# SOME RULES FOR IDENTIFYING SPATIAL RELATIONSHIPS OF CONTOURS BASED ON CONTOUR TREE

Chaofei QIAO[1,2], Jun CHEN[2], Zhilin LI[3], Jianjun LIU[2]

[1] China University of Mining and Technology ( Beijing Campus)
qiaocf@sina.com.cn
[2] National Geomatics Center of China
[3] Department of Land Surveying and Geo-Informatics, Hong Kong Polytechnic University

**Commission II, WG II/6**

**KEY WORDS:** Contour Tree, Graph Theory, Spatial Relationships of Contours, Rules

**ABSTRACT:**

The spatial relationships of contours play important roles in many areas such as extracting topographic features from contour maps, automatically identifying elevations of scanned contours, *et al*.. In those areas, one key issue is making computers identify the relationships of contours. The spatial relationships of contours are usually represented by contour tree. Contour tree is essentially a tree structure which is a concept in *graph theory*. In this paper, some rules for identifing spatial relationships of contours are introduced on the basis of the knowledge of contour tree. One experiment of identifying hills from contour map is illuatrated to verify the correctness of the rules.

## 1. INTRODUCTION

Contour lines have been used for the representation of continuous variations of 3D phenomena on topographic maps for a long history. A contour is a line on a map which represents an imaginary line on the ground, all points of which are of equal elevation as referred to a specified common datum plane (Wu, 1993, pp.22). Groups of contour lines have some characters which are non-intersection, parallelism and geometric similarity. These characters determine the spatial relationships of contour lines.

The spatial relationships of contours play important roles in many areas such as extracting topographic features from contour maps, automatically identifying elevations of scanned contours, *et al*.. Identifying topographic features from contour maps is of essential importance for some applications such as automatic terrain generalization (Wu, 1999), autonomous navigation in natural terrain (Kweon and Kanade, 1994), and contour-based terrain reasoning (Cronin, 1995).

People can easily recognize topographic features through interpreting contour lines drawn on paper map. In this process, knowledge of spatial relationships, geometry and elevations of contour lines play important roles. In order to make computers automatically interpret the contour maps which are stored in computers in digital format, those knowledge must be identified by computers. In this paper, we intend to use some rules defined on the basis of contour tree which well represent spatial relationships of contour lines.

In section 2, the spatial relationships of contour lines are classified firstly. Then the theory of contour tree, which is the main way of representing the spatial relationships of contours, are introduced. On the basis of the knowledge of contour tree, some rules of representing the spatial relationships of contours are given out. One experiments of identifying topographic features using these rules is presented in section 3. The possible feature work is discussed in section 4.

## 2. SOME RULES BASED ON CONTOUR TREE

### 2.1 Classification of Spatial Relationships of Contour Lines

In general, people think that there are two kinds of spatial relationships of contours: *in* and *paratactic* (Guo, 1995). Given two contours $c_1$ and $c_2$ in a map, if $c_1$ is inside of the region formed by the line of $c_2$, we call $c_1$ and $c_2$ hold *in* relationship. If the elevation of $c_1$ is equal to the elevation of $c_2$, we call $c_1$ and $c_2$ hold *paratactic relationship*. In fact, besides in and paratactic relationships, the more essential relationship contours hold is *adjacent* relationship. There isn't a unique definition of spatial ajacent, because the defintion is based on applications (Anders, 1997).

### 2.2 Contour Tree

At present, the main means to represent the spatial relationships of contours is contour tree. Contour tree was invented by (Boyell and Ruston, 1963), which is a kind of tree structure , an important structure in graph theory. A graph is a mathematical structure used to model systems where the relations between the objects in the systems play a dominant role. Terms used in the paper follow terminology of graph theory when possible. For detailed information concerning graph theory, readers may refer to (Bollobas, 1998; Wang, 1997).

A contour tree $T$ consists of a finite set $V$ of objects called *vertices*, and a finite set $E$ of objects called *edges*. Usually the vertices represent contour lines and the edges represent the adjacent relationship of the two vertices the edge links. A pair of vertices that determine an edge are *adjacent* vertices. The set of vertices adjacent to a vertex $a \in T$ is the *neighbourhood* of $a$ (Bollobas, 1998, pp.3).

