J. Infrastructure Plan. and Man., JSCE, No. 688/IV-53, 135-144, 2001 October

EXTRACTING CONTOUR LINES UTILIZING FEATURES OF A RASTER IMAGE TOPOGRAPHIC MAP

Weiqing LI[1], Eihachiro NAKAMAE[2] and Takaharu MIYOSHI[3]

[1] Ph.D.,Associate Prof.,State Key Lab.of CAD & CG, Zhejiang University(Hangzhou, 310027 P.R.China)
[2]Member of JSCE, Dr. Eng., Sanei Co. Ltd.(3-13-26 Kagamiyama, Higashi-hiroshima, 739-0046 Japan)
[3]Member of JSCE, Assist. Prof.,Hiroshima Inst. of Tech.(2-1-1 Miyake, Saeki-ku, Hiroshima, 731-5193 Japan)

It is a difficult task to automatically extract contour lines from a raster image topographic map and give each contour line its inherent elevation. In this paper, the authors propose an automatic contour line extraction algorithm based on the continuity among the contour lines. All of the points on a contour line indicate the same elevation, the difference of elevations between every neighboring contour line is constant, and most wide contour lines have numeral arrays pointing out their inherent elevations. By employing the information of these features, the shapes and elevations of contour lines can be extracted and recognized. Usefulness of the proposed algorithm is demonstrated with several practical maps.

Key Words: *topographic map, contour line, raster-to-vector conversion*

1. INTRODUCTION

Nowadays, the technology of the raster-vector conversion of a binary terrain map is mature and widely used in civil engineering. Automatic recognition of contour lines getting the following information, however, has not been successful, as far as authors know:

(a) Automatic recognition of contour lines and discrimination between index contours and intermediate contours by using the information of their different width,

(b) Automatic recognition of the information of index contours by recognizing their numeral arrays,

(c) Automatic verification of intermediate contours by employing the information of the index contours,

(d) Automatic restoration of contour lines separated due to their own numeral arrays, some symbols, lines etc.

Extracting contour lines from a raster image topographic map and giving each one its inherent elevation are one of many important tasks that require significant human and computer resources. During the past decade many systems[1),2),3),4),5),6),7),8)] have been described and published for various purposes such as constructing roads and city planning, etc. Undoubtedly, the most important application field of map data processing is in GIS (Geographic Information System)[9)].

The main goal in this field is to recognize every type of significant features in a topographic map, understand their meaning, and represent them by suitable methods. There are many recognition algorithms for interpreting topographic maps. H.Yamada et al. proposed the MAP concept[10), 11)] to extract lines, sets of points, buildings, hatched areas, roads and character regions of topographic maps. The recognition method in MARIS system[2)] can extract building lines, contour lines, and the lines representing railways, roads and water areas, and store them within a layered data structure. Several different recognition methods are presented in references 3) and 4). Concerning recognition of contour lines employing color information of 1/25,000 and 1/50,000 scale maps, the algorithms based Voronoi-line[12)], and normalizing color images[13)] were developed.

All these methods can be classified into two categories: raster image based processes, references 3), 4), 6), 7), 10), 11), 12) and 13) and vector based ones, references 1), 2), 5) and 8).

In the raster image based processes, the recognition techniques are usually directly applied to raster images. Theoretically, the raster images can present the original data information of every feature recognition procedure. These kinds of methods, however, can deal with practically only simple figures such as straight lines, numerical characters, and other simple symbols, and the processing time is significant. Moreover, hardware requirements(e.g.

memory size and computing time) are too limited to practically process a large topographic map.

In vector-based processes, the raster images of topographic maps are usually converted to vector-based image initially. The processes for recognizing and understanding them are then based on the vectorized results. In this case, the processing time can be drastically reduced, the memory requirements are substantially reduced, and appropriate techniques can easily handle the different characteristics of each kind of features in topographic maps.

One of weak points in vector conversion may be due to the distortions around intersections resulting from the raster-to-vector conversion. Further processing is required to repair them.

The contour lines expressing shapes and elevations of the terrain in topographic maps are an essential element. Recognizing these contour lines correctly makes it possible to extract key data points for the further processing options such as 3D terrain reconstruction, CAD for roads and bridges, etc. Regrettably, until now, the limited information available on the subject has been focused on automatic contour line extraction from scanner-input images, especially in a vector-based process.

An extraction method based on vector data is found in reference 2). In this paper, according to the periodicity of contour lines, all long lines are labeled, and a neighboring relation table is created to extract the contour lines. However because of no extraction of elevation information, the elevations of contour lines cannot be automatically recognized, and the validity of relationship between contour lines can not be automatically examined.

In this paper the authors propose a contour line extraction algorithm that is able to utilize the features of raster images and the continuity of vectorized contour lines. The following section describes the basic idea how to extract contour lines and recognize their elevations. Section 3 discusses classification of line segment. Section 4 explains how to extract contour lines. In section 5 some examples demonstrate the efficacy of the proposed algorithm. Section 6 gives some conclusions and discusses future work to solve the remaining problems.

