Automatic Text Recognition from Raster Maps

Yao-Yi Chiang and Craig A. Knoblock
University of Southern California
Department of Computer Science and Information Sciences Institute
4676 Admiralty Way, Marina del Rey, CA 90292
yaoyichi, knoblock@isi.edu

Abstract

Text labels in raster maps provide valuable geospatial information by associating geospatial locations with geographical names. Although present commercial optical character recognition (OCR) products can achieve a high recognition rate on documents, text recognition on raster maps is still challenging due to the varying text orientations and the overlapping between text labels. This paper presents an automatic text recognition approach which focuses on locating individual text labels in the map and detecting their orientations and leverages horizontal text recognition of commercial OCR software. We show that our approach detects all strings in the test maps and achieves 96.8% precision and 95.7% recall on character recognition.

1. Introduction

Maps are easily accessible compared to other geospatial data, such as vector data, databases of geographical names, geospatial information systems (GIS), etc. Due to the popularity of high quality scanners and the Internet, we can now obtain various maps in raster format for areas around the globe. By converting the text labels in a raster map to machine-editable text, we can produce geospatial knowledge for understanding the map covering area where other geospatial data is not ready accessible. Moreover, we can register other geospatial data (e.g., imagery) to a raster map [3] and exploit the recognized text from the map for indexing and retrieval of the geospatial data.

Text recognition from raster maps is a challenging task. First, the image quality of the raster maps usually suffers from compression and scanning noise. Second, the text labels can have various font types and font sizes and very often overlap with each other or even other features in the maps, such as road lines. Third, classic OCR research focuses on documents containing text lines in the same direction (usually horizontal text lines); however, the text labels within a map do not follow a fixed orientation.

In this paper, we present a general approach to overcome these difficulties for recognizing text labels from raster maps. We first quantize color space of the raster map and generate a color palette for extracting the pixels of text labels (i.e., the text layer) using user specified colors. Then, we perform connected-component analysis on the text layer to identify characters, group characters into strings, and split overlapping text labels into individual strings. Finally, we detect the orientation of the multi-oriented strings and rotate the image of each individual strings to the horizontal direction. The string images of horizontal direction then can be processed using commercial software for recognizing the text. We tested on two maps with 1,655 characters and 308 words of varying text fonts and sizes and show that we can achieve accurate recognition rates automatically.

The remainder of this paper is organized as follows. Section 2 discusses related work on text recognition from raster maps. Section 3 presents our approach to detect and prepare string labels for commercial OCR software. Section 4 reports on our experimental results and Section 6 presents the discussion and future work.

2. Related Work

Text recognition from raster maps has been an active research area. One type of research separates from classic OCR research (i.e., working on text with the same orientation) and builds specific character recognition components for handling multi-oriented text labels. Both Deseilligny et al. [4] and Adam et al. [1] uses rotation-invariation features to compare the target string with trained character samples for recognizing text labels from raster maps. These methods require intensive training, such as providing sample characters for each test map.

For the techniques that employ classic OCR method as their character recognition components, Li et al. [6] identify the graphics layer and text labels using connected-component analysis and extrapolate the graphics layer to remove the lines that overlap with characters within each identified text label. Then, a template-matching based OCR component is used to recognize characters from the text labels. Cao and Tan [2] also analyze the geometry properties of the connected component to first separate text labels from graphics. The separated graphics layer is then decomposed into line segments and a size filter is used to recover the character strokes that touch the lines. Finally, an OCR software from HP is then used to recognize the text labels. In both [6] and [2], the identified text labels are manually rotated to the horizontal direction for the final character recognition tasks.

