Contour Line Extraction from Color Images of Scanned Maps

Marc Lalonde and Ying Li

Knowledge-Based Systems Group, Centre de recherche informatique de Montréal, 1801 McGill College Avenue, Suite 800, Montréal, Québec, Canada H3A 2N4

Abstract. This paper reports our work on contour line extraction from color images of scanned maps. Color processing extracts the basic colors of an image by switching from RGB to L*a*b color space, projecting the image on its principal axes and using modified histogram splitting. After the user has selected one of the extracted colors, pixel color classification yields a binary image that is thinned for connected-component analysis. Line extraction and merging follows, along with contour line formation based on connectability between the line segments.

1 Introduction

Contour lines in topographic maps provide terrain information for a geographical information system (GIS) application such as constructing 3D models of strategic areas for flight or navigation simulation. Current map digitization process requires hours of an operator's manual work on a single map, and consequently, there is a need for an automatic or semi-automatic data collection tool as a preprocessing module in a generic GIS application design environment.

Automating contour line extraction from color images is not a trivial task. Color analysis is complicated by the poor quality of the images produced by the scanner and the use of dithering in the map printing process. The contour lines are often broken due to the overlaying of other map objects. Our contributions are twofold. 1. Colors in a color image are extracted by switching from RGB to L*a*b color space, projecting the image on its principal axes, and performing modified histogram splitting; 2. Line segments are extracted and merged to form contour lines based on connectability between them.

In the following, Section 2 reviews some related work on contour line extraction. Section 3 describes the color processing algorithms applied to color map images. Section 4 describes some line manipulation algorithms working on the binary images produced by the color module. In Section 5, contour line formation is explained, followed by preliminary results presented in Section 6. Finally, Section 7 outlines the future work required to complete the project.

2 Related Work on Contour Line Extraction

Work on automatic contour line extraction is relatively limited, and it is usually presented as an extra feature of a basic map processing package. For example, the

recognition system of Suzuki et al. [1] can perform extraction of buildings and various other constructs made of lines such as railways, roads, and contour lines. After some preprocessing, thinning of the original binary image is performed and lines of all widths are extracted. Of the extracted lines, contour lines are isolated using the knowledge about their widths and layout. The extraction rate (the ratio of the total length of correctly extracted lines to the total length of the lines to be found) was 82%. This work led to the development of a complete map recognition system, MARIS [2], which has a recognition rate of about 90% for contour lines.

Ebi et al. [3] present their system designed for extracting GIS information from color maps. A map image is sliced into layers, each corresponding to one color, and map objects (including contour lines) are extracted from each layer. To separate color layers, color pixels are transformed from RGB to the L*u*v color space, and the color clusters are determined through the detection of peaks in the histogram of the $u'v'$ coordinates.

Shimada et al. [4] adopt an agent-based method, where each agent is responsible for detecting a contour line, and a supervisor agent makes sure that all agents complete their tasks and talk to each other in case of ambiguous situations (e.g. touching lines). The system is semi-automatic since an operator is asked to draw cutting lines between ridges and valleys so that starting points could be assigned to the agents. No recognition results are given.

In a closely related domain, Taniguchi et al. [5] present a system for the automatic interpretation of weather reports, in which a sub-module is in charge of extracting lines of constant pressure from weather maps.

3 Color Processing

Our color processing stage for dividing a map image into color layers has two modules: 1. color extraction that analyses an image and extracts its colors; 2. color segmentation that classifies image pixels according to a selected color. The interface to the system is such that a color palette is displayed to the user who selects a color found in the map image. This allows the user to easily input the color of the contour lines to be extracted by clicking on the relevant color patch.

3.1 Extraction of the Colors

The low quality of the scanner combined to the use of dithering in the map printing process complicate the design of a reliable color extraction module. The very high spatial frequencies on the map cause aliasing during the pixel sampling, which results in the production of new colors around the dithering dots and along edges. Smoothing the image with a three-dimensional median filter [6] would remove some noise but also destroy the thin contour lines.

Our approach is based on the histogram splitting algorithm of Tominaga [7] with the following measure to reduce the dithering effect.