A contour map with sixteen contour lines and its corresponding contour tree are shown in Fig1. The sixteen vertices of the contour tree correspond to the sixteen contour lines on the map shown in Fig1(a). The contour 16 is the *root* of the contour tree.

It has two *offsprings*: contour 12 and 15, which are *siblings* each other (Kolman, *et al.*, 2000, pp.246).

The *degree* of a vertex of a contour tree is the number of edges having that vertex as an end point (Kolman, *et al.*, 2000, pp.281). We denote *degree* ($v_i$) as the degree of vertex $v_i$. For every contour $c_i$ in a contour tree, we can obtain the total number of the neighbors of $c_i$ by means of the degree of $c_i$. For example, the degree of contour 12 in Fig 1 is 4, which means contour 12 has four neighbors which are contour 16, 11, 2 and 3.

The *level* of a vertex of a contour tree $T$ is the *length* of the *shortest path* (Wang, 1997, pp.206-219) from the root to this vertex, i.e., the least number of the edges from the root to this vertex. We denote *level* ($v_i$) as the level of vertex $v_i$. To compute the level of the vertices in a contour tree, we can use the algorithms used in the shortest path problem in graph theory. There are two methods often used, i.e., Dijkstra algorithm and Floyd algorithm (Wang, 1997, pp.206-219; Yin, *et al*., 1999, pp.283-290). Dijkstra algorithm is used to compute the shortest path from one vertex to another one in a graph. By contrast, Floyd algorithm is used to compute *all* the shortest path between every pair of vertices in a graph. Obviously, Floyd algorithm is more efficent in computing the level of every vertices in a contour tree.

Acctually, the contour tree such as the one shown in Fig 1(b) is the graphic representation of the spatial relationships of contours. However, the contour tree should be stored in computers in some format so that computers can recognize it. The contour tree can be represented in computer as the *adjacency matrix* (Bollobas, 1998, pp.54; Wang, 1997, pp.105). The adjacency matrix $A(T) = (a_{ij})$ of a contour tree $T$ is a $n \times n$ matrix given by

$$\alpha_{ij} = \begin{cases} 1 & \text{if } v_i v_j \in E(T) \\ 0 & \text{otherwise.} \end{cases}$$

Where $n$ is the number of contours, and $v_i$ and $v_j$ are two contours.

In Fig1, the contour 1 is adjacent to contour 6, so $a_{16} = a_{61} = 1$. The contour 1 isn't adjacent to contour 5, so $a_{15} = a_{51} = 0$.

Because the elements of $A(T)$ indicate the situation of adjacent relationship of contours, the value of the degree of every vertex is the sum of the value of every element of the line or column in $A(T)$. The line or column corresponds to this vertex (Wang, 1997, pp.106).

Now, we can give some rules on the basis of the above terminologies and definitions.

### 2.3 Some Rules for Identifying Spatial Relationships of Contours

Let $C = \{c_1, c_2, ..., c_n\}$ be a set of closed contours on a map. $a$, $b \in C$.

- Rule 1: Enclosing rule

If

is_neighbor ($a$, $b$) ,
*level* ($a$) = *level* ($b$ )-1,

Then

$a$ contains $b$ , or $b$ is inside of $a$.

In Fig 1, the contour 16 contains other fifteen contours.