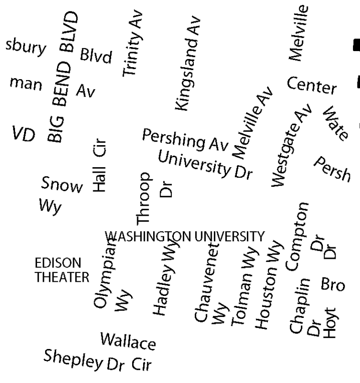
Pouderoux et al. [8] use component analysis and string analysis with dynamic parameters generated from the geometry of the connected components to identify strings in the raster maps. The strings are then rendered horizontally for character recognition using the average angle connecting the centroids of the components in a string. However, the average angle can vary much when the characters have very different height or width. For example, considering the substring ‘afa’ from one of our test maps, the angle of the line connecting the centroid of the first ‘a’ and the centroid of ‘f’ is almost perpendicular to the line connecting ‘f’ and the second ‘a’. On the other hand, we adopt the skew detection method in [7] to identify the orientation of each string automatically.

3. Text Labels Recognition

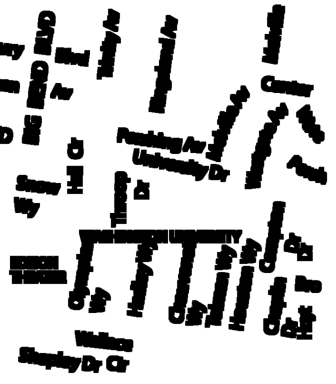
This section describes our techniques to locate groups of text labels in the map, separate overlapping labels, and detect the text orientation of each label.

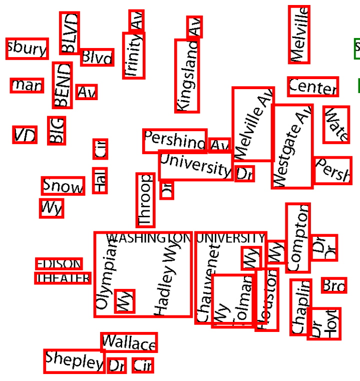
3.1 Extracting Text Pixels

Raster maps usually contain numerous colors due to the scanning or compression processes. To generate a color a color palette for extracting the text pixels using user specified colors, we apply color segmentation techniques to reduce the number of colors in the maps. We first apply the Mean-Shift filtering algorithm, which merges two colors into one by considering the distance in the color space and the image space. The Mean-Shift filtering algorithm preserves the object edges in the map and prevents pixels of two different objects to have the same color. Next, we utilize a common color quantization method called the median-cut [5] to generate a image with maximum 1,024 colors. The quantized image is present to the user for selecting a set of colors that represents text in the map; however, if the same color



(a) An example text layer


(b) Dilated text layer


(c) Identified strings

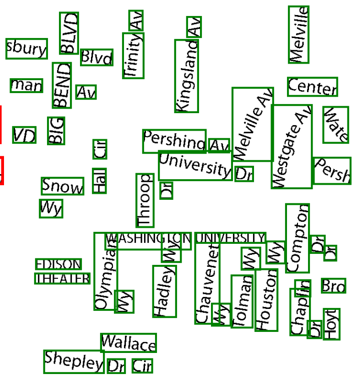

(d) Split strings

Figure 1. Locating strings in the map

is used on both text and other features, text separation techniques such as [2] can be used to remove graphics and preserve text pixels. Figure 1(a) shows an example of the extracted text layer.

3.2 Identifying Strings

With the extracted text layer, the user provides a sample string for each of the font size used in the text layer as follows: the user select a bounding box of a horizontal string and indicate how many characters are in the string. We then compute the character width and character spacing of each font size using the width of the bounding box and the number of characters. If more than one font size is used in the text layer, we separate the text layer into sub-text layers by performing a connected-component analysis with a size threshold on every connected component. Therefore, every sub-text layer should contain characters with similar size.

To group characters in each sub-text layer into string, we use the dilation operator as in [2] to merge nearby characters. The iteration of the dilation operator is determined by the character spacing. Figure 1(b) shows the results of merged characters and Figure 1(c) shows the identified strings in red rectangles. As note in [2], the dilation operator has the benefit on identifying curved strings, but it also merged two nearby strings.