2. BASIC IDEA

In order to extract contour lines from a raster image topographic map and assign each one its inherent elevation, the authors employ the following features in the map.

(1) Exploring continuity of contour lines

For extracting contour lines, following two features of continuity are positively employed:

(a) Dot continuity: Each dot compositing one contour line has the same elevation. It ensures that every dot is located on its inherent horizontal plane when 3D terrain reconstruction is realized.

(b) Line continuity: The difference of elevations between any of two neighboring contour lines in a topographic map keeps a constant. If the elevations of any of two contour lines are known, the elevations of every contour line can be calculated theoretically. Contours, however, are separated by the symbols of cliffs or touch each other because of a very steep slope. In these cases the current system has no means of settling.

(2) Exploring geometric features

The contour lines have the following geometric features:

(a) Usually contour lines consist of long vector chains compared with the others, e.g. roads, rivers, buildings, some symbols such as numerals, Chinese characters, etc.

(b) According to width, contour lines can be classified into two types, wide lines and thin lines.

(c) The wide contour lines usually hold numeral arrays indicating their inherent elevation.

The proposed algorithm, the continuity-based contour line extraction, is constructed by the following several steps: 1)Raster to vector conversion; 2)Classification of polygonal lines; 3)Recognition of numerals indicating the inherent elevation of contour lines; 4)Extraction of index contours (e.g. every 10 or 5 meters); 5)extraction of the other contour lines.

Through these steps, the contour lines can be recognized automatically and correctly.

3. CLASSIFICATION OF LINE SEGMENTS

Still now in many cases an original topographic map is given as a raster image. The algorithm positively employs its drawing rule: each object is drawn with the following either one, e.g., 1)wide lines, 2)thin lines, 3)doted lines, 4)broken lines etc., and some symbols consisting of a set of short lines. In order to extract contour lines, the algorithm uses some pre-processes.

First, the proposed system vectorizes the raster image, and classifies them into four types: wide long lines, thin long lines, wide short lines, and thin short lines.

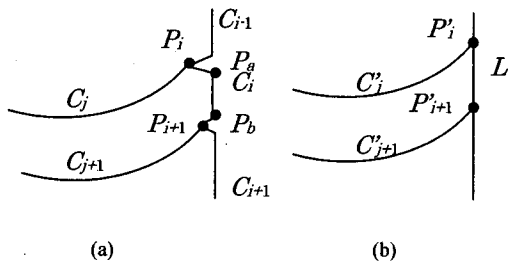


Fig.1 Correcting intersections : **(a)** vector chains, **(b)** after correcting intersections.

(1) Raster-to-vector conversion

There are many references in this document concerning raster-to-vector conversion processing[14),15),16)]. The conversion process in this paper consists of three steps:

Step 1. Thinning of raster images of a topographic map: Each foreground pixel of the map has one of three positional behaviors: edge pixel, internal pixel, or skeleton pixel. On the thinning process when one edge pixel is erased (it becomes a background pixel), its neighboring internal pixels (from one pixel to three pixels) become edge ones. Note that in order to keep topological structures, the skeleton pixels are never removed[17)].

For every foreground pixel, a set of masks(3×3) is used for testing the positional behaviors of the current pixel. If it is an edge pixel, it is changed to a background one. By iterating this process several times, the original raster image is transformed to a set of one-pixel-width images.

Step 2. Creating chain codes: A one-pixel-width image is edited as a set of chain codes consisting of a sequence of which pixels are numbered from zero to seven. Each pixel has one of eight directions related to its next neighboring pixel. The chain codes decrease the memory required to express the information of the entire image. The end pixel of a chain represents either a free end point of the chain or an intersection with some chains.

Step 3. Vectorizing chain codes: By employing the information of the chain codes, a set of short straight line segments with the common end points one by one, (it is often called a vector chain), can be made. In order to maintain precision of the vector chains, the length of every line segment is set as its deviation is held within one-pixel width. Then the polygonal lines are organized by these straight line segments.

Because of the distortions around intersections resulted from the raster-to-vector conversion, the end points of vector chains in most cases are not set at the theoretical position, and the lines result in zigzag (see **Fig.1(a)**).

This distortion occurring at the boundary lines of the current map gives serious problems when it is connected to the neighboring one. By processing the following steps the adjacent maps can be continuously connected.

Step 1. For one vector chain(in this example C_i),create a straight line (P_aP_b) through two points which splits the chain(C_i) into three parts(P_iP_a), (P_aP_b) and (P_bP_{i+1}).Calculate the distances from each vertex P_i/P_{i+1} of the chain to the vertex of the P_a/P_b of the straight line segment (P_aP_b). If at least one distance is greater than a given value (e.g. 10 pixels),the chain is reckoned as to be not a straight line. Repeat to treat the next chain. Otherwise, go to the next step.

Step 2. From one end point P_i of chain C_i reckoned to be a straight line according to step 1, find out any other vector chains C_{i-1} sharing the same end point P_i.