- Rule 2: Paratactic rule

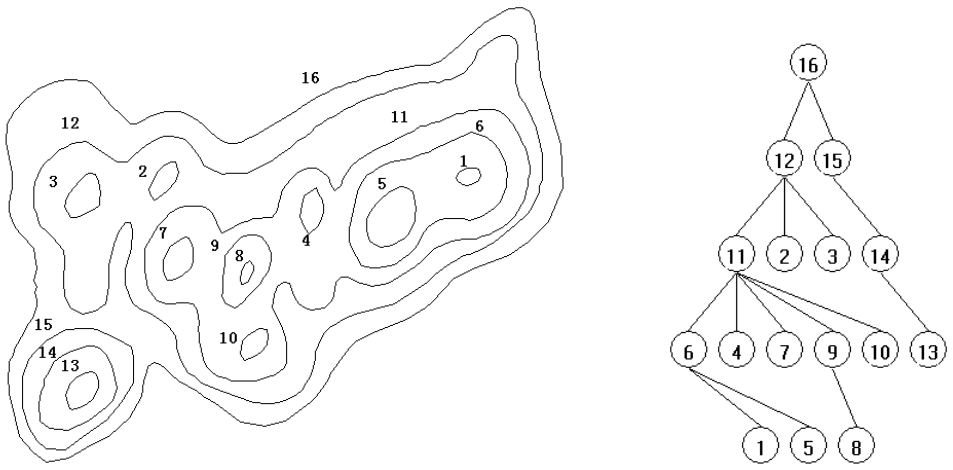
If

*level* ($a$) = *level* ($b$ ),

then

$a$ and $b$ hold paratactic relationship.

In Fig 1, *level* (1) = *level* (5) = *level* (8), so contour 1, 5 and 8 hold paratactic relationship. However, there is difference between the relationship of contour 1 and 5 and the one of contour 1 and 8, i.e., contour 1 and 5 are contained by the same contour 6, while contour 1 and 8 don't contained by the same contour. To distinguish these two kinds of relationship, we give out the following two rules.




(a) (b)

Figure 1. A contour map and its corresponding contour tree

- Rule 3: First paratactic rule

If

*level* (*a*) = *level* (*b* ),
the length of the shortest path from a to b is 2,

then

*a* and *b* hold first paratactic relationship.

In Fig 1, the contour 1 and contour 5 hold first paratactic relationship.

- Rule 4: Second paratactic rule

If

*level* (*a*) = *level* (*b* ),
the length of the shortest path from a to b is larger than 2,

then

*a* and *b* hold second paratactic relationship.

In Fig 1, the contour 1 and contour 8 hold second paratactic relationship.

- Rule 5: Local extremal elevation rule

If

contour *a* isn't the root of a contour tree,
*degree* (*a*) = 1,

then

*a* is a peak or pit.

In Fig 1, the contour 1, 2, 3, 4, 5, 7, 8, 10 and 13 are peaks.

These rules can be easily translated to a computer language and can be used in the process concerning spatial relationships of contours.

## 3. IDENTIFYING HILLS FROM CONTOUR MAPS: PRELIMINARY EXPERIMENT

In this section, we present an experiment to illustrate how to use the rules given in last section in the process of identifying hills from contour maps by computers.

One of the prerequisites for identifing topographic features from contour maps is the formal definition of various topogrphic features. Currently, there hasn't a good classification of landforms（Mark and Smith, 2001）. Two typical topographic features are hills and dales. Here we denote a *hill* is a group of contours which nest each other, and the elevation of every contour in the group is larger than the elevation of the contour containing it. Similarly, a *dale* is a group of contours which nest each other, and the elevation of every contour in the group is less than the elevation of the contour containing it. A *peak* is a contour with the largest elevation in the contours constituting one hill. A *pit* is a contour with the least elevation in the contours constituting one dale. Particularly, we call those hills or dales among which there is only one contour in every nesting level as *simple hills* or *dales*. Correspondingly, the peaks of simple hills are called *simple peaks*, and the pits of simple dales are called *simple pits*.

Were the elevation attribute not considered or unknown, the contours constituting hills and the ones constituting dales have similar graphic characters. Therefore, in this experiment, we only identify hills from contour maps.

In this experiment, a map with 24 contours is used, which is shown in Figure 2. The data are at the scale of 1: 50 000 and in vector format. The database in which the contours are stored is named as *TempData*. There are four steps in the process of identifying hills from contour maps in this experiment, i.e.,

(1) Computing all the adjacent relationship of contours
(2) Computing the adjacent matrix of contour tree and degree and level of vertices
(3) Identifying simple hills
(4) Identifying other hills

These four steps are introduced in the following.

