MVA'94 IAPR Workshop on Machine Vision Applications Dec. 13-15, 1994, Kawasaki

RECONSTRUCTION OF 3-D TERRAIN DATA FROM CONTOUR MAP

Jin-Seon Lee and Seung-Jong Chung
Department of Computer Engineering
Chonbuk National University

Dukjin-dong 664-14,
Chonju, Chonbuk 560-756, Korea

ABSTRACT

This paper presents a raster-based algorithm for reconstruction of 3-D terrain data from contour information appearing on the conventional 2-D map. The height of an interior point between two neighboring contours is defined by a linear interpolation formula whose variables are the height of those contours and distance from the point to the contours. The distance from an interior point to the contour is computed by using the distance transformation operation. The distance transformation is a raster-based operation which computes the value of a pixel using the values of neighboring pixels. So our algorithm has the advantage of easy implementation, and it is fast since it performs only simple computation during raster scanning. Also the experimental results show that the reconstructed terrain preserves faithfully the terrain information appearing on the contour map.

1. INTRODUCTION

The 3-D terrain data have been used effectively by GIS(Geographical Information System) for several decades for many purposes such as displaying the terrain in meshed or shaded form, and planning automatically the roads or missile paths [1, 2]. So, in map data processing, how to obtain these terrain data is one of the major problems [3].

With stereo vision technique which uses two or more aerial photographs, we can obtain terrain data for a particular area. One of the most difficult problem in this technique is the correspondence problem which is not solved completely yet [4]. Another difficulty is the large cost required in acquiring the source images by airplane or satellite. Alternative solution is to use contour information on the conventional paper maps which represents the terrain information implicitly. In this solution, the main point is how to interpolate height value at an interior point from the height values and shapes of the neighboring contours.

The method popularly used considers points on the contours as randomly distributed and applies spatial interpolation algorithms like Kriging algorithm [2]. It does not use the useful property of contours that they are composed of connected points. Another method is to divide the contours into small line segments, convert them into TIN(Triangulated Irregular Network) structure by triangulating them, and interpolate smooth surfaces from each triangle [5]. Due to its useful properties, TIN is used by most of commercial GIS. A problem of TIN method is that when we vectorize the contours, some information loss is inevitable. Also it needs solution for each of four subproblems which are not trivial [6], and whose implementation is hard. Another method chooses the most reliable vertical or horizontal line, and interpolates heights of all the points on that line [7]. It repeats the same process until all the points determine their heights.

In this paper, we present a new method which efficiently reconstructs 3-D terrain from contour map. The height of an interior point between two neighboring contours is defined by a linear interpolation formula whose variables are the heights of those contours and distance from that point to the contours. The distance is length of the shortest path from the point to the contour which never crosses any other contour. We believe that this definition is reasonable for various situations existing in the actual maps. Our algorithm is based on the distance transformation algorithm proposed by Borgefors. To compute the distance for all the interior points simultaneously, we modify Borgefors's algorithm. Since the distance transformation algorithm is a raster-based algorithm which requires simple arithmetic operations, our algorithm has some advantages of being fast and easy to implement. The experimental results show that the terrain reconstructed by our algorithm faithfully preserves the information contained implicitly in contour map.

Section 2 formulates the reconstruction problem, and Section 3 describes our algorithm to compute the distance from interior points to neighboring contours. Experimental results are shown in Section 4. Section 5 concludes the paper.

2. FORMULATION OF HEIGHT INTERPOLATION PROBLEM

Contours appearing on the conventional paper map comprise many layers and the outer contour completely encloses the inner ones. A contour forms a closed curve, and all the points on a contour have the same height. The difference between heights of two neighboring contours is constant. The height of an interior point lying between two neighboring contours has a value between the height values of those contours.

The reconstructed height of an interior point must satisfy some important properties in order to produce a pleasant terrain, though the exact solution does not exist. First, height of an interior point must be decided according to how much it is close to the inner and outer contour. For example, if an interior point is very close to the inner contour and far from the outer contour, the height of this point must be close to that of the inner contour. Second, the reconstructed terrain must be smooth such that there exists no spike or pit. Third, in proceeding to inner contour from outer one, the height must continuously increase or decrease.

