

<Proceedings of 1997 China-Japan Symposium on Advanced Energy and Transportation Engineering> p.392-397

## **Automatic Generation of Vector-Map**

*Haitao Wang, Zhenwang Yao, Lingyu Duan*

*Department of Automation*

*University of Science and Technology of China, Hefei, Anhui, 230026, China*

### **ABSTRACT**

*It is very useful to generate the vector map from the bitmap data by extracting the key information in pattern recognition method. Base on our research, a relatively successfully implemented system is the automatic coastline generation system. After trained by several model maps, which are out of a set of maps of the Chinese fishing grounds, the system performs well on the other maps in the same set. The irrelevant features are filtered and only the coastlines are kept. After further process and vectorization, the final vector map comes out. We are considering automatic generation of the maps of city traffics now.*

### **1. INTRODUCTION**

A map, when saved in database in the format of bitmap, is not convenient for further use and process. Vector format is far more efficient in time and memory space. If we can generate the vector map from the bitmap data by extracting the key information and erasing the other feature in pattern recognition method, the generation of the vector map will be easier and faster. Because the bitmap data can be gotten from the printed map or photograph by scanner, it is very useful where the electronic geographical data is not available.

Because of the variety of the map, the recognition method varies subject to the type of the map. Based on our research, a relatively successfully implemented system is the automatic coastline generation system. It aims at a set of maps of Chinese fishing grounds. In this case, the main recognition method is bases on the color difference between land and sea and the continuity of the coastline. The system contains two main parts, coastline extract and vectorization. In the following we will discuss it in detail.

### **2. COASTLINE EXTRACT**

Coastline extract is the recognition part of the system. After extract we get the black-white bitmap which represent the coastline in the original map.

#### **2.1 Scan**

Use a scanner to convert the printed map into a bitmap file that can be processed by computer. Use fixed dpi and brightness parameters when scanning, and use 24-bit

true color. This is to ensure the consistency between the different bitmaps that are scanned in different times.

## 2.2 Sample Define And Color Classify

Because of the error in the origin map and the noise in the scanning procedure, there are some differences between the points in the same color area. In order to recognize the coastline correctly, we must classify the colors of the points in the bitmap into several color classes.

At first user select some sample areas in the bitmap using a mouse, and designate the properties of the area such as land, sea or border. Each area represents a color class. Then the program travels the whole bitmap, and classify each point into a color class. As to a set of maps, because their tones are almost same, user only needs to define samples in a few maps. These samples can represent all color classes in the whole set of maps.

Classify algorithm is described below:

- A. Define color classes  $C_i$ , and for each color class define a base color  $c_i$  and two thresholds  $K_{i1}$  and  $K_{i2}$ .
- B. During the first travel, if the difference between the color of a point and  $c_i$  is less than  $K_{i1}$ , then classify this point into  $C_i$ .
- C. Then travel the bitmap many times. In odd times travel from top to bottom, from left to right, and in even times from bottom to top, from right to left. When reach a point  $P$ , if one of its neighbor point  $P_j$  is in  $C_i$ , and the color difference between  $P$  and  $P_j$  is less than  $K_{i2}$ , then classify  $P$  into  $C_i$ . This is called color trace or color expanding. This step finishes when there is no new point be classified in a travel. The main purpose of color trace is to keep the continuity of the line shape feature in the bitmap.

## 2.3 Coastline Extract

We can extract the points that consist the coastlines after further process on the color-classified bitmap. The key point in this step is to increase the precision and the ability of resisting noises.

After color classify, a point in the bitmap can has four statuses: land, sea, border or unknown. Define a coastline point as

- A. land point which has a neighbor sea point  
or
- B. point on the outer edge of the border points that has a land point and a sea point on the different sides.

In order to keep the continuity of the coastline and increase the noise-resist ability, we travel the color-classified bitmap twice. In first time use a looser condition and find a super set of coastline points, then select the outer side of this set as the coastline points. We got fine effects using this method.

## 3. VECTORIZATION

Vectorization is an independent part that can be used not only in this system. Its main function is to convert a black-white bitmap which represents the line-shape figures into a vector format graph. It contains the following steps:

### 3.1 Smoothing

Because of the noises and errors, there are some indents and bulges in the line-shape features. Removing these indents and bulges before further process can improve the precision of vectorization.