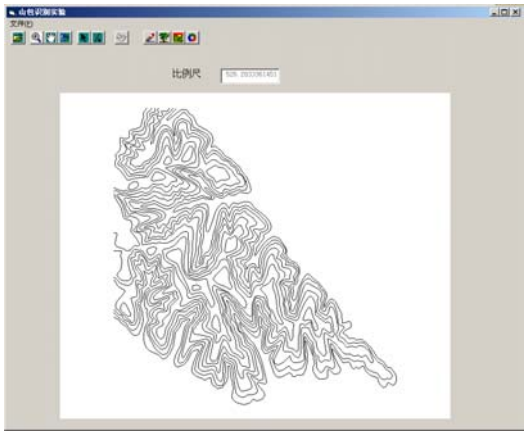


Figure 2. The original state of the contour map

### 3.1 Computing All the Adjacent Relationship of Contours

In the experiment, we use Voronoi diagram to obtain all the adjacent relationship of contours. Voronoi diagram is a geometrical constructure. It has been used in many scientific fields because of its interesting and surprising mathematical properties. In this step, we generated the Voronoi diagram of the contour lines shown in Fig 2. If there is a Voronoi edge between two contours, then the two contour hold *adjacent relationship*. Otherwise, they don't. All of the adjacent relatioship is stored in an adjacency matrix. We denote this matrix as *Adj*. Correspondingly, the adjacency matrix of the contour tree is denoted as *TreeAdj*. The main difference between the matrix *Adj* and the matrix *TreeAdj* is that the latter only stores part of the adjacent relationship other than all the adjacent relationship.

### 3.2 Computing the Adjacent Matrix of Contour Tree and Degree and Level of Verteices

In the beginning of this step, the value of every element in the adjacency matrix *TreeAdj* of the contour tree is zero. Then we found the neighbourhood (see section 2.2) of the root of the contour tree by inspecting the value of the elements of the adjacency matrix *Adj* . If *Adj* ($c_i$, $r$) = 1, where $c_i$ is a certain contour and $r$ is the root, then $c_i$ is an element of the neighbourhood. let *TreeAdj* ($c_i$, $r$) = 1. Having found the neighbourhood of the root, we continually found in the left contours the neighbourhood of the contours in the neighbourhood of the root, and aasign 1 to the value of the corresponding elements in *TreeAdj*. Repeat this process until all the contours were found.

The value of the degree of every vertex is the sum of the value of every element of the line or column in the matrix *TreeAdj*. The line or column corresponds to that vertex. The value of the

level of every vertex was got by using the Floyd algorithm in this experiment.

### 3.2 Identifying Simple Hills

As mentioned in the beginning of section 3, simple hills are hills among which there is only one contour in every nesting level. In this step, we first identified all the peaks of hills in term of rule 5 given out in section 2. Second we identified the peaks of simple hills in term of rule 3 and 4. Third we identified other contours constituting the simple hills by means of adjacent relationship of contours. Finally the contours constituting simple hills were deleted from the database *TempData*.

### 3.3 Identifying Other Hills

This process consists of three sub-process. Firstly, find in the database *TempData* the contours with the largest level in contour tree. Take one contour from these contours, denote it as *Cmax*. Secondly, find those contours that hold first paratactic relationship with *Cmax*, then delete those contours from database *TempData*. Repeat the former two steps until the root was identified. This process can be seen as a process of flood submerging the hills gradually.

Figure 3 and Figure 4 are two hardcopies of the experiment.