- The image is first thresholded with respect to its saturation component. The saturation of a pixel with RGB color values (r,g,b) is defined as

$$\text{Saturation} = 1 - 3 \cdot \frac{\min(r, g, b)}{r + g + b} \quad . \tag{1}$$

  The resulting image is interpreted as a mask that is used for processing the chromatic and achromatic layers of the map separately.
- A connected-component analysis is performed over the color mask image. When small blobs are considered to be part of dithering regions, their pixels are also masked.
- The remaining pixels are remapped from the RGB to the $L^*a^*b^*$ color space and passed to the histogram splitting process:
  - the L^*a^*b image is decomposed into its principal components. Each color vector $\mathbf{c}$ is recomputed as $\mathbf{c}' = \mathbf{U^t}(\mathbf{c} - \mathbf{m})$, where $\mathbf{m}$ is the mean color vector computed over all non-masked pixels, and $\mathbf{R}$ is the covariance matrix of $\mathbf{c}$, while $\mathbf{U}^t$ is the result of the decomposition of $\mathbf{R}$ into its eigenvectors.
  - The histogram of the first principal component is constructed. The contribution to the histogram is weighted according to the neighborhood of the pixel and the histogram is smoothed.
  - Upon determination of the dominant peak, the pixels falling into the corresponding region are given a unique label and are also masked to withdraw them from the process.
  - If the histogram of the first principal component is noisy or is unimodal, the second, and possibly third, principal component is analyzed. Histogram splitting ends when the histograms are either unimodal or noisy.

Extraction of the dominant grey levels (using the achromatic mask image) follows the same idea. Only a section of the image with a rich color contents is sufficient to capture the dominant colors.

3.2 Pixel Classification

A classification rule evaluates if a pixel belongs to the class of interest. We use a nearest-neighbor rule in the RGB color space with a rejection threshold set empirically to 2σ, where σ is the standard deviation of the color cluster center in the RGB space as measured after color extraction. Region growing refines the classification by merging unlabeled pixels whose hue difference with respect to the mean hue of the neighboring blob is lower than a threshold. The hue value of a pixel with RGB values (r,g,b) is an angle between $-\pi$ and π and is given by:

$$\text{hue} = \text{atan}\left(\frac{\sqrt{3} \cdot (g - b)}{2r - g - b}\right) \quad . \tag{2}$$

Although sharing elements with Ebi et al. [3], our algorithm has advantages: 1. Separation of the image into chromatic and achromatic planes increases resolution; 2. The use of principal component analysis yields more discriminant

images, and thus richer histograms; 3. The introduction of weights on pixels according to their neighborhood is new.

4 Line Extraction and Merging

Line extraction is performed on the binary image yielded by the color segmentation module. The image is thinned using Guo and Hall's algorithm [8] and the pixels are labeled as LINELINK (pixels in the middle of a line segment), FREEOL (pixels at endpoints), or LINEJUNC (pixels at junctions). To find the connected pixels that form lines, the image is scanned until a pixel marked as LINELINK is found. The neighboring LINELINK pixels are recursively extracted from the image until FREEEOL or LINEJUNC pixels are reached.

Line merging is needed to amend either imprecise segmentation or line breaks produced by other map items overlaying the contour lines. Two cases for merging are depicted in Fig. 1. The first rule (Fig. 1a) merges L_1 with L_2 instead of L_3 if the ratio of line_dist(L_1, L_2) to line_dist(L_1, L_3) is within a proximity threshold, where the line distance line_dist(L_i, L_j) is the minimum of the distances between the endpoints of L_i and L_j. This rule would not merge A to B in Fig. 1b because too many possible connections are allowed in the restricted radius around A. Let point_dist(p, q) be the Euclidean distance between points p and q. We add a second rule to merge between A and B if

$$\text{point\_dist}(A, B) < \text{point\_dist}(A, C) \ , \text{ and} \tag{3}$$

$$\text{point\_dist}(A, D) > \text{point\_dist}(A, B) + \text{point\_dist}(B, C) \ . \tag{4}$$

These merging rules usually reduce the number of lines found in the image by one half.