If chain C_{i-1} is also a straight line, and has almost the same direction as that of chain C_i, repeat step 2 until no such straight chains is found out, i.e. until getting chain C_0. From another end point P_{i+1} of chain Ci, do the same process as that from end point P_i of chain C_i i.e. until getting C_n. Finally a vector chain sequence C_0, C_1, $\cdots$ C_n can be discovered.

Step 3. Create straight line L passing through the middle points of C_0 and C_n.

Step 4. For each end point P_i (i=1, $\cdots$; n) find out other chains C_j(j=0, 1, $\cdots$, n) intersecting the chain sequence C'_j (j=0, 1, $\cdots$, n).

For each chain C'_j calculate the new intersection P'_i at which straight line L intersects chain C'_j. The new intersection P'_i is set as the new end point of C'_j.

(2) Classification of polygonal lines

In a topographic map, comparing with the length of polygonal lines forming characters and symbols, the length of contour lines are quite long. After vectorization the polygonal lines can be classified according to their length. The polygonal lines of which length is longer than a given value are tagged to be long lines, otherwise, they are to be short lines.

In general, index contours, e.g. every five or ten meters elevation, are drawn with a wide line, while the other contour lines a thin line. Usually these wide contour lines have numeral arrays indicating their elevation. Therefore, employing pattern recog-

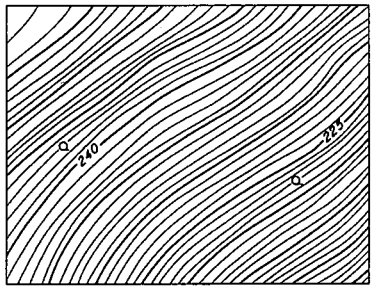


Fig.2 Classifying lines.

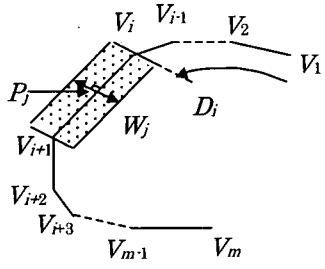


Fig.3 Calculating thickness of a line.

nition techniques for raster image data, the width information of contour lines provide a strong clue to discovering each contour elevation.

The difficulty of classifying the long polygonal lines into two types is that the width of the input raster images has a critical dependency on the original stroke width and scanning resolution. Conveniently, in the case of the contour lines, wide lines are regularly distributed; a certain number of thin lines, e.g. four, are placed between neighboring wide lines where no obstacle exists (see **Fig.2**). Thus, by comparing the width of each long line with that of its neighboring lines, its class can be distinguished.

Let's assume that the length of a long line S is L, its vertices are V_i (i=1, ···, m) as shown in **Fig.3**, and the number of sample points to be tested is n ($n > 0$). The positions of sampling point P_j (j=1, ···, n) located on line segment $V_i V_{i+1}$ are calculated by the following equations:

$$\begin{aligned} P_j &= (L_j - D_i)V_{i+1} + (|V_{i+1} - V_i| \\ &\quad -(L_j - D_i))V_i, \\ L_j &= \frac{j}{n_i + 1}L, \\ D_i &= \sum_{k=2}|V_k - V_{k-1}|, \\ L_j &\geq D_i, \\ L_j &< D_{i+1}, \\ D_0 &= 0. \end{aligned} \tag{1}$$

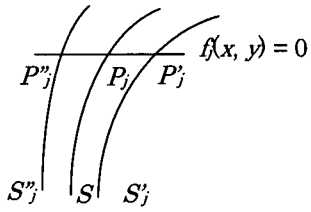


Fig.4 Finding thickness of a line.

After calculating line segment width W_j(j=1, ···, n) at point P_j(j=1, ···, n), these widths are averaged as follows:

$$W_s = \frac{\sum_{j=1}^{n} W_j}{n}. \tag{2}$$

For each sampling point $P_j(x_j, y_j)$ on line S(see **Fig.4**), let's create the following two scan lines to test the width of line S:

$$\begin{aligned} y &= y_j, \\ x &= x_j, \end{aligned} \tag{3}$$

where the total number of testing scan lines for line S is $2 \times n$. For a scan line $f(x_j, y_j) = 0$, let's find out two intersections P_j' and P_j'' neighboring upon point P_j. The long lines containing intersections P_j' and P_j'' are assumed to be S_j' and S_j'', respectively. They are the neighboring lines of current line S at the place of P_j.

