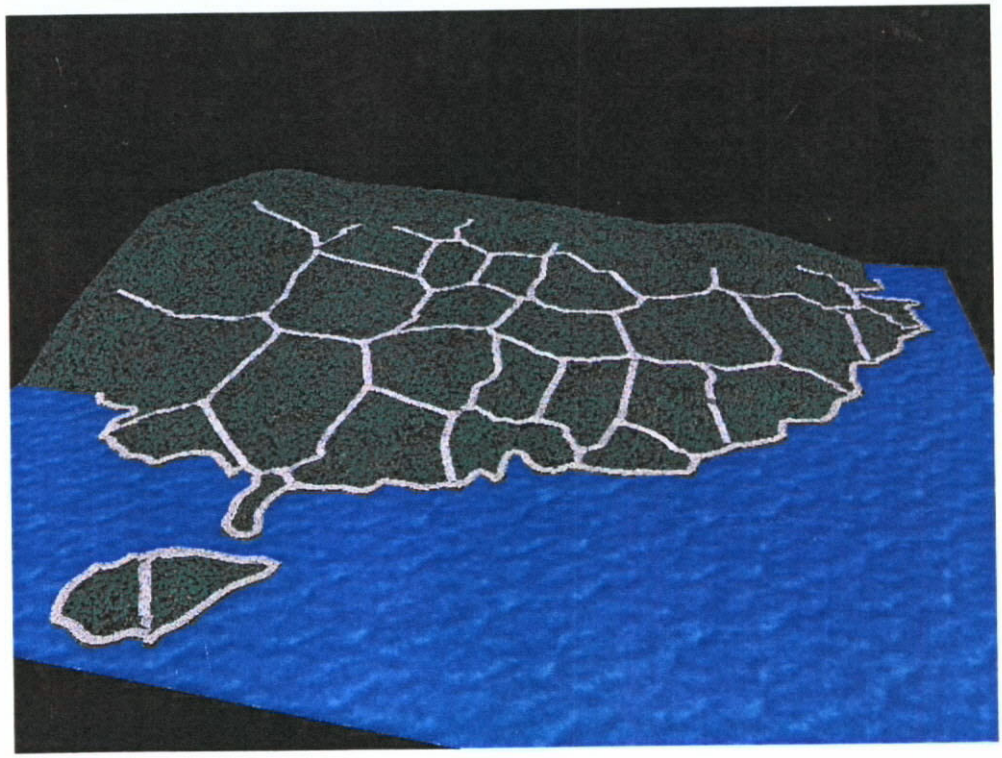


Figure 7.18: Integrated road network of South China.

Figure 7.18 shows the result of 3D model of south China with the road network shown in Figure 5.13. The blue area is ocean and the green area is terrain. The breakpoints above are land boundary breakpoints which connect roads of other regions.