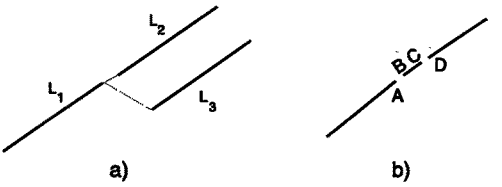

Fig. 1. Merge cases

5 Contour Line Formation

To form contour lines, the line segments obtained from line extraction and merging are chained into closed curves, for which confidence values are computed for determining whether they form valid contour lines.

To chain segments into closed curves, candidate connections for each line end are selected based on a smoothness index. A recursive algorithm selects a line with length above a threshold as the top node and builds a connection tree by connecting more line segments until a closed curve is obtained. When a closed contour is found, i.e. when the free endpoint of the last concatenated segment is close to the free endpoint of the starting line, the child node in the tree is labeled accordingly. Retrieving all combinations of closed contours in the tree is thus easy (tests are made to ensure that all contours are unique).

The smoothness index for each possible connection is computed based on the lines' lengths, the tangent vectors at the endpoints to be connected, and the gap between their endpoints. In cases where a line end is too noisy, it is marked as unreliable and a crude approximation is used for the tangent vector. When the gap is small (a few pixels, as in Fig. 2a), the smoothness is measured as

$$\text{smoothness} = \mathbf{tg}_{L_i} \cdot \mathbf{tg}_{L_j} \ , \tag{5}$$

if the two line ends are reliable. Otherwise, the smoothness is set to 1 (the maximum value). When the gap is large, then an additional vector $\mathbf{v}_{ij}$ joining the two endpoints (Fig. 2b) is used:

$$\text{smoothness} = \begin{cases} \min(\mathbf{tg}_{L_i} \cdot \mathbf{v}_{ij}, \mathbf{v}_{ij} \cdot \mathbf{tg}_{L_j}) & L_1 \text{ and } L_2 \text{ reliable} \\ \mathbf{tg}_{L_i} \cdot \mathbf{v}_{ij} & L_1 \text{ reliable} \\ \mathbf{v}_{ij} \cdot \mathbf{tg}_{L_j} & L_2 \text{ reliable} \\ \min(\mathbf{tg}_{L_i} \cdot \mathbf{v}_{ij}, \mathbf{v}_{ij} \cdot \mathbf{tg}_{L_j}) & L_1 \text{ and } L_2 \text{ unreliable} \end{cases} \tag{6}$$

The last rule states that it is preferable to resort to bad estimates of the tangent vectors than to reject the possibility of a connection. The general idea is similar to that of Taniguchi et al. [5] with the addition of the 'reliability' qualifier.

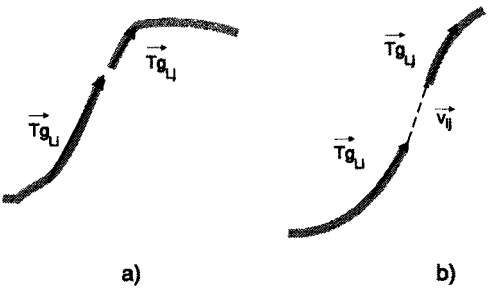

Fig. 2. Definition of the vectors

The confidence that a closed line might be a valid contour line is a function of the quality of connection along the line: for each connection, the ratio (smoothness of connection)/(best smoothness from the starting point) is computed, and

the confidence of the closed contour is the minimum ratio found. Connection trees can be reused for interactively extracting polylines from the image. When the user clicks on a line segment, it triggers a search in the corresponding connection tree and all segments logically attached to the selected line segment (in a 'smoothness' sense) are selected as well. This allows the user to add pieces of lines to the current contour model built with the closed lines already isolated.

6 Results

Experiments have been conducted on maps of varying quality. We show our results on two pieces of maps: one from the Salzburg area and one from the Montreal area. The maps are digitized at 600 dpi using an off-the-shelf true color (24-bit RGB) scanner. Figures 3 and 4 show the result of extracting the colors for the contour lines. The Montreal map (Fig. 4) has shown to be more difficult to handle because of the greater use of dithering in the map. The processing time for a 1000x900 image is 55 seconds on an SGI Indigo2 workstation.