The main point in our reconstruction problem of 3-D terrain data from contour map is to define mathematically how much an interior point is close to two neighboring contours. The definition should satisfy above properties on quality of the reconstructed terrain, and should lead to a good computational efficiency.

Considering above remarks, we define the distance from an interior point to a neighboring contour as length of the shortest path which connects the point and the contour and never cross any other contour.

Now we define some terms, and by using these we can define the height interpolation formula. In Figure 1, an outer contour is termed as C_{out}, and an inner one as C_{in}. The height of C_{out} and C_{in} is H_{out} and H_{in}, respectively. An interior point whose height we want to know is termed as P. The distance from P to C_{out} and C_{in} is D_{out} and D_{in}, respectively.

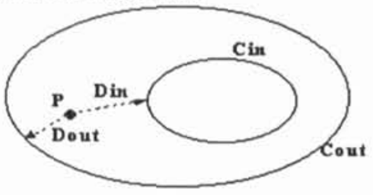


Figure 1. Some terms used in height interpolation formula.

Then, the height at P is defined by usual linear interpolation formula as below.

$$H = \frac{D_{out}}{D_{out}+D_{in}} * (H_{in} - H_{out}) + H_{out} .$$

3. DISTANCE COMPUTATION

When we formulate the reconstruction problem as above, the most difficult problem is how to compute D_{out} and D_{in}, the distance from an interior point to two neighboring contours. Our idea is to use the distance transformation algorithm for the computation of D_{out} and D_{in}. Distance transformation is a raster-based operation that transforms an image consisting of object pixels and non-object pixels into the image in which the non-object pixel has the distance value to the nearest object pixel. We use Borgefors's algorithm [8] with some modifications to be suitable to our application. In this section, we first describe briefly the Borgefors's distance transformation algorithm and then describe our algorithm for computation of the distance which is based on Borgefors's algorithm. Also, we analyze the computational and memory efficiency of our algorithm.

3.1 BORGEFORS'S DISTANCE TRANSFORMATION ALGORITHM

Borgefors's distance transformation algorithm is described as follows. The algorithm makes an initial image I in which object pixels are set to 0 (it means that the distance is 0) and non-object pixels to L (it means the infinite distance). For L, a sufficiently large value is used. To compute the distance from non-object pixels to the nearest object pixel, the algorithm uses two masks shown in Figure 2. In these masks, pixel represented by 0 is center pixel and value of other pixels is local distance to the center pixel (a=1 and b=$\sqrt{2}$).

Performing a forward scanning on the initial image I and then performing a backward scanning on the result image, the process of distance transformation is completed. During raster scanning, forward mask is

F=

b	a	b
a	0	

Forward mask

B=

	0	a
b	a	b

Backward mask

Figure 2. Distance transformation masks.

applied from left top to right bottom on image, and backward mask is applied from right bottom to left top. During scanning, value of the current pixel is replaced by the minimum value among five values obtained by adding the mask values and the corresponding pixel values of the image. A pseudo code for the algorithm is as follows.

```
Forward scanning :
  for j = 2, ... , YRES - 1    do
  for i = 2, ... , XRES - 1    do
      $I[i][j] = \min_{(x,y)\in F} (I[i+x][j+y]+F[x][y])$
Backward scanning :
  for j = YRES - 1, ... , 2    do
  for i = XRES - 1, ... , 2    do
      $I[i][j] = \min_{(x,y)\in B} (I[i+x][j+y]+B[x][y])$
```

XRES and YRES is the image resolution, I[i][j] is the value of pixel at the position (i,j). (x,y) is position of a pixel in the mask ($-1\leq x\leq 1$, $-1\leq y\leq 1$). F[x][y] and B[x][y] is the local distance (0, +1, or $+\sqrt{2}$) at a position (x,y) in forward mask and backward mask, respectively.

