Contour Extraction and Visulization from Topographic Maps

Christopher Hansen
Department of Electrical Engineering
Stanford University
Stanford, CA
Email: cthansen@stanford.edu

***Abstract*—Recognition of features in topographic maps is useful for GIS applications. In particular, elevation data can be extracted and utilized by recognizing contour lines in maps. In this project, the viability of contour extraction using image segmentation and Delaunay triangulation for line reconstruction was investigated. Results suggest that, while the method has a high success rate for well behaved maps, it may be limited when applied widely.**

I. Introduction

Topographic maps offer a rich selection of data on the surface of the Earth, ranging from terrain type to land features, both natural and man-made. Perhaps one of the most ubiquitous features on topographic maps is the presence of contour lines to represent elevation data. These contour lines are often monochromatic brown and are closed with themselves or the edges of the map, by definition of being a contour.

However, as mentioned, contour maps include a great deal of non-contour information that often cross contour lines, making the task of recognition difficult. In the past, map providers such as the USGS have had to resort to manual means to extract map data [1]. Such methods are time consuming and costly, and therefore motivate the desire to extract contour lines and other map features automatically.

II. Algorithm

Investigation into related work shows that while there is a high level of agreement for some parts of the contour extraction process, such as image segmentation, approaches diverge when attempting to reconstruct contour lines from their recognized pieces [1], [2].

To motivate further discussion, each step shall be applied and discussed on the example image given in Figure 1 [2]. This map shall be defined as "well behaved": as can be observed, the contours are monochromatic and approximately brown. More importantly, each contour is distinct from all others; in other words, there is a sufficient gap between neighboring contours so that each contour can be recognized as being separate. As can be seen in the image, there are a number of features such as grid lines, rivers, and text that overlap the contours and would hinder extraction if it were attempted from the original image.

A. Segmentation

The first step in processing the map image is to isolate the contour lines from other map components. To accomplish this, one can leverage the previously mentioned monochromaticity of contour lines to segment the original image by color. As is suggested by most previous research [1], [2], the RGB color space is not ideal for this segmentation, due to being perceptually non-uniform. It is therefore proposed that the HSV color space be used during image segmentation because of its improved uniformity while being easily invertible to RGB [1].

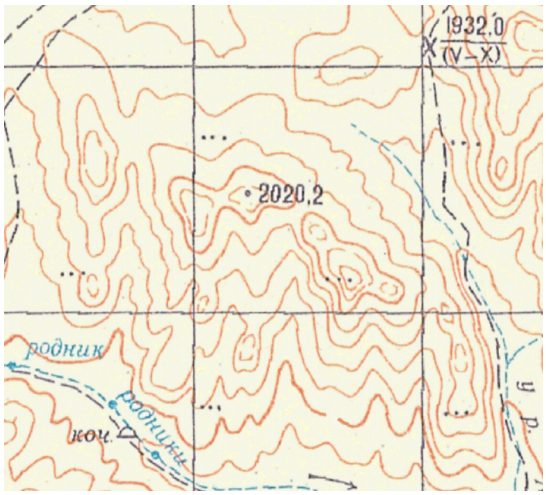

Fig. 1. Original map image

When considering segmentation, it is important to consider the other colors that may be encountered in the map. Probably the most dominant colors will be green for forested areas, blue for water, white for areas with no vegetation, black for grid lines and roads, brown for contour lines, and various other colors for other features. During segmentation it is important to ignore all these other colors. Salvatore et.al [2] proposes a heuristic for segmenting HSV images where colors with $value < 0.25$ can be classified as black and colors with $saturation < 0.20$ and $value > 0.60$ can be classified as white and can be ignored. Among the remaining results, colors with $0.1 < hue < 0.3$ can be classified as brown contours. Experimental results showed that this heuristic was decent, but results were able to improved by changing the black classification to $value < 0.5$ and the brown classification to $0.0 < hue < 0.2$. The results of applying this segmentation heuristic to the original image can be seen in Figure 2.

As can be observed, the image has segmented fairly well, such that it still has the "well behaved" behavior defined

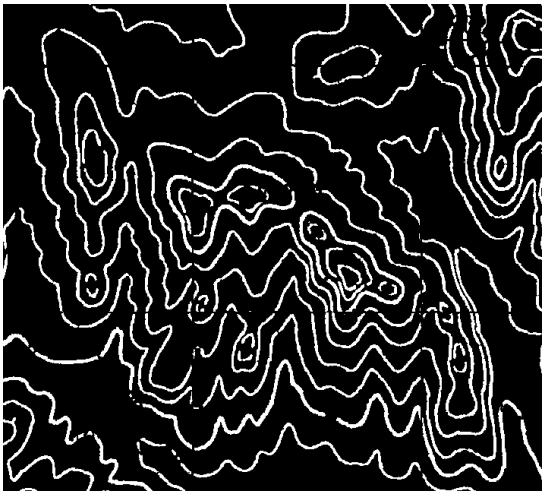

Fig. 2. Segmented image

previously. As expected, though, there are now breaks in the contours due to the other overlapping features that must be closed.