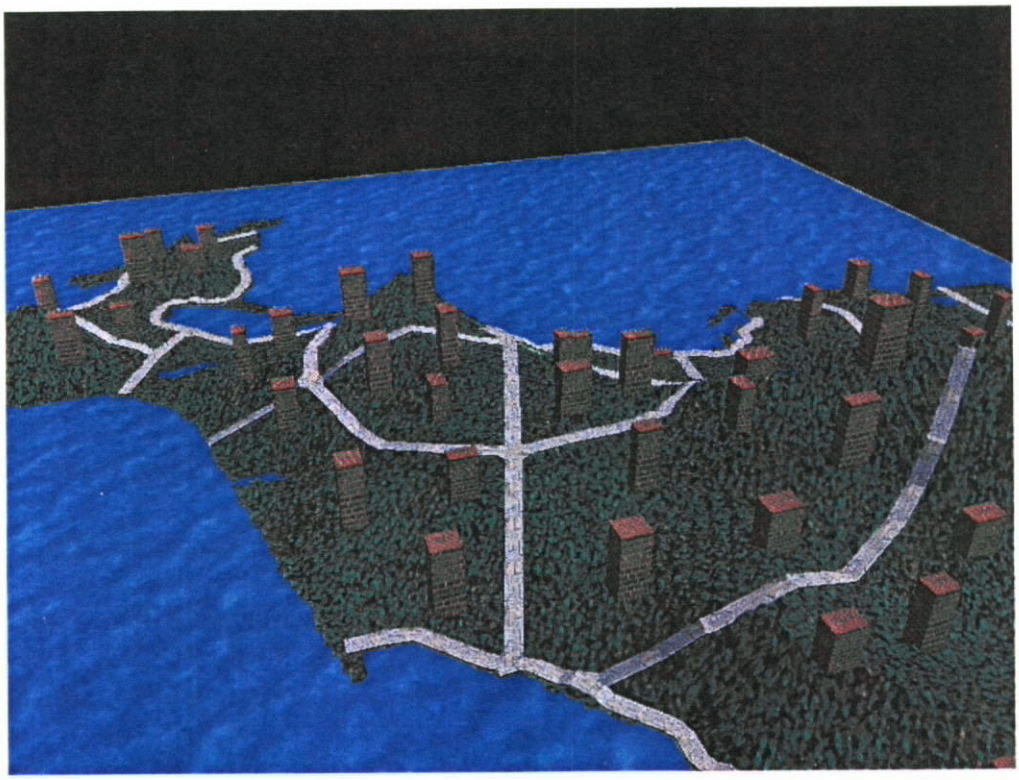


Figure 7.19: Blended terrains and roads.

Figure 7.19 shows a radial example of Ontario with houses built on the terrain. The detailed roads can be seen in this figure.

8 Conclusion

We conclude this thesis with a review of where we have been and a proposal of where we should go.

8.1 Concluding Remarks

In this thesis, we have presented a whole system for image-based urban model generation with the focus on road network construction.

With templates, road maps of different patterns can be generated from the same input data. Our system also allows pre-designs and alteration of the final road network synthesized. These provide the user with various editing choices and considerable flexibility in customization. This system has been designed as a scalable kernel for synthesizing large 3D models of urban environments with application to 3D games and VR. We must emphasize that this is a road network generating system rather than a design system in which urban planners employ drafting tools to create road structures. Our system is a constraint-based automatic synthesis tool that provides a quick road network prototype taking into consideration geographical constraints and demographic data in the attempt to simulate the growth of road networks in real environments. After auto-generation of 2D road map, the roads are expanded into 3D space. After terrain simulation of the whole urban area, the roads and terrains are blended together to form a complete urban surface model.

Compared to related work done by Kato and Parish, (1) our system is easy to incorporate future extensions of new road patterns by adding new road templates into the template library. (2) It is built on 3D terrains rather than a plane, taking elevation into consideration while building roads. The roads are segmented and follow along the direction of elevation. (3) It can also help to plan regions based on the current condition of development - existing roads and buildings extracted from telemetric images, which is a fatal defect of the related two projects due to the limitations of L-systems.

Many important concepts and techniques in computational geometry and computer graphics are introduced in our system. For road templates generation, we use Voronoi diagram to represent the population-based pattern and L-system for the rule-based growing patterns like raster and radial. An inverse medial axis method is designed for the road expansion. In the end, Delaunay triangulation and an arbitrary polygon clipping method is presented for the terrain simulation.

8.2 Future Directions

In order to build a more detailed and integrated city model, a few challenging problems still require further research. In this section, we will introduce some new topics about city modeling which are also parts of our future work.

8.2.1 Analysis and integration of telemetric information

Development of data acquisition provides us more accurate and detailed information. The general trend in remote sensing is to increase the geometric resolution of the imaging sensors which leads to higher data rate and data

volumes. Resolution of the Radar Star-3, which is developed by the Canadian Space Agency, will reach three meters. The resolutions of American military satellites even exceed two meters. Since the launch of commercial satellite, such as IKONOS and QuickBird, high-resolution satellite images are available periodically. The quality and quantity of satellite images can satisfy the demand of deep use in city modeling.

Comparing to satellite telemetric technology, the telemetric image processing technology is more mature. A series of feature extraction algorithms [21], ranging from image preprocessing, image transform and partition to classification, has been introduced. There are also some methods used to solve practical problems such as road [4] [13] [17] or building [4] [34] extraction.

Figure 8.1: Original satellite telemetric image and the road map extraction result using semi-automatic road extraction method.

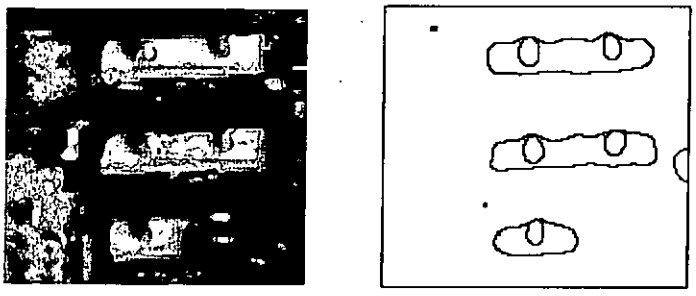


Figure 8.2: Original satellite telemetric image and the building extraction result using multi height bins (MHBs) method.