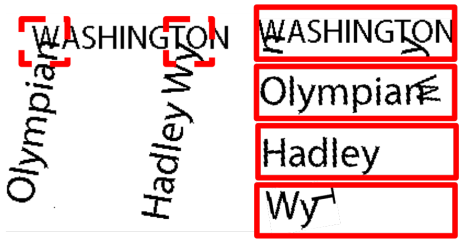


Figure 2. Splitting merged strings

To separate merge strings, we first perform a connected-component analysis within each of the merged strings and use the distance transformation to calculate the pixel distance between each connected component in a merged string. Two connected components are linked if the distance in pixel between them is smaller than a threshold, which is also determined by the character spacing. Next, we start to trace the connected component following the links in each merged string and identify individual strings. Figure 2 shows a merged string, where ‘W’ overlaps ‘n’ and ‘T’ overlaps with ‘y’. To split the merged string, we identify two types of connected component as splitting points: First, the ones that have more than three links, such as ‘Ty’. Second, the ones those constitute an angle smaller than a threshold with their neighbors, such as ‘Wn’ and its neighbors ‘a’ and ‘A’. The angle between three connected component are calculated using the centroid of each connected component. In the case that three connected components of the same height constituting a straight label, the angle is 180 degree. However, since the connected components have various heights, we use an angle threshold of 145 degree to prevent breaking a continuous string. For the maps with curvy labels, we use an angle threshold of 125 degree to preserve curvy labels. After we identify the splitting points, we can produce individual strings as shown in the right side of Figure 2.

3.3 Detecting String Orientation

Skew correction is very well developed in modern OCR techniques; however, classic skew correction can only apply on documents with multiple lines since the space between lines are exploited to detect the tilt angle [7]. To detect the orientation of each string, we modify the morphological based skew correction method for multi-line document in [7] by select different sizes of structure elements for each string image. We first apply the closing operator using a structure element of size equal to the character width plus character spacing. Then, we rotate the string image from 0 degree to 179 degree and we use the maximum horizontal width of the rotated string to determine the length of the structure elements of the erosion operator. Therefore, the erosion operator is able to erase a portion of the foreground pixels when the string is not in the horizontal direction while not overly elongate to erase every foreground pixels in every rotated image. Figure 3 shows examples of intermediate results of the orientation detection where the horizontal strings has the most remaining pixels after we apply the erosion operator.

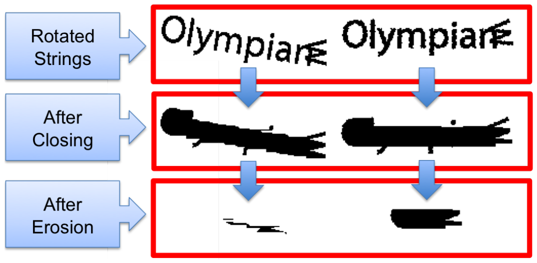


Figure 3. Detecting orientation using morphological operators

We only apply the orientation detection technique on the strings with more than three connected components since the orientation of short strings can be dominated by the height of the components. To assign orientation for the short strings, we search from the centroid of a short string for nearby strings and use the orientation of the nearby strings as the short string’s orientation. This is because short strings in a raster map are usually part of a longer label. For example, the most common short strings in our test maps are ‘Av’ as avenue, ‘Dr’ as drive, ’Cir’ as circle, which are all part of a road name.

4. Experimental Setup and Results

We tested our approach on maps from two sources. The first test map is a digital map (850x850 pixels each) from Rand McNally (RM map). In addition to the digital map, we tested our technique on processing commonly accessible scanned maps using a map tile (2750x2372 pixels) cropped from a scanned maps (350dpi) published from by International Travel Maps (ITM map). We applied our techniques in this paper to identify text labels from the test maps and detect the orientation of the labels. We generate two images for one text label by rotating the text label clockwise and counterclockwise to the horizontal direction according to its orientation. Then, we selected the correctly rotated string image (i.e., not the upside down one) for the character recognition task, for which we used a commercial OCR software called ABBYY FineReader 10.