B. Thinning and Tracing

Given the segmented image, it is desired that the original set of potentially broken contours, which shall be referred to as sub-contours, be extracted. In order do so, something as simple as using the Moore contour tracing algorithm, as is suggested by Pradhen et.al [5], can be done. However, in order for Moore tracing to achieve good results for this application, it is desirable that each contour that is traced be only one pixel thick.

In order to reduce the segmentation results, which clearly have contours which are thicker than one pixel, to a single pixel, a thinning algorithm should be run on the segmented results. Research of existing literature [4] proposes that a morphological thinning operation would be sufficient to reduce the segmented image to a thinned image suitable for Moore contour tracing.

The Moore tracing algorithm is implemented uses a clockwise search pattern and has the termination condition of revisiting the starting pixel twice. In addition to extracting the pixels that fall on a contour, it is desired to also find the endpoints of the contour if it is not closed with itself. Knowledge of endpoint locations is necessary for future steps in the algorithm, and they are most easily and accurately computed in parallel with the Moore tracing algorithm. A heuristic definition for an endpoint is any pixel on the contour that is visited half as often as the interior pixels. Experimentation with this heuristic shows that not only does it extract the expected endpoints, but it also identifies branches on contours as endpoints. These branches, an example of which can be seen in Figure 3), are often artifacts from image segmentation and thinning. It is problematic if extraneous endpoints are detected since they will introduce false data used during line reconstruction, so a secondary heuristic is necessary to ensure that branch endpoints are removed. The implemented heuristic uses the fact that branches are often only a few pixels in length and occur towards the middle of contours to make the assumption that the set of two possible endpoints that are the farthest

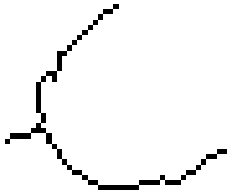

Fig. 3. Branched contour

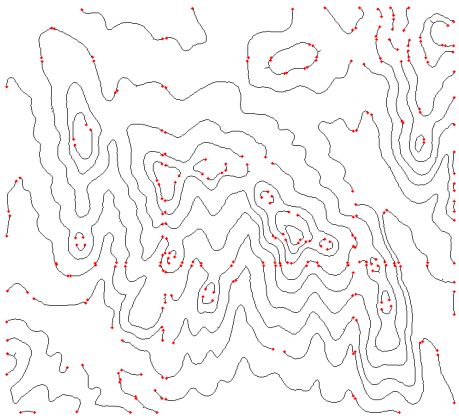

Fig. 4. Sub-contour extraction

apart from one another (in terms of contour pixels) are the two actual endpoints for a sub-contour. Experimentation with this heuristic showed that it produced correct results in nearly every "well behaved" map. Running the segmented image through the thinning, Moore contour tracing, and endpoint finding algorithms yields each contour in a vectorized form which can then be processed further. As can be seen in Figure 4, which shows the results of these steps on the previously shown segmented image, the results from this step are quite good.

C. Curve Reconstruction

As was previously observed, as a result of segmentation, there are many gaps and broken contours. The process of thinning and tracing reduces the segmented contours to vectorized form, but the gaps still remain. Therefore, the key aspect of correct contour extraction is to have have a robust method of coalescing broken contours back into closed contours.

1) Delaunay Triangulation: Research conducted by Amenta et.al [3] proposes a method for curve reconstruction. To summarize their work, they propose that the Voronoi vertices from a sufficiently sampled set of input curves approximate the medial axis of the sample points. The Voronoi vertices, along with the curve's vertices, can then be used to construct a Constrained Delaunay Triangulation. The observation was that any Delaunay edge from the triangulation that connects a pair of original points can be surmised to be part of a contour.