3.2 ALGORITHM FOR DISTANCE COMPUTATION

As shown in Figure 1, our problem is to compute both of D_{out} and D_{in} for all the interior points P between every pair of neighboring contours. So our problem is more complex than the pure distance transformation problem. We should modify Borgefors's algorithm to be suitable to our problem. The difficulty in applying Borgefors's algorithm to the problem for computation of the distance D_{out} and D_{in} is illustrated in Figure 3. In case of contours in this figure, the distance from P to C_{out} has been defined as D_{out} in Section 2. However, if we designate only the points on C_{out} as object pixel, D_{out}' is computed as the distance by Borgefors's algorithm. Also if we designate all the points on C_{out} and C_{in} as object pixel, D_{out}'' is computed as the distance. Thus we need to define three kinds of pixel.

The points on C_{out} are set to value 0 as object pixel, the interior points between C_{out} and C_{in} set to L (sufficiently large value) as non-object pixel, and the points on C_{in} set to W (L+1) as wall pixel. W-pixels play the role as a wall which the path should not cross. We use maximum integer value a pixel can have for W and W-1 for L. For example, if one byte per pixel is used, W is 255 and L is 254.

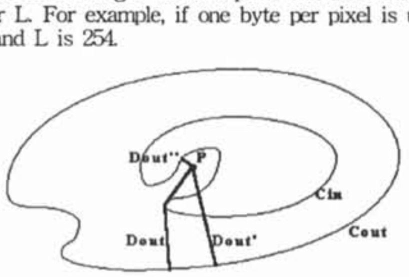


Figure 3. Difficulty in applying Borgefors's algorithm to our problem.

Till now, we consider the procedure for only one pair of neighboring contours C_{out}-C_{in}. If D_{out} and D_{in} is separately computed for each contour pair, computational time is required so much in case that many contours exist because number of image scannings is excessively large. So we should design an algorithm which computes D_{out} and D_{in} concurrently for every pair of neighboring contours with a constant number of image scannings. For this, we make an initial image as follows. In Figure 4, contours with odd numbers (i.e., C_1, C_3, C_5) are initialized to 0, contours with even numbers to W, and all the interior points in the image to L. After initialization, we perform the forward scanning and backward scanning on the initial image as shown in the following code. (To avoid the floating-point computations, we use 3 and 4 instead of 1 and $\sqrt{2}$ for a and b in the mask, respectively.)

```
Forward scanning :
  for j = 2, ... , YRES - 1     do
  for i = 2, ... , XRES - 1     do
    if (I[i][j]≠W)
      I[i][j]= minimum       (I[i+x][j+y] + F[x][y]))
            (x,y)∈F AND I[i+x][j+y]≠W
Backward scanning :
  for j = YRES - 1, ... , 2     do
  for i = XRES - 1, ... , 2     do
    if (I[i][j]≠W)
      I[i][j]= minimum       (I[i+x][j+y] + B[x][y]))
            (x,y)∈B AND I[i+x][j+y]≠W
```

In the result image, interior points in R_i with even number for i have the value D_{out} to C_{i-1}, and interior points in R_i with odd number for i have the value D_{in} to C_i. For example, interior points in R_4 have the value D_{out} to C_3 and interior points in R_3 have the value D_{in} to C_3.

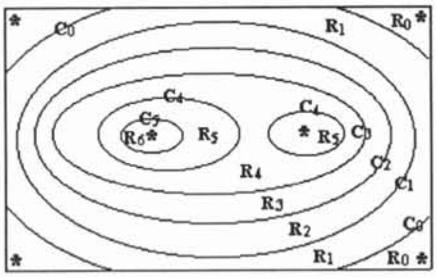


Figure 4. Example contours for describing our algorithm.

Now, we exchange the role of contours with odd number and ones with even number. That is, contours with even number (i.e., C_0, C_2, and C_4) are initialized to 0, contours with odd number to W, and all the interior points to L. Then we perform the forward scanning and backward scanning to the initialized image. In the result image, interior points in R_i with odd number for i have the value D_{out} to C_{i-1}, and interior points in R_i with even number for i have the value D_{in} to C_i.