We use pattern match method to fill the indents and erase the bulges. Using 1 to represent the object points, and 0 to represent the background points, we get the following indent patterns.

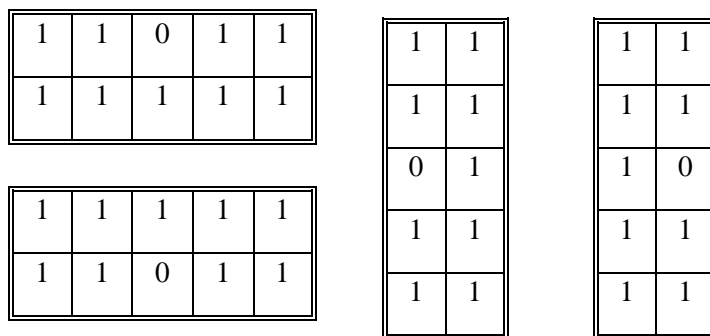


Fig. 1 Patterns of the indent.

As to the bulges, the following patterns are used.

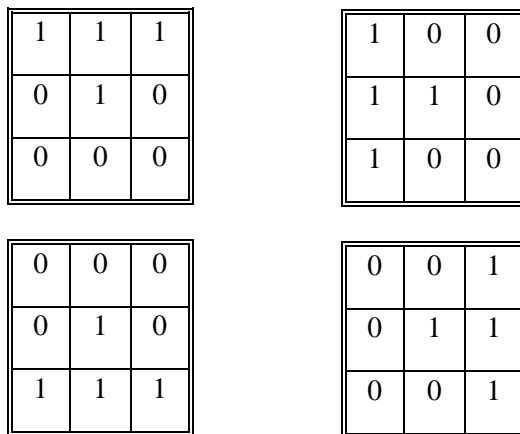


Fig. 2 Patterns of the bulge.

Change 0 in indent patterns to 1 then all indents will be filled, and change the 1 on the center of bulge patterns to erase all bulges.

### 3.2 Thinning

Thinning is the process to erase the points that do not affect the connection of the line shaped figures. The purpose of it is to get the one-point-wide center line to

make the later vectorization applicable. Now we use the classical thinning algorithm.

Consider an object point (point on the final center line)  $P$ . Mark the number of the disconnecting elements within the 8 neighbor points of  $P$  (points that connect to  $P$ ) with  $N_C$ , and mark the eight neighbor points of  $P$  with  $P_0$  to  $P_7$ , then we can delete those points satisfy the following conditions:

- a.  $P_0 + P_2 + P_4 + P_6 = 3$
- b.  $N_C = 1$
- c. There is at least one object point in  $P_i$  ( $0 \leq i < 7$ ).
- d.  $P_2 = 1$  or ( $P_2 = 0$  and  $N_{C2} = 1$ )
- e.  $P_4 = 1$  or ( $P_4 = 0$  and  $N_{C4} = 1$ )

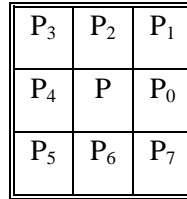


Fig. 3 Definition of  $P_1$  to  $P_7$

Travel the bitmap until no new point can be deleted, then the thinning process ends.

### 3.3 Joining Broken Lines And Deleting Spurs

These processes are to fix the errors in the original map or those entered during the former procedures.

#### A. Joining broken lines

Define the point whose number of neighbor object points is not two as *node*. Compute the distance between each two nodes, and when it is less than a designated value, join the two nodes with a line of object points.

#### B. Deleting spurs

After joining broken lines, compute the distance between each two nodes again. When it is less than another threshold, delete the line of object points between the two nodes. That is, join the two nodes with a line of not-object points.

### 3.4 Vectorization

After the former procedures, we got a bitmap of single-point-wide lines. We can begin to vectorize it now. Vectorization can greatly reduce the memory space required to save the map information, and make the further operation on the map easier.