Figure 5 shows the best contour lines (of confidence value over 0.90) found by the system. The other line segments are not displayed since the system does not have enough knowledge about them being parts of contour lines. Either more research is needed to improve the construction of the model by automatically merging these line segments into the solution, or some interface functionality is to be provided to the user for selecting polylines and merging them manually. Despite some verifications that no two closed contours are perfectly equal, it sometime happens that many instances of the same contour line are found, with each being slightly different from its 'siblings' (e.g. closed contour A has a one more small segment than closed contour B). Future work will provide a function that picks the best closed line among those that share the same basic segments. The line processing operation takes about 30 seconds where more than half of the time is spent on thinning.

7 Conclusions

This paper described our method for extracting contour lines from color images of scanned maps. The quality of line extraction depend mostly on the result of the color processing, which has been giving acceptable results in spite of the color noise generated by the scanner in the dithered images. We hope more tests will reveal the sensitivity of the quality of the segmentation to the saturation threshold (see Section 3.1). We expect low sensitivity when dealing with bright colors, whereas problems might arise when extracting low saturated colors (such as light yellow or dark brown).

Future work will involve integration of additional algorithms such as functions for line thickness measurement and verification of line inclusion. We plan to build a user interface that lets the user browse the list of found closed contours, edit them or aggregate line segments or polylines to the current set.

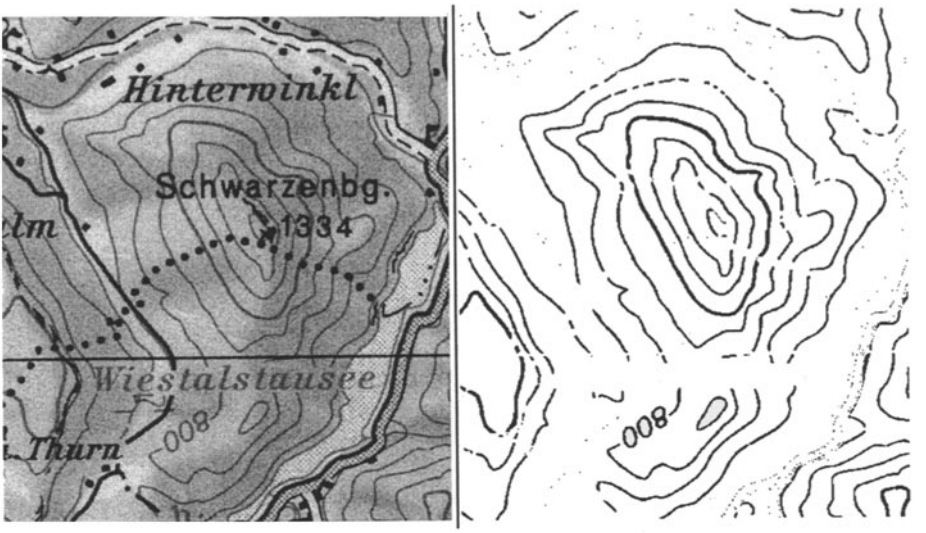

Fig. 3. The Salzburg map. Left: Original color image (printed in grey scale). Right: Result of the color segmentation; the user has selected the color *orange* for pixel classification.

