An Approach for Recognizing Text Labels in 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 geographical names with geospatial locations. Although present commercial optical character recognition (OCR) products can achieve a high recognition rate on documents containing text lines of the same orientation, text recognition on raster maps is challenging due to the varying text orientations and the overlap of text labels. This paper presents a text recognition approach that focuses on locating individual text labels in the map and detecting their orientations to then leverage the horizontal text recognition capability of commercial OCR software. We show that our approach detects accurate string orientations and achieves 96.2% precision and 94.7% recall on character recognition and 80.6% precision and 84.1% recall on word recognition.

1. Introduction

Maps are easily accessible compared to other geospatial data, such as vector data, satellite imagery, gazetteers, etc. Due to the availability 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 region while other geospatial data are not ready available. Moreover, we can register a raster map to other geospatial data (e.g., imagery) [3] and exploit the recognized text from the map for indexing and retrieval of the other geospatial data.

Text recognition from raster maps is a challenging task. First, the image quality of the raster maps usually suffers from the scanning and/or image compression processes. Second, the text labels in a raster map can have various font types and sizes and very often overlap with each other or with other features in the map, such as roads. Third, 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 in raster maps. We first quantize the raster map to generate a color palette with a limited number of colors, and we use user-specified colors from the color palette to extract the pixels of text labels (i.e., the text layer). Then, we perform the connected-component analysis on the text layer to identify characters, group characters into strings, and split overlapping strings. For the identified strings, we detect their orientations and rotate individual string to the horizontal direction. Finally, we process the horizontal strings using commercial software for recognizing the characters. We tested our approach on two maps of varying text fonts and sizes and show that we can achieve accurate recognition rates.

The remainder of this paper is organized as follows. Section 2 discusses related work on text recognition from raster maps. Section 3, 4, and 5 present our approach to detect and prepare string labels for commercial OCR software. Section 6 reports our experimental results, and Section 7 presents the discussion and future work.

2. Related Work

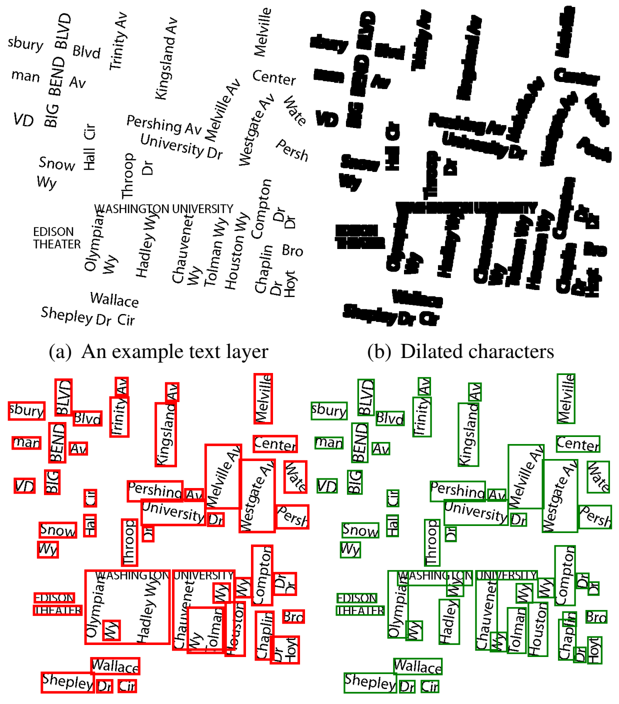