We should also mention the followings in describing our algorithm. First, all contours should be 4-connected. If they are not, the computed distance is not guaranteed to be to the neighboring contours because the wall pixel cannot play its role properly. Second, as shown in Figure 4, outermost contour is broken into several segments, and outermost and innermost contours have only one neighbor. So they must be handled specially. We assign same value to the segments belonging to a contour. Also, in case that the neighboring contour is only one, we put a point (in Figure 4, represented by *) at a suitable position with a suitable height. This point is treated as another neighboring contour.

3.3 EFFICIENCY ANALYSIS

Timing and memory efficiencies of our algorithm are analyzed as follows.

(1) Timing efficiency

To compute D_{in} and D_{out} of all the interior points, we repeat twice a same sequence of the processes: initialization, forward scanning, and backward scanning. In each of the initialization, forward scanning and backward scanning, one image scanning is needed. Thus six image scannings are performed totally. In addition, extra scannings (such as scanning for assigning identity number to each contour) are needed about three times. Thus by less than ten image scannings, D_{in} and D_{out} for all the interior points are computed. This number of image scannings is independent on number of input contours. All the operations performed during an image scanning are simple ones such as assignment of initial values, comparison and addition of integer numbers. After obtaining D_{in} and D_{out}, to compute the height of an interior point, only simple floating-point operations (in the interpolation formula in Section 2, one floating-point division, one multiplication, and one addition) are needed. Concluding, our algorithm has high computational efficiency because only fixed number of image scannings, simple integer operations, and limited number of floating-point operations are needed.

(2) Memory efficiency

In our algorithm, to store the computed D_{in} and D_{out}, two 2-dimensional arrays with same size as the input image I are needed. Also small amount of additional memory is needed.

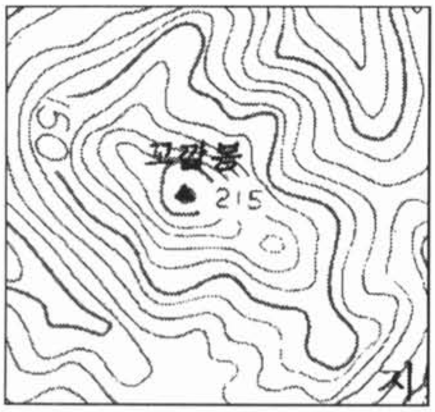
4. EXPERIMENTAL RESULTS

Image capturing and contour extraction are performed on IBM PC/486 computer. Figure 5(a) shows an example image input by a scanner from an actual paper map. A paper map for Chonju city located southwest in Korea with scale of 1:25,000 is used for our experiment. The size of input area in map is about 2.5cm*2.5cm. Actual area is a part of surrounding region of Chonju city whose size is about 0.6km*0.6km.

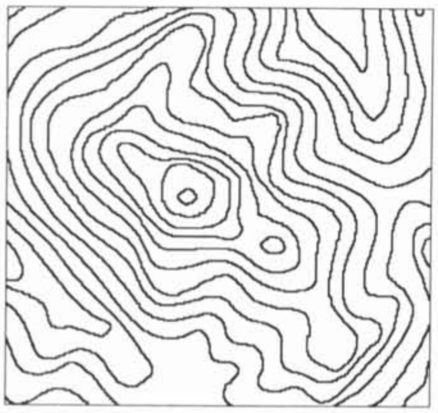
Since original image is a color image, we separate RGB component images by using Photostyler image editing software, and select a component image which best preserves contour lines while keeping other symbols to the minimum. Figure 5(a) shows the image obtained by above procedure. From this image, we remove letters and symbols, and connect the broken contour lines. One-pixel thick contour lines are obtained by performing the SPTA thinning algorithm [9]. The postprocessing for contour lines to have 4-connectivity is performed. The result image is shown in Figure 5(b).