Calculate widths $W_{S_j'}$and $W_{S_j''}$ of line S'_j and S''_j, respectively. If

$$\begin{aligned} & W_{S_j} < K \times W_{S_j'}, \\ \textit{and} \quad & \\ & W_{S_j} < K \times W_{S_j''} \end{aligned} \tag{4}$$

are satisfied (K>1.0), line S must not be at least a wide line. For each sampling point P_j (j=1, ···, n) of line S, if the inequalities expressed below are all satisfied, line S is defined to be a wide line:

$$\begin{aligned} & W_S \geq K \times W_{S_j'}, \\ \textit{and} \quad & \\ & W_S \geq K \times W_{S_j''}. \end{aligned} \tag{5}$$

(3) Recognition of numerals

The elevations of index contours are expressed by a set of numerals e.g., a decimal point, and/or a plus/minus symbol. It is not very difficult to recognize these notations by using the structural method[18], if their images are relatively clear. In order to ensure the accuracy, an intelligent reasoning process can be combined to the recognition method, e.g. if a plus/minus symbol exists along

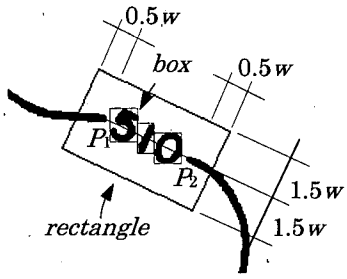


Fig. 5 Picking up elevation figures.

contour lines, it should be located at the first position of a numeral array, and the first one should not be zero. If some images in the map are unclear, some interactive operations will be required to recognize rejected characters as discussed in the following section.

4. EXTRACTING CONTOUR LINES

The process of extracting contour lines consists of the following three steps:

(a) Extract and recognize numerals indicating elevations existing along most wide long lines, and set these values to the corresponding contour lines;
(b) Extract the other contour lines by scanning the grid spread on the map and using continuity of contour lines;
(c) Connect the same elevation contour lines.

(1) Extracting figures of index contours

According to the drawing rules of a topographic map, the figures of an index contour must be placed along its line. The direction of a numeral array is almost the same as the tangent of the contour line segments holding it. The numeral array usually separates the same elevation contour line onto both sides (see **Fig.2**).

Although it is not very obvious that the size of the figures expressing the contour elevation is different from other kinds of small symbols, those figures have the following distinctive features:

(a) Generally, the numeral array is held in a limited area between two end points of wide long lines.
(b) In most cases these two end points belong to two wide-long lines, though in some cases they may belong to one wide-long line when it forms a loop contour line.
(c) The wide line segments of the pair of end points holding the numeral array have almost the same direction as numeral one.
(d) The figures belong to its inherent contour line; neighboring contour lines with figures are separated each other by certain obscured long thin lines, i.e. non-marked contour lines (see **Fig.2**).

For extracting contour lines as many as possible automatically the algorithm positively employs the features mentioned above. It consists of the following three steps.

Step 1. Find out valid pairs of end points:

For any two end points of wide-long lines, if they meet the following fundamental requirements, they can be considered as a valid pair of end points of equi-index contours.

(a) The distance between the pair of end points is within given length (e.g. six figures).
(b) The two wide-long lines can be smoothly connected by a straight line segment which connects the pair of their end points; the angles between the straight line segment (P_1P_2 in **Fig.5**) and both the connected wide-long line segments must be obtuse (e.g. greater than 135°).
(c) The connected line segment does not intersect any other long lines.
(d) In a certain area around these two end points, no end point belonging to wide-long lines exists.

If a pair of end points satisfying the above conditions exists, create a rectangle surrounding the area in which a numeral array is searched. **Fig.5** gives the relationships between the rectangle and one valid pair of the end point, where w is the maximum elevation and/or width of the raster numerals.

Step 2. Pick up the numeral array:

Pick up all the short lines within the rectangle. Create a bounding box to enclose each connected short lines (boxes in **Fig.5**). They make a symbol cluster. From the point of view of the raster image, it is assumed that a single numeral should form a connected area. Because the thinning algorithm can keep the topological structures, the short lines composing each numeral also connects to each other. The numeral array expressing a contour elevation is drawn along a certain direction, i.e. the line segments connecting the centers of every neighboring bounding boxes have almost the same tangential angle. Calculate every tangential angle of these connection lines, and obtain averaged angle α.

Let's assume that the number of connection lines is n, the angle of the ith cluster is α_i $(i=1,\ldots, n)$, and a given threshold angle is $\delta\alpha$ For each angle α_i, if the inequality,

$$|\alpha_i - \alpha| < \delta\alpha, \qquad (6)$$

is satisfied, the cluster is considered as a symbol array. Each symbol of this array is sorted in order.

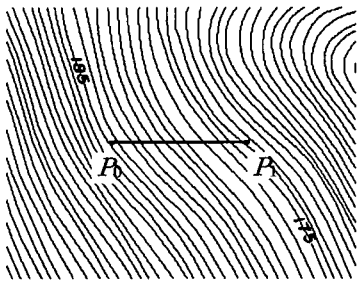


Fig.6 One-way case.

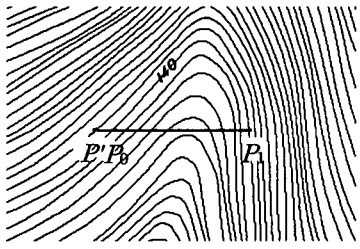


Fig.7 Forth-back case.