Therefore, the proposed algorithm is to take the set of all

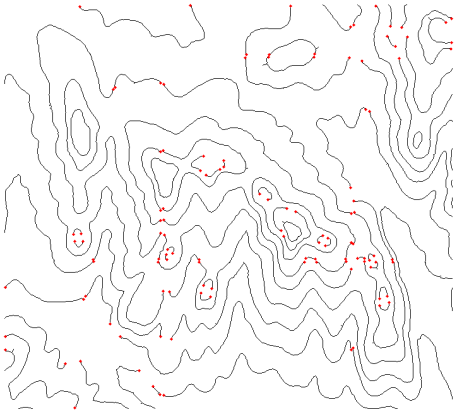

Fig. 5. One iteration of Amenta's algorithm

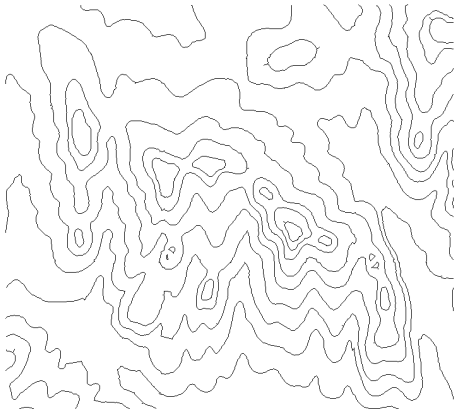

Fig. 6. Post reconstruction

pixels from all curves found during tracing and find the set of Voronoi vertices. The union of these sets of pixels is used in a Constrained Delaunay Triangulation. Based on the resulting Delaunay edges, the algorithm searches for any edge where both the start and end pixels are in the set of endpoints, also gathered during tracing. For any such edge, the set of pixels and endpoints belonging to each sub-contour connected are merged, along with the pixels that fall on the line connecting the two endpoints. The connected endpoints are then removed from the set of endpoints for the new sub-contour. The results of running this algorithm once can be seen in figure 5.

After running this process it can be observed that, while some sub-contours were merged and closed, others were not since no Delaunay edge fell between endpoints. As observed by Salvatore et.al [2], if contours that have been closed are removed and only the unclosed sub-contours are considered, the Delaunay triangulation is simplified. Using this fact, an iterative approach is proposed where the algorithm described by Amenta is run repeatedly, where each closed contour is removed from the set of contours processed by Amenta's algorithm. This is run until the set of sub-contours "stabilizes", in other words no other contours are being closed using Amenta's algorithm.

One important consideration for this iterative algorithm is how to recognize a contour as being closed. A simple method that holds as long as endpoints are correctly identified is to say a contour is closed if and only if the set of endpoints associated with the contour is empty. Since the endpoint extraction method discussed previously proved to be robust, this method for detecting closed contours worked well as well.

2) Euclidean Closing: Since Delaunay Triangulation may not close all sub-contours, it is necessary to have a fall-back algorithm such that all contours can be closed. It is proposed that connecting sub-contours that have endpoints the lowest Euclidean distance apart is a valid approach for the alternate closing algorithm. While experimentation with Euclidean closing on sub-contours prior to Delaunay Triangulation yielded many false connections, due to the close proximity of contours, Delaunay Triangulation successfully closes enough contours that the failure rate of Euclidean closing is substantially lower. Furthermore, when finding endpoints to close using Euclidean closing, both endpoints must agree that the other endpoint is the lowest Euclidean distance away in order for closing to occur.

A special case of Euclidean closing is a contour with the edge of the image. In order to do this, four artificial endpoints are considered when finding the the endpoint with the minimum Euclidean distance. These four endpoints fall on each edge such that the distance of the endpoint being considered to each edge is minimized. With the addition of these endpoints, a sub-contour will now close with an edge if it is closer to an edge than any other endpoint.

While this method works fairly well, it proves to be problematic in the case of a broken contour that falls very close to the edge, as the sub-contour will close with the edge rather than the other sub-contour. A heuristic to attempt to fix this is to multiply each edge distance by a certain constant, making them less likely to be chosen in this edge case. While the heuristic helps, it clearly does not completely remove the problem, and may actually cause additional problems of sub-contours not closing with the edge when they should.

It is the intent of the Euclidean closing algorithm to be run once each time the iterative Amenta's algorithm stabilizes, so that further progress can be made in closing the contours. The steps of running Amenta's algorithm until it stabilizes followed by one iteration of Euclidean closing is repeated until all sub-contours are closed.

III. Results

The results of running the delineated algorithm on the original image shown in Figure 1 can be seen in Figure 6. As can be observed, all contours have been closed, with a high degree of accuracy to the original image. However, there are a number of invalid closings which occur as an artifact of using Euclidean closing. These happen because, as can be seen in the example in Figure 7, the distance between two endpoints on adjacent contours is less than the distance between two endpoints on the broken contour.