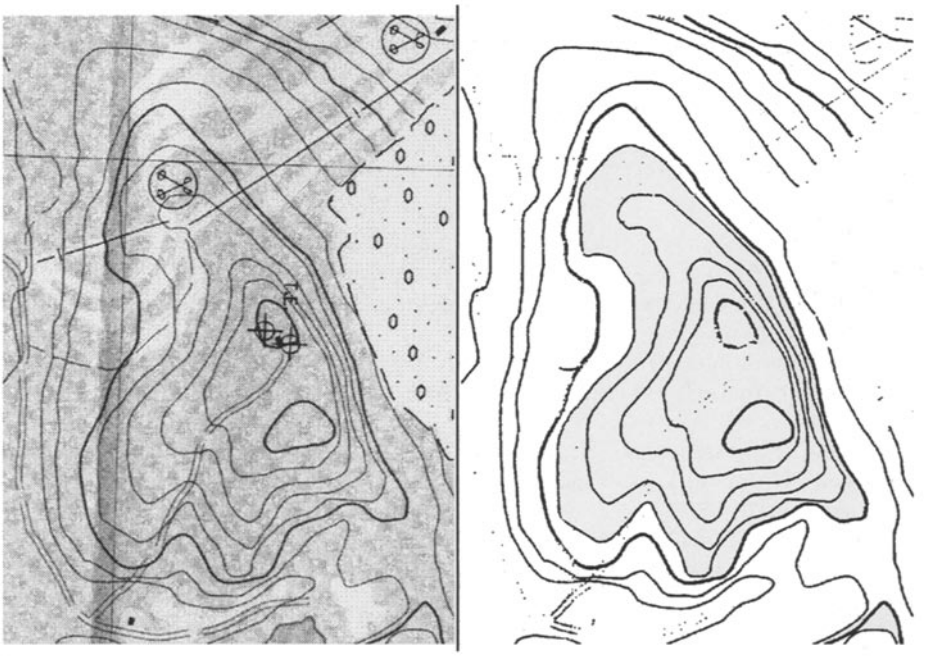

Fig. 4. The Montreal Map. Left: Original color image (printed in grey scale). Right: Result of the color segmentation; the user has selected the color *brown* for pixel classification.

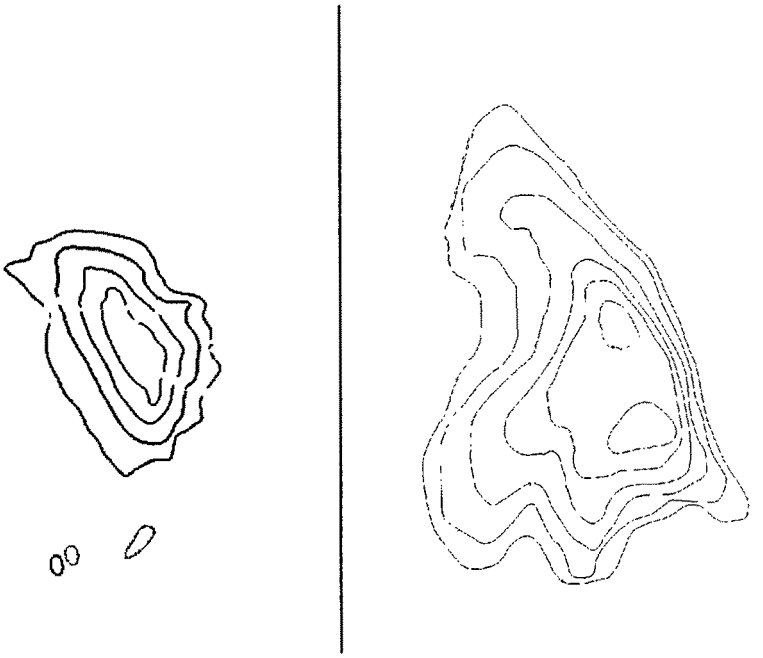

Fig. 5. Extracted closed contours for the Salzburg map (left) and the Montreal map (right).

References

1. Suzuki, S., Kosugi, M., Hoshino, T.: Automatic line drawing recognition of large-scale maps. Optical Engineering **26** (1987) 642–649
2. Suzuki, S., Yamada, T.: MARIS: map recognition input system. Pattern Recognition **23** (1990) 919-933
3. Ebi, N., Lauterbach, B., Anheier, W.: An image analysis system for automatic data acquisition from colored scanned maps. Machine Vision and Applications **7** (1994) 148–164
4. Shimada, S., Maruyama, K., Matsumoto, A.: Agent-based parallel recognition method of contour lines. Proc. of the 3rd Int. Conf. on Document Analysis and Recognition **1** (1995) 154–157
5. Taniguchi, R., Yokota, M., Kawaguchi, E., Tamati, T.: Knowledge-based picture understanding of weather charts. Pattern Recognition **17** (1984) 109–123
6. Pitas, I., Venetsanopoulos, A.N.: Nonlinear Digital Filters: Principles and Applications. Kluwer Academic Publishers, 1990
7. Tominaga, S.: A Color Classification Method for Color Images Using a Uniform Color Space. Proc. 10th Int. Conf. Patt. Recog. 1990 803–807
8. Guo, Z., Hall, R.W.: Parallel thinning with two-subiteration algorithms. Comm. of the ACM **32** (1989) 359–373