Step 3. Recognize index contours:

If the process in step 2 can confirm that each bounding box only contains the data of one numeral, the system can then automatically use a surrounding process of the character to recognize the elevation value. In some cases numeric characters connect to each other or to other lines unfavorably, depending on the quality of maps. In this case, it is desired to use an interactive recognition operation in order to make the system more practical.

Note that not all the symbols, which are within the rectangles made by a pair of end points of wide-long lines, are elevation figures. If the automatic recognition algorithm refuses to recognize any symbol, it blinks. An interactive operation is then required to determine whether it is really figures or not. In the current system, in the following three cases an interactive operation is implemented:

(a) A short line segment composing a numerical character connects to a wide-long line.

(b) The box of connected short lines has an edge, and that length is longer than the maximum length of the edge of the box surrounding a numeral expressing contour elevation, i.e. six figures row length.

(c) Figures of contour elevation cannot be automatically recognized.

During an interactive recognition of the figures, an 'enlarge' operation is automatically carried out to clarify the image of the string. The user can then conveniently input the correct elevation data. After each elevation of contour lines is obtained, they are immediately stored into the data structure together with the wide-long lines connected to the rectangles.

(2) Extracting typical contour lines

Besides the obviously marked index contours exist, there are a lot of non-marked contour lines. According to the continuity of contour lines, the elevation of non-marked contour lines can be searched through reasoning from the marked elevations.

First the grid with a suitable interval (e.g. 200×200 pixels) is spread onto the whole area of the vector map. Then the marked wide-long lines are searched from the top left to the right and then the top to the bottom in order; note that the mark of the elevation of every wide long line is not always given.

Search the points crossing the marked contour lines along a scanning line, and connect each neighboring these points with a line segment called 'reference line segment'. Usually each reference line segment intersects other long lines.There are two typical cases (see **Figs.6** and **7**) and some cases including difficult problems.

a) one way case:

Every intersection of the long lines of which elevations are to be searched and located on the reference line P_0P_1 never belongs to the same long line (see **Fig.6**).

Let's assume that the elevation values of reference points P_0 and P_1 are H_0 (in **Fig.6**, H_0 =175 meters) and H_1 (in **Fig.6**, H_1 =185 meters), respectively, and $H_0 \neq H_1$ is satisfied. If the lines intersecting reference line segment P_0P_1 are all contour lines, the number N of these lines must satisfy the following condition:

$$N = \frac{H_1 - H_0}{d} - 1, \tag{7}$$

where d is a constant expressing of elevation difference between neighboring contour lines (in this example, d=1).

The elevation of each contour line can be expressed by

$$h_i = H_0 + d \times i, \tag{8}$$

where i=1, ⋯;N. Therefore, by counting the number of long lines that intersect the reference line segment, the judgment whether the set of lines is in the one-way case or not is easily done. And then, the elevations of these contour lines are stored.

b) forth-back case:

In this case, at least two intersections that belong to the same long line exist (see **Fig.7**). In a forth-back case, it is not enough to use only the two reference points. In **Fig.7**, two candidates may exist

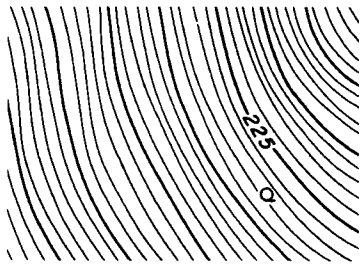


Fig.8 Scan line crosses gap.

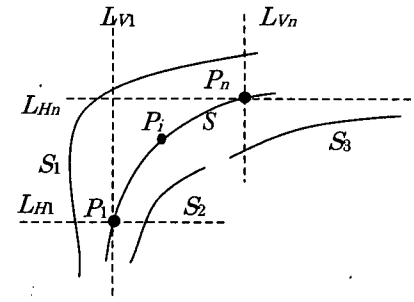


Fig.9 Further processing for extracting contour lines.

when reference points P_0 and P_1 are the same elevation: (from the left 140, 141, 142, 143, 144, 144, 143, 142, 141, 140 meters) or (140, 139, 138, 137, 136, 136, 137, 138, 139, 140 meters). The system needs the third reference point in this case. Let's assume that the elevations of two reference points P_0 and P_1 are H_0 and H_1, respectively, in this case $H_0 = H_1$. The neighboring contour line located at outside the reference point P_0 intersects extended line segment P_0P_1 in direction to the left side at point P', and its elevation is H'. If $H' < H_0$, elevation interval D, which is used for calculating the elevation difference between every neighboring contour lines, is set to be $D = d$, otherwise $D = -d$. All intersections are searched one by one along the reference line segment. Assume that the current elevation is h (the initial value is H_0 at point P_0). If the next intersection exists and it is not point P_1, its elevation is set as $h = h + D$, and the point becomes the current point. If both the next point and the current point belong to the same long line, change the interval elevation value D by $D = -D$, and skip the next point. If the next point is P_1 and $h + D = H_1$ is satisfied, all lines between the reference contour lines can be regarded as contour lines and be stored.