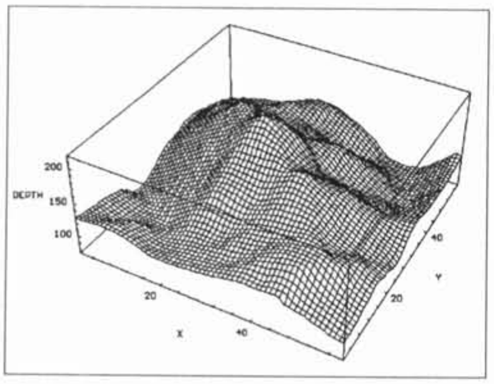
The proposed reconstruction algorithm is implemented in C language under SUN SPARC station 10. Figure 5(c) shows the 3-D terrain reconstructed by our algorithm in meshed form. In this figure, we can see that the peak at 215m appears correctly near center of the image. Another peak appears at right below of the center peak. Several areas can be also seen like a valley at top right, abrupt descent at middle left, and planar region at bottom middle.

To illustrate the exact form of the reconstructed terrain, we overlap the sectional shapes for three horizontal lines on the corresponding positions in Figure 5(d). This figure shows that the reconstructed terrain preserves faithfully the terrain information implicitly appearing on the contour map.



(a) input map image (295*288 spatial resolution)


(b) extracted contour map image



(c) meshed-form display of reconstructed 3-D terrain

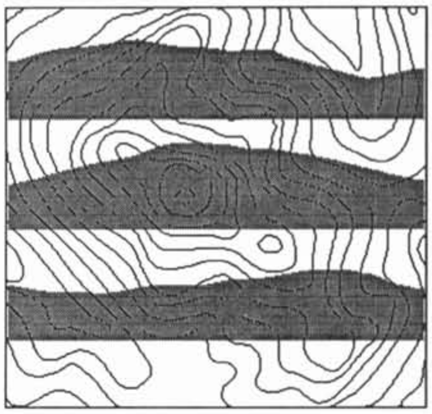

(d) cross-sectional display of reconstructed 3-D terrain

Figure 5. Reconstruction of 3-D terrain from contour map.

5. CONCLUSION

We propose an efficient raster-based algorithm for reconstruction of 3-D terrain from contour map extracted from a 2-D paper map. The algorithm performs a fixed number of image scannings, and performs only simple computations during image scanning. Also, experimental results show that the reconstructed terrains preserve faithfully the terrain information appearing implicitly on contour maps.

There are some further research topics such as automatic extraction of contours, and automatic assignment of numbers and height values to the extracted contours, merging the reconstructed terrains at their common boundaries after separately reconstructing them.

REFERENCES

[1] R. Kasturi, et al., "Map data processing in geographical information systems," *IEEE Computer*, Vol. 22, No. 12, pp.10-21, December 1989.

[2] P.A. Burroughs, *Principles of Geographical Information Systems for Land Resources Assessment*, Oxford University Press, 1986.

[3] S. Viseshsin and S. Murai, "Automated height information acquisition from topographic map," *Proceedings of IAPR Workshop on Machine Vision Applications*, pp.219-221, November 1990.

[4] B.K.P. Horn, *Robot Vision*, MIT Press, 1986.

[5] ESRI Inc., *ARC/INFO User's Guide: Surface Modeling with TIN*, 1991.

[6] D. Meyers, et al., "Surfaces from contours," *ACM Transactions on Graphics*, Vol. 11, No. 3, pp.228-258, July 1992.

[7] K. Cheng and M. Idesawa, "A simplified method of data form conversion from contour line surface model to mesh surface model," *Proceedings of IEEE International Conference on Pattern Recognition*, pp.582-585, 1986.

[8] G. Borgefors, "Distance transformations in digital images," *Computer Vision, Graphics, and Image Processing*, Vol. 34, pp.344-371, 1986.

[9] N.J. Naccache and R. Shinghal, "SPTA: a proposed algorithm for thinning binary patterns," *IEEE Trans. on SMC*, Vol. 14, No. 3, pp.409-418, May 1984.