The satellite images will provide more information for our city modeling project. For example, the road map could be extracted from the input and, if an automatic pattern recognition method is designed, it no longer needs any manual operation such as parameter adjustment. The roads and the buildings extracted from the images, combined with the new roads and buildings developed by the system, will show an integrated model of the future appearance of current city.

8.2.2 Building generation

Besides road network, building is another important component of city. Nobuko Kato [25] use a genetic algorithm to properly lay out building in the urban area whose result is some raw parameters of each building such as height and floor size. Parish [40] introduced a parametric stochastic L-system to model buildings in virtual cities, which can output more detailed outlook parameter of buildings such as texture of facades (shown in Figure 8.3).

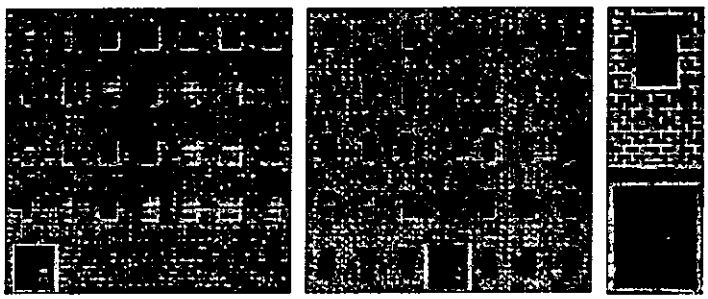


Figure 8.3: Façades of buildings generated by Parish's method

A city in the real world includes thousands of residential buildings, business centers and factories, which are different in size and appearances. Comparing to road map, the buildings of a city show more local styles. The design of building includes more artistry and individuality which are difficult to simulate and reproduce by computer.

8.2.3 Texture modeling

Textures of most of the man-made objects, such as wood or stone wall of buildings, different road planes, are easy to model. Comparatively, solid textures of natural scenes such as mountain and vegetation are much harder to simulate (shown in the right picture of Figure 8.4).

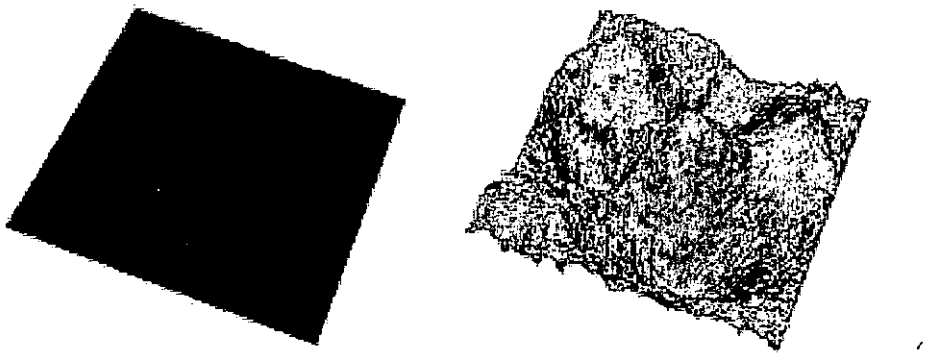


Figure 8.4: Example of different textures.

A city has variant surfaces which look quite different even made of the same material. For example, some rock hills look smooth while some are cliffy. Furthermore, water, air, fire and other natural phenomena are another kind of texture that is important in real world. Simulating these textures will make the city model looks more realistic.

8.2.4 Internet based application of city model

Internet is a media for sharing information. Digital city and digital earth are public resources. People can browse 3D city map through network, looking for shopping centers or find an optimized way from home to office. Another application of city model related to network is reality society or reality games. People can cruise the virtual world or play game with a virtual friend in virtual environment.

Network based application involves technologies such as 3D data compression, transportation and rendering. Some of these technologies have mature algorithms and could be applied on city model without or with minor modification. Some of them are new tasks, for example the 3D browser technology.

References

[1] C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, and S. Angel. *A Pattern Language*. Oxford University Press, New York, 1977.

[2] R.D. Andreev. Algorithm for Clipping Arbitrary Polygons. Computer Graphics Forum 8. pages:183-191, 1989.