Unfortunately, no formal test metric was able to be designed due to time constraints. However, a proposed metric could be randomly generating input images, artificially inserting gaps in these images, executing the algorithm to extract the vectorized contours, and computing the percent of pixels that

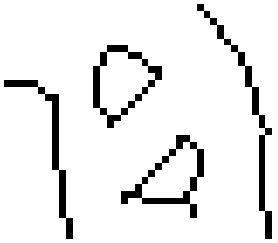

Fig. 7. Invalid closing

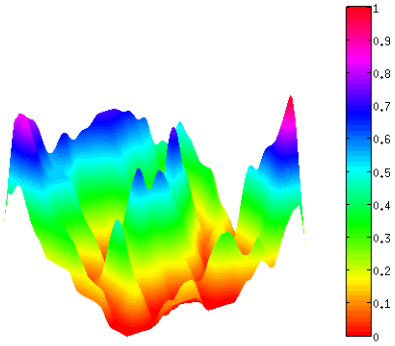

Fig. 8. Surface model

agree between the output and input images. Despite the lack of a formal metric, a fair amount of testing was performed on images acquired from the Internet that found the algorithm to perform fairly similarly as to the example traced through this paper, given that the image was well behaved.

Once result contours are extracted and reconstructed from a map, the data can be used for any application desired. As a proof of concept, a three dimensional surface model was created and can be seen in Figure 8.

IV. Future Work

Though the algorithm presented achieves good results, the key aspect that has been reiterated throughout this article is that the image must be "well behaved", in other words must conform to an expected color for the contours and have the contours sufficiently spaced. Additional changes to the algorithm can be made as well to improve performance, as shall be further discussed.

A. Variable Contour Color

As mentioned, currently contours must fall within a color range in order to be recognized, which presents issues if a map being processed does not have contour of that color. An improvement for the algorithm would be to dynamically estimate the contour color and segment based on this estimation. San et.al [1] propose a technique for contour color estimation by inferring the background color and utilizing histograms of the hue channel in order to improve segmentation.

B. Closely Spaced Contours

In images where neighboring contours are to closely spaced, an example of which can be seen in Figure 9, segmentation of the image results in the coalescing of neighboring contours, which is causes the current implementation of contour tracing to fail to recognize each contour as distinct. There is no clear solution to this issue at this time, but it is worthy of further research and consideration.

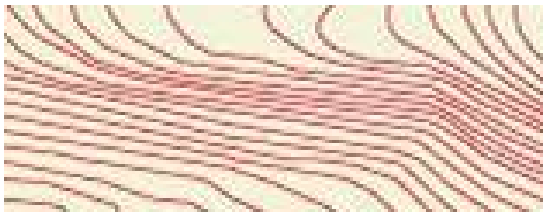

Fig. 9. Closely space contours

C. Improve Euclidean Closing

As was seen previously, Euclidean closing the the main point of failure for the application, and it is therefore worth finding an alternate secondary contour closing method. Basic consideration suggests that the direction of a sub-contour prior to its endpoint should be considered when closing. Salvatore et.al [2] proposes further utilizing Voronoi vertices by considering the Voronoi vertices adjacent to each endpoint when choosing a endpoint to close with.

D. Text and Other Feature Recognition

Often, having the contours is not enough; rather it desired that each contour is labeled with its height. Therefore, it is proposed that map images be passed through a preprocessor that performs OCR operations to extract height labels, and then these labels will be assigned to contours postprocessing based on their original locations. While conceptually straight forward, there is no doubt a great deal of work in accurately performing OCR on the original images.

Acknowledgment

The author would like to thank Professors Bernd Girod and Gordon Wetzstein, as well as the whole EE368 teaching for creating and teaching such a quality course.

References

[1] San, L.M.; Yatim, S.M.; Sheriff, N.A.M.; Isrozaidi, N., "Extracting contour lines from scanned topographic maps," Computer Graphics, Imaging and Visualization, 2004. CGIV 2004. Proceedings. International Conference on , vol., no., pp.187,192, 26-29 July 2004

[2] Salvatore, Spinello; Guitton, Pascal, "Contour line recognition from scanned topographic maps," Journal of WSCG 12.1-3 2004.

[3] Amenta, Nina; Bern, Marshall; Eppstein, David, "The Crust and the -Skeleton: Combinatorial Curve Reconstruction," Graphical Models and Image Processing, Volume 60, Issue 2, March 1998, Pages 125-135, ISSN 1077-3169

[4] Jang, B.-K.; Chin, R.T., "Analysis of thinning algorithms using mathematical morphology," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.12, no.6, pp.541,551, Jun 1990

[5] Pradhan, Ratika; Shikhar Kumar; Ruchika Agarwal; Mohan P. Pradhan, Ghose M. K., "Contour line tracing algorithm for digital topographic maps." International Journal of Image Processing (IJIP) 4, no. 2 (2010): 156-163