(3) Process for difficult conditions:

Misconception of the contour lines and/or their elevations often happens near the border of a contour field. By aiming at the following features of the contour lines, the possibility of their extraction can be improved.

(a) The raster image of an index contour is drawn by a wide line and in most cases its inherent elevation is entered.

(b) The elevations of index contours can be divided wholly by the specified number, e.g. in case of the scale, 1/1000 or 1/2500, its elevation can be divided wholly by 5 or 10, respectively.

(c) Four intermediate contours drawn by a thin line exist between every neighboring index contour; they can not be divided wholly by the specified number.

Thus if the elevation of a thin line is divided wholly by the specified number, the lines are invalid.

Generally, other kind of long lines such as rivers and roads may also be located in a contour line area. These lines may make the process of contour line extraction confuse. Therefore, it is not enough to validate elevations of these contour lines with only a single examination. If all the process on the elevations can lead to an equi-value, the system determines that the result is valid.

If the results examined by the other reference lines are not the same elevation as these of the previous ones, the system delays the validation on them.

Depending on the image quality and scanning resolutions, a contour line, (it should be theoretically continuous), may be broken up in some places (see upper center area in **Fig.8**). If a reference scan line crosses the gaps of a broken contour line, it will be ignored, and the other lines located near the broken line may give in-accurate results. In this case, the results will not be validated. On the other hand, many small symbols consisting of short lines are located in contour fields also break up long lines (see **Figs.2** and **8**). If such short lines are found, the elevations of the long lines crossing the reference lines will not report correct values. In these cases, validation should also be postponed.

In order to overcome these conditions, the iterative-extraction process for the postponed contour lines should be initiated. At each iteration process, the validated results from previous processing are actively used.

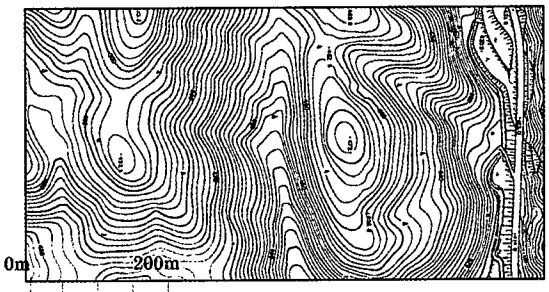
After the processes above are performed, and if there are some lines that have not been properly extracted, further processing should be given:

It is assumed that long lines S_1, S_2 and S_3 are contour lines of which elevations of H_1, H_2 and H_3 ($H_2 = H_3$) are known. Line S is a long line with an unknown elevation.

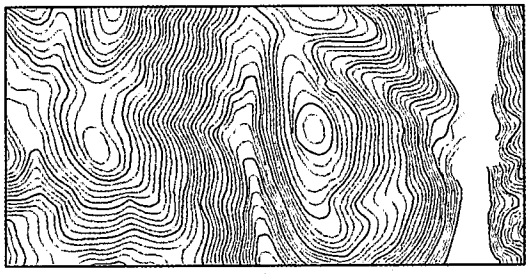
The further processing is:

Step 1: Create several sample points $P_1, \ldots P_i, \ldots, P_n$ with some interval on line S. Make horizontal and vertical lines from each sample points (e.g., lines L_{H1} and L_{V1} from P_1, and lines L_{Hn} and L_{Vn} from P_n).

Step 2: For each sample point, if at least one line (horizontal or vertical line) intersects two contour



(a)



(b)

Fig.10 Example 1: **(a)** raster image, **(b)** extracted contour lines.

lines placed on the two sides of line S, calculate the elevation of that sample point. In **Fig.9**, horizontal line L_{H1} intersects contour lines S_1 and S_2, and the elevation of sample point P_1 is $H_{p1}=(H_1+H_2)/2$. Vertical line L_{Vn} intersects contour lines S_1 and S_3, and the elevation of sample point P_n is $H_{pn}=(H_1+H_3)/2$.

Step 3: If all the elevations H_{pi} (i=1, ...,n) of all sample points are the same (equal to H_p), then long line S is a contour line with an elevation H_p.

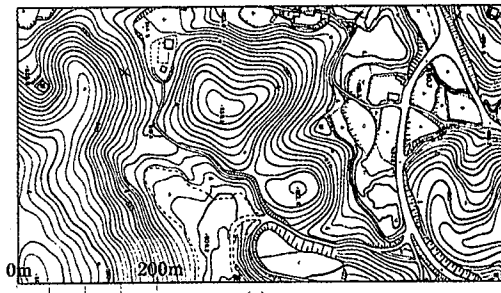
(4) Connecting contour lines

As mentioned before, one contour line must theoretically form a continuous line. In fact, most contour lines may be split into several lines because of cut by some symbols, characters, the lines of roads and rivers, etc., and sometimes due to low resolution. Therefore, the process of connecting these lines to a whole one is required.

In order to make the connection part recover as much as precisely, the system applies the Bezier curves to the gaps.