[3] E. Baltsavias, S. Mason and D. Stallmann. Use of DTMs/DSMs and Orthoimages to Support Building Extraction, *Swiss Federal Institute of Technology*, 1995.

[4] M. Barsohar and D. B. Cooper. Automatic Finding of Main Roads in Aerial Images by Using Geometric-stochastic Model and Estimation, *IEEE Transactions on PAMI*, Vol. 18, pages 707-721, 1996.

[5] Mark de Berg. *Computational geometry: algorithms and applications (2^{nd} Edition)*. Springer, 2000.

[6] H. Blum. *Models for the Perception of Speech and Visual Form.* Cambridge, MA: MIT press, 1967.

[7] S. Chen. Computer-aided Visualization Simulation in an Urban Context. *MsC Thesis*. College of Environmental Science and Forestry, State University of New York.

[8] W. Chen. 3-D City: Prototyping Techniques for Urban Design Modelling. *http: www.geocomputation.org/1999/002/gc_002.htm.*

[9] J. Cremer, J. Severson, S. Gelo, J. Kearney, M. McDermott. "This old digital city" one year later: experience gained, lessons learned, and future plans. *Proceedings of Seventh International Conference on Virtual Systems and Multimedia*. pages: 49 -56, 2001.

[10] V. Cutini. Urban Space and Pedestrian Movement - A Study on the Configurational Hypothesis. http://www.cybergeo.presse.fr/modelis/cutini/cutini.htm.

[11] W. DoKonal and B. Martens. A Working Session on 3-D City Modeling. *Modeling & City Planning*. pages417-422.

[12] D. Dorling. The Visualization of Spatial Social Structure. *PhD Thesis*. University of Newcastle upon Tyne, U.K.

[13] R. Fiset, F. Cavayas, M.C. Mouchot and B. Solaiman. Map-image Matching Using Multi-layer Perceptron: the Case of the Road Network, *ISPRS Journal of Photogrammetry and Remote Sensing*, Vol. 53, pages 76-84, 1998.

[14] C. Focas (ed.) *The Four World Cities Transport Study.* London Research Center, The Stationery Office, London, 1998.

[15] J. D. Foley, A. Dam, S. K. Feiner and J. F. Hughes. Computer graphics. Principle and Practics. Addison-Wesley. Reading. Ma. 1990.

[16] K. Fuesser. *Stadt, Strasse&Verkehr(City, Roads and Traffic),* Vieweg Verlag, 1997.

[17] D. Geman and B. Jedynak. An Active Testing Model for Tracking Roads in Satellite Images, *IEEE Transactions on PAMI,* Vol. 18, pages 1-14, 1996.

[18] Geometry in Action. *Medial Axes.* http://www.ics.uci.edu/~eppstein/gina/medial.html

[19] A. Gore. The Digital Earth: Understanding Our Planet in the 21st Century, http://digitalearth.gsfc.nasa.gov/VP19980131.html.

[20] G. Greiner, K. Hormann. Efficient Clipping of Arbitrary Polygons. *ACM Transactions on Graphics.* Vol. 17, No. 2, pages 71-83, 1998.

[21] A. Gruen and H. Li. Linear Feature Extraction With 3D LSB-Snake, *Automatic Extraction of Man-made Objects from Aerial and Space Images,* pages 287-297, 1997.

[22] O. Henricsson, A. Streilein and A. Gruen. Automated 3-D Reconstruction of Buildings and Visualization of City Models. *Workshop on 3D-City Models Proceedings*, October 1996.

[23] B. Jiang. SimPed: Simulating Pedestrian Flows in a Virtual Urban Environment. *Journal of Geographic Information and Decision Analysis*, 3(1):21-30, 1999.

[24] O. Jokinen. Building 3-D City Models from Multiple Unregistered Profile Maps. *Proceedings of International Conference on Recent Advances in 3-D Digital Imaging and Modeling*. pages 242-249, 1997.

[25] N. Kato, T. Okuno, A. Okano, H. Kanoh and S. Nishihara. An ALife approach to modeling virtual cities. *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*. pages 1168-1173, 1998.

[26] H. Kawasaki, T. Yatabe, K. Ikeuchi and M. Sakauchi. Automatic Modeling of a 3D City Map from Real-World Video. *Proceedings of the seventh ACM international conference on Multimedia*, pages 11-18, October 1999.