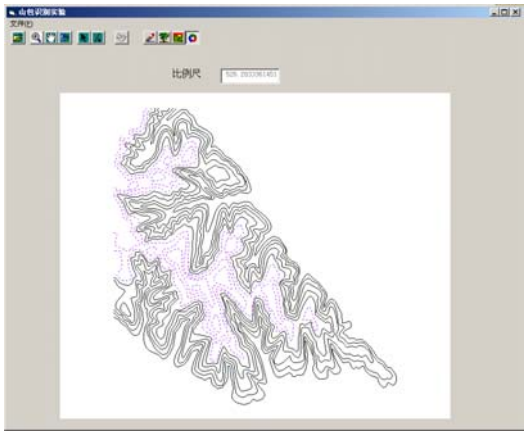


Figure 3. Having identified some hills

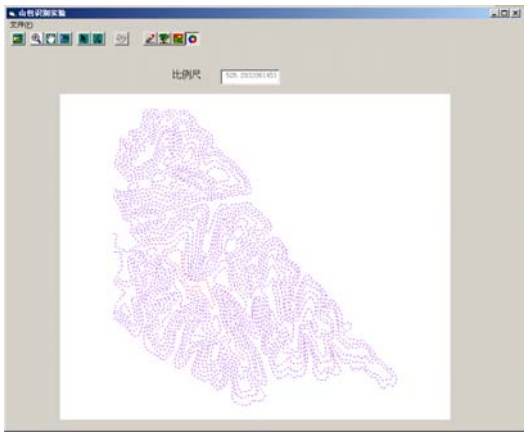


Figure 4. Identified all the hills

## 4. DISCUSSION AND FUTURE WORK

In this paper, some rules for identifying spatial relationships of contours are presented. These rules are defined on the basis of the relavant knowledge about contour tree. We present an experiment of automatically identifying hills from contour maps by computers. The experiment indicate that the algorithms developed on the basis of these rules work well.

Besides the area of extracting some topographic features from contour map, these rules can be used in other areas where spatial relationships of contours play roles. For instance, they can be used in the field of assignment of elevation to scanned contours (Wu, 1993), and the field of checking the quality of digitized contours (Liu, *et al*., 2001), etc.

The rules presented in the paper are the most basic ones defined on the basis of contour tree. Nevertheless, the reality of the terrain differ in thousands of ways. If one want to indentify more complex topographic features, additional information such as morphology must be concerned.

## ACKNOWLEDGEMENTS

The work described in this paper was supported by the National Natural Science Foundation of China under grant No.40025101. The authors would thank to Dr. Jiang jie, Mr. Zhao Renliang and Miss. Wang Yanhui for their comments on this work.

## REFERENCE

Anders, K., 1997. Automated interpretation of digital landscape models. In: *Fritsch and Hobbie (eds.). Proc. of Photogrammetrische Woche* 1997, pp.13-24

Bollobas, B., 1998. *Modern graph theory*. Springer-Verlag, 408 pp.

Boyell, R. and Ruston, H., 1963. Hybrid techniques for real-time radar simulation. In: *Proc. of the Fall Joint Computer Conference*, Las Vegas

Cronin, T., 1995. Automated Reasoning with Contour Maps. *Computers & Geosciences*, 21(5): 609-618

Guo, Q., 1995. The intelligent method of building hierarchical structure of contours. *Journal of Wuhan Technical University of Surveying and Mapping*, Vol.20, Supplement, pp.69-75 (in Chinese)

Kolman, B., Busby, R. and Ross, S., 2000. *Discrete Mathematical Structures*. Prentice Hall, Inc., 505 pp.

Kweon, I. and Kanade, T., 1994. Identifying Topographic Terrain Features from Elevation Maps. *CVGIP : Image Understanding*, 59(2):171-182

Liu, J., Wang, D. and Shang, Y., 2001. One method of automatically checking the quality of digitized contour. *Science of Surveying and Mapping*, 26(4): 36-38 (in Chinese)

Mark, D. and Smith, B., 2001. A Science of Topography: Bridging the Qualitative-Quantitative Divide. In: Michael P. Bishop and Jack Shroder (eds.), *Geographic Information Science and Mountain Geomorphology*, Chichester: Springer-Praxis

Wang, C., 1997. *Graph theory*. The Publishing House of Beijing Polytechnic University, Beijing, 399 pp. (in Chinese)

Wu, C., 1993. *Automated Identification of Scanned Contour Information for Digital Elevation Models*. Doctoral Thesis. The Graduate School, University of Maine, USA.

Wu, H., 1999. *Principle of computer – assisted map generalization*. Wuhan Technical University of Surveying and Mapping, Wuhan, China（in Chinese）

Yin, R., Tao, Y., Xie, R. And Sheng, X., 1999. *Data structure---illustrated using O-O method and C++*. The Publishing Housing of Tsinghua University, Beijing, 402 pp. (in Chinese)