5. EXAMPLES

The authors have installed the contour line extraction algorithm based on continuity,and applied for more than ten practical terrain maps used foroad



(a)

(b)

(c)

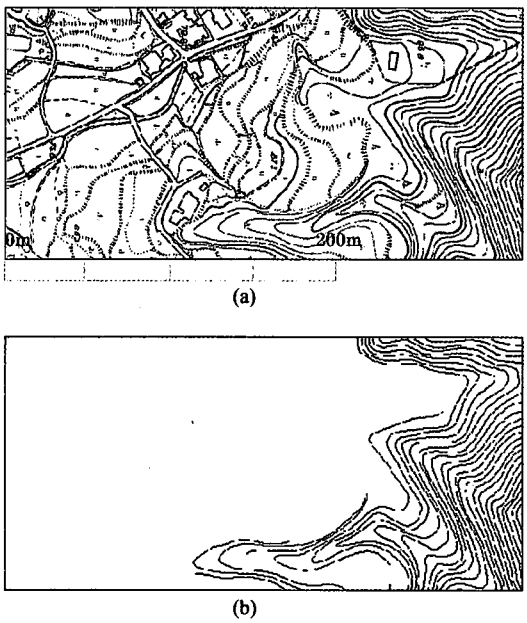
Fig.11 Example 2: **(a)** raster image, **(b)** extracted contour lines, **(c)** raster image except extracted contour lines.

designs(scale factors 1:1,000 and 1:2,500). In the following efficacy of the proposed algorithm is demonstrated by using typical three cases(see **Table 1**). The first one is a mountainous area, most of regions are converted by the contour lines except one highway and two narrow roads, small rice fields, paths, many small symbols etc. The second one is a village area located at the foot of mountains. And the third one is the suburbs surrounded by several hills and large rice fields.

Fig.10(a) shows a part of original raster image(about one sixtieth of the whole image) of example 1. The extracted contour lines are displayed in **Fig.10(b)**. Needless to say it is the easiest to extract the contour lines because of few other objects except the contour lines. In example 2 as shown in **Fig.11(a)** some local loads, their cut/back, rice fields, etc. separate the contour lines. **Fig.11(c)**

Table 1 Detail results of three typical examples.

No. of Examples		1	2	3
(Scale)		1:1,000	1:2,500	1:2,500
Image Size (pixels)	Width	28,816	28,816	28,816
	Height	38,979	32,379	35,137
Polylines		33,405	34,854	92,288
End-points		39,561	41,187	122,437
Figures of Contour Elevations	Pairs of End-points(a)	184	36	66
	Automatically Recognized Numeral Arrays(b)	180	32	56
	(b)/(a)	0.98	0.89	0.85
Long Lines	Wide-Long Lines	2,981	2,950	5,424
	Long-fat Lines	725	533	942
Contour Lines	Total Contour Lines(c)	843	380	613
	Extracted Contour Lines(d)	817	321	478
	Mis-extracted Contour Lines(e)	3	4	3
	Un-extracted Contour Lines(f)	23	55	132
	(d)/(c)	96.9%	84.5%	78.0%
	(e)/(c)	0.4%	1.1%	0.5%
	(f)/(c)	2.7%	14.4%	21.5%



(a)

(b)

Fig.12 Example 3: **(a)** raster image, **(b)** extracted contour lines.

is the raster image in which the contour lines of **Fig.11(b)** was erased; this figure suggests that extracting algorithms of these objects and finding out symbols such as elevations, trees, etc. must become easy. In example 3 as shown in **Fig.12 (a)** and **(b)** many contour lines are cut and/or surrounded by other figures: roads, rivers, rice fields, buildings etc. Thus, only one or sometimes none of numeral arrays of figures expressing the elevations of contour lines can be picked out.

The results of **Table 1** obviously reflect these facts; the correct extraction percentages of examples 1, 2 and 3 decrease 96.9, 84.5 and 78.0% in order, contrarily the numbers of un-extracted contour lines increase 2.7, 14.4 and 21.5% in order. The wrong extracted ones are very few, i.e. most of all are less than 1%. These results demonstrate that the proposed algorithm is sufficient for practical use.

6. CONCLUSION

The authors have proposed an algorithm to automatically extract contour lines of topographic maps based on the continuity of contour lines. The quality of a raster image can be measured by the degree of how many isolated lines and symbols are connected and how many continuous lines are split. If the quality is not so bad and the contour fields mixture little of other symbols, e.g. a mountain area and country sides, the program proposed can extract great majority of contour lines. The graphic editor is useful to correct the mis-extracted lines and pick up un-extracted lines.

The following tasks should be solved in the near future:

(a) Besides the elevation representing contour lines, there are many elevation points such as a small

point indicating the top of a mountain.If the proposed algorithm can extract and recognize the information, the probability of extracting contour lines becomes higher.

(b) There are a quite few short contour lines located near the frame of a topographic map. The system should treat this problem, so the extraction percentage must rise up more.

(c) In a few cases the current system extracts non-contour lines e.g. roads and rivers, how to employ an intelligent judgment techniques to make the recognition probability rise up must be future work.