[27] R. Levy. Visualization of Urban Alternatives in Environment and Planning. *Planning and Design*. Vol. 22, pages 343-358, 1995.

[28] Q. Li, W. Shi and B. Yang. 3D City Modeling Based on An Integrated Data Model. *Proceedings of Geoinformatics and Socieinformatics'99*, pages 1-8, 1999.

[29] Youdong Liang and Brian A. Barsky. An Analysis and Algorithm for Polygon Clipping. Communication of the ACM. Vol. 26, pages 868-877, 1983.

[30] R.S. Liggett and W. Jepson. Implementing an Integrated Environment of Urban Simulation: CAD, Visualization and GIS. *Visual Databases in Architechture.* pages 145-160, 1995.

[31] R. S. Liggett and W.H. Jepson. An Integrated Environment for Urban Simulation. Environment and Planning. Vol.22, pages 291-302, 1995.

[32] A. Lindenmayer. Mathematical models for cellular interaction in development, Parts I and II. *Journal of Theoretical Biology 18*, pages280-315, 1968.

[33] P. G. Maillot. A New Fast Method for 2D Polygon Clipping: Analysis and Software Implementation. ACM Transaction on Graphics. Vol. 11, pages 276-290, 1992.

[34] H. G. Mass and G. Vosselman. Two algorithm for Extracting Building Models from Raw Laser Altimetry Data, *ISPRS Journal of Photogrammetry and Remote Sensing*, Vol. 54, pages 153-163, 1999.

[35] M. Melkemi and J.M. Chassery. Edge-region segmentation process based on generalized Voronoi diagram representation. *Proceedings of 11th IAPR International Conference on Pattern Recognition*. pages 323-326, 1992.

[36] M. Molenaar. A Topology for 3D Vector Maps. *ITC Journal*. pages 25-33, 1992.

[37] C. Montani and M. Re. Vector and Raster Hidden Surface Removal Using Parallel Connected Stripes. IEEE Computer Graphics and Applications. Vol. 7, pages 14-23, 1987.

[38] C.W. Niblack, D.W. Capson, P.B.Gibbons, Generating skeletons and centerlines from the medial axis transform. *IEEE Proceedings of Pattern Recognition'90*, Pages 881 -885, June 1990.

[39] J. O'Rourke. *Computation Geometry in C-second edition*. Cambridge University Press, 1998.

[40] Y.I.H. Parish, P. Müeller. Procedural Modeling of Cities. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 301-308, August 2001.

[41] P. Prusinkiewicz, M. James and R. Mech. Synthetic Topiary. *Proceeding of SIGGRAPH'94*. pages 351-358, 1994.

[42] S. Sheppard. *Visualization Simulation: A User's Guide for Architects, Engineers and Planners.* Van Norstrand Reinhold, New York, 1989.

[43] I. E. Sutherland and G. M. Hodgman. Reentrant Polygon Clipping. Communication of the ACM. Vol.17, pages 32-42, 1974.

[44] J. C. Sutton. Role of Geographic Information Systems in Regional Transportation Planning. *Transportation Research Record*, No1518, pages 25-31,1996.

[45] G. Thomas and S. Donikian. Modelling Virtual Cities Dedicated to Behavioural Animation. *EUROGRAPHICS 2000*, 19(3), 2000.

[46] B R. Vatti. A Generic Solution to Polygon Clipping. *Communications of the ACM*. Vol 35, No 1, pages 56-63, 1992.

[47] X. H. Wang and A. Gruen. A 3D City Model Data Structure and Its Implementation in A Relational Database. *Proceedings of Spatial Information Science, Technology and Its application*. Pages 429-436, 1998.

[48] Kevin Weiler and Peter Atherton. Hidden Surface Removal Using Polygon Area Sorting. ACM SIGGRAPH 77, Pages 214-222.

[49] X.H. Yu, John A. Goldak, L.X. Dong. Constructing 3-D discrete medial axis. *Proceedings of the first ACM symposium on Solid modeling foundations and CAD/CAM applications.* Pages 481 – 489, 1991.

[50] J. Zara, P. Chromy, J. Cizek, K. Ghais, M. Holub, S. Mikes and J. Rajnoch. A scalable approach to visualization of large virtual cities. *Proceedings of Fifth International Conference on Information Visualisation.* pages: 639-644, 2001.

[51] Y. Zhu. 3D Building Modeling based on Surface Triangulations. *Journal of Wuhan Technical University of Surveying and Mapping.* Vol 23, No. 2.