For the RM map, we identified 54 strings and 29 of them have more than 3 characters and hence are sent to detect their orientation. After manual verification, we detected accurate orientation for 28 of the 29 strings and the orientation offset for the only inaccurate string are 2 degrees. For the ITM map, we identified 254

Table 1. Character recognition results

Map	# of Char.	Precision	Recall
RM map	258	96.4%	94.9%
ITM map	1,397	96.9%	95.7%

strings and 196 of them are sent to detect their orientation. After manual verification, we detected accurate orientation for 183 of the 197 strings and the average orientation offset for the 5 inaccurate strings are only 5.4 degrees. The orientation offset came from shorter strings or string with symbols such as a quotation mark.

There are 25 strings and 58 strings in the RM map and the ITM map, respectively, have less or equal to 3 characters and hence we search nearby strings to assign their orientation. Among the 25 strings in the RM map, 5 of them we cannot find a nearby string to inherit the correct orientation. This is because four of the strings are near the boarder of the map and hence the short strings do not follow any of the nearby string orientation in the map. The other one of the incorrect orientation is the string 'Av' shown in the left-upper part of Figure 1(a), where a road name, 'BEND' is in between the string 'Av' and its counter part 'man'. Among the 58 short strings in the ITM map, 6 of them we cannot find a nearby string to inherit the correct orientation. This is because three of the short strings are near the boarder of the map and the other three are isolated characters.

Table 1 shows the OCR results on the character level. The errors were resulting from: 1. We do not have the orientation of the strings. 2. Overlapping characters, such as the 'n ' and 'w' shown in Figure 2. The problem of missing orientation detection can be done by performing OCR on multiple degrees to recognize the characters since there are only a few of strings we do not detect correct orientation. For the overlapping text, additional knowledge such as geographical name database could help as a dictionary to improve the results. For the string level accuracy, we extracted every string from both maps. 43 of 54 extracted strings in the RM map (80%) have all of their characters successfully recognized. 218 of 254 extracted strings in the ITM map (86%) have all of their characters successfully recognized.

5. Discussion and Future Work

In this paper, we present an approach to automatically recognize text labels from raster maps. Our approach focuses on locating individual strings and detecting string orientation while leverages the advance of horizontal text recognition of commercial OCR software. By doing so, our approach benefits from future improvement on commercial OCR software. Our experiments show accurate results on detecting the orientation of the identified text labels and recognizing the text labels. In the future, we plan to include additional knowledge of the map covering area to build a database of geographic names and use the database as a dictionary for improving the OCR accuracy on overlapping characters.

6 Acknowledgments

The author would like to thank Dr. Chia-Hsiang Yang for his input on the article. This research is based upon work supported in part by the University of Southern California under the Viterbi School Doctoral Fellowship.

References

[1] S. Adam, J. Ogier, C. Cariou, R. Mullot, J. Labiche, and J. Gardes. Symbol and character recognition: application to engineering drawings. *IJDAR*, 3(2): 89–101, 2000.

[2] R. Cao and C. L. Tan. Text/graphics separation in maps. In *Proceedings of the 4th GREC Workshop*, pages 167–177, 2002. ISBN 3-540-44066-6.

[3] C.-C. Chen, C. A. Knoblock, and C. Shahabi. Automatically and accurately conflating raster maps with orthoimagery. *GeoInformatica*, 12(3):377–410, 2008.

[4] M. P. Deseilligny, H. L. Mena, and G. Stamonb. Character string recognition on maps, a rotation-invariant recognition method. *Pattern Recognition Letters*, 16(12):1297–1310, 1995.

[5] P. Heckbert. Color image quantization for frame buffer display. *SIGGRAPH*, 16(3):297–307, 1982.

[6] L. Li, G. Nagy, A. Samal, S. C. Seth, and Y. Xu. Integrated text and line-art extraction from a topographic map. *IJDAR*, 2(4):177–185, 2000.

[7] L. Najman. Using mathematical morphology for document skew estimation. *SPIE DRR IX*, pages 182–191, 2004.

[8] J. Pouderoux, J. C. Gonzato, A. Pereira, and P. Guitton. Toponym recognition in scanned color topographic maps. In *Proceedings of the 9th ICDAR*, volume 1, pages 531–535, 2007.