The vectorization algorithm is described below:

Also define the point whose number of neighbor object points is not two as *node*. Start tracing the object points from nodes for vectorization. Search object points from a node until reach to another node, and mark these points found so that we can skip it next time. These points form a curve. Then use a serial of lines to approach the curve. This is vectorization. Vectorization may bring errors, and the

error is determined by approaching algorithm.

A recursive approaching algorithm is used. Find the point P that has the maximum distance  $D_P$  to the line between the two nodes  $P_1$  and  $P_2$ . If  $D_P$  is less than the predefined maximum error  $E_{max}$ , the approaching is completed. Otherwise approach the curve between  $P_1$  and P and another between P and  $P_2$  respectively. The algorithm must be convergent because when  $P_1$  is adjacent to  $P_2$ ,  $D_P$  must be 0.

Normally the data after vectorization is far less than before, because only the nodes need be saved.

#### 4. AUTOMATIC SITING AND DATABASE ADDING

A map generated by the former procedures must be sited before it can be used. Also some information besides line features need be added to make the map more useful.

##### 4.1 Automatic Siting

Suppose a point with coordinate (X, Y) has longitude and latitude (lon, lat). Select a base point ( $X_0, Y_0$ ), and suppose its longitude and latitude is ( $lon_0, lat_0$ ). Make the scale in the two axis directions with ScaleX and ScaleY, and rotation with rot, we can get

$$\begin{aligned} X = & \text{ScaleX} \cdot \cos(lat_0) \cdot \cos(rot) \cdot lon \\ & + \text{ScaleY} \cdot \sin(rot) \cdot lat \\ & - \text{ScaleX} \cdot \cos(lat_0) \cdot \cos(rot) \cdot lon_0 \\ & - \text{ScaleY} \cdot \sin(rot) \cdot lat_0 \\ & + X_0 \end{aligned}$$

$$\begin{aligned} Y = & \text{ScaleX} \cdot \cos(lat_0) \cdot \sin(rot) \cdot lon \\ & - \text{ScaleY} \cdot \cos(rot) \cdot lat \\ & - \text{ScaleX} \cdot \cos(lat_0) \cdot \sin(rot) \cdot lon_0 \\ & + \text{ScaleY} \cdot \cos(rot) \cdot lat_0 \\ & + Y_0 \end{aligned}$$

Mark

$$\begin{aligned} A_1 = & \text{ScaleX} \cdot \cos(lat_0) \cdot \cos(rot) \\ B_1 = & \text{ScaleY} \cdot \sin(rot) \\ C_1 = & X_0 - \text{ScaleX} \cdot \cos(lat_0) \cdot \cos(rot) \cdot lon_0 \\ & - \text{ScaleY} \cdot \sin(rot) \cdot lat_0 \\ A_2 = & \text{ScaleX} \cdot \cos(lat_0) \cdot \sin(rot) \\ B_2 = & \text{ScaleY} \cdot \cos(rot) \\ C_2 = & Y_0 - \text{ScaleX} \cdot \cos(lat_0) \cdot \sin(rot) \cdot lon_0 \\ & + \text{ScaleY} \cdot \cos(rot) \cdot lat_0 \end{aligned}$$

then

$$X = A_1 \cdot \text{lon} + B_1 \cdot \text{lat} + C_1$$

$$Y = A_2 \cdot \text{lon} + B_2 \cdot \text{lat} + C_2$$

$A_1, B_1, C_1, A_2, B_2, C_2$  is not correlative to  $X, Y, \text{lon}$  and  $\text{lat}$ . Thus by using data analysis method, we can compute  $A_1$  to  $C_2$  by several groups of  $X, Y, \text{lon}$  and  $\text{lat}$ .

#### **4.2 Database Adding**

This is a human-computer interaction procedure. User input some information such as a mark with name and icon, or an area with color and name, by mouse and keyboard, then computers draw it on the map and add it to database.

After these steps, a vector map becomes a complete geographical information system.

#### **REFERENCES**

- [1]Wenge Shi, Standard Forms of Images used in Micro Computers, Ocean Press, Peking, 1992
- [2]J.Ekblad & N.E.Eriksson, "Hailing GPS: A Swedish taxi security system", GPS World, 5, pp32-36, 1996