REFERENCES

1) Ablameyko, S., Bereishik, V. and Kryuchkov, A.: Automated Map-Drawing Digitizing Technology Based on Scanner Input, *it MVA'94 IAPR Workshop on Machine Vision Applications*, pp.48-51, Dec.13-15, 1994.

2) Suzuki, S. and Yamada, T.: Maris:Map Recognition Input System, *Pattern Recognition*, Vol.23, pp.919-933, 1990.

3) Boatto, L., Consorti, V., Buono, M., D., Zenzo, S.D., Eramo, V.,Esposito,A.,Melcarne,F.,Meucci,M.,Morelli,A.,Mosciatti, M.,Scarci,S. and Ticci,M.: An Interpretation System for Land Register Maps, *IEEE COMPUTER*, Vol.25, No.7, pp.25-33, 1992.

4) Kasturi, R., Fernandez, R., Amlani, M.L. and Feng, W.: Map Data Processing in Geographic Information System, *IEEE COMPUTER*, pp.10-21, Dec. 1989.

5) Hayakawa, T., Watanabe, T., Yoshida, Y. and Kawaguchi, K.: Extraction of Topological Road-Network from an Urban Map, *IEICE D-II*, Vol. J74-D-II, No. 6, pp.757-765, June 1991(in Japanese).

6) Kim,W., Hirai, Y., Furukawa, T. and Arita, H.: Extraction of Road Segments by Spatial Filters, *IEICE D-II*, Vol. J76-D-II, No. 3, pp.566-574, May 1993(in Japanese).

7) Akagi,Y., Ito, H. and Kumamoto, A.: Extracting Road Information by Collating Multiple Maps, *T.IEE Japan*, Vol. 118-C, No. 12, pp.1722-1729, 1998(in Japanese).

8) Jiao, G.F., Nakamae, E., Tadamura, K. and Inuyama, H.: On the Extraction of Various Regions in Vector Maps, *MVA'94IAPR Workshop on Machine Vision Applications*, pp.230-234, Dec.13-15, 1994.

9) Amin, T.J. and Kasturi, R.:Map Data Processing: Recognition of Lines and Symbols, *Optical Engineering*, Vol.26, pp.354-358, 1987.

10) Yamada, H., Yamamoto, K., Saito, T. and Matsui, S.:MAP: Multi-Angled Parallelism for Feature Extraction from Topographical Maps, *Pattern Recognition*, Vol.24, No.6, pp.479-488, 1991.

11) Yamada, H., Yamamoto, K. and Hosokawa, K.:Directional Mathematical Morphology and Reformalized Hough Transformation for the Analysis of Topographic Maps, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.15, No.4, pp.380-387, April 1993.

12) Mizutani, N., Watanabe, T., Yoshida, Y. and Okabe, N.: Extraction of contour Lines Using Neighboring Relationships Based on the Volonoi-Line Graph, *IEICE D-II*, Vol. J74-D-II, No.11, pp.1499-1506, Dec. 1991 (in Japanese).

13) Mori, M., Seto, H. and Nakamura, A.: Automatic Extraction of Contour Lines in a Topographical Map and Its Application, *IPSJ*, Vol. 29, No. 3, pp. 221-232, Mar. 1988.

14) Nieuwenhuizen, P.R.V., Kiewiet, O. and Bronsvoot, W.F.: An Integrated Line Tracking and Vectorization Algorithm, *Computer Graphics Forum Eurographics'94*, Vol.13, No.3, pp.c349-c359, 1994.

15) Sriharii, S.N. and Govindaraju, V.: Analysis of Textual Image Using the Hough Transform, *Machine Vision and Applications*, pp.141-153, 1989, 2.

16) Tan, J.R. and Peng, Q.S.: A Global Approach for Line Recognition Based on the Scanned Image of Engineering Drawings, *Proc. of 3rd Intel. Conf. on CAD/CG*, pp.815-821,1993.

17) Wu, R.Y. and Tsai, W.H.: A New One-Pass Parallel Thinning Algorithm for Binary Images, *Pattern Recognition Letters*, pp.715-723, 13, Oct. 1992.

18) Khan, N.A. and Hegt, H.A.: Recognition of Hound-Written Numerals Based on a Structural Approach, *Proc. of IASTED Inter. Conf. Signal and Image Processing(SIP'98)*, Las Vegas, Nevada, USA, pp.593-597, Oct.28-31, 1998.

(Received November 6,2000)

ラスタ地形図の特徴を用いた等高線の抽出

李偉青・中前栄八郎・三好孝治

ラスタ地形図からの等高線の自動抽出とその固有の高度を求めることは困難な作業である．本論文では，等高線間の連続性に注目した自動抽出アルゴリズムを提案する．すなわち、一つの等高線のすべての点は同一標高値を示し，すべての隣接等高線の高度は異なりほとんどの巾広の等高線はそれ自身の高さを示す数字をもっている．これらの情報を活用することによって，等高線の形状と高度を自動的に検出し認識することができる．提案手法の有用性が幾つかの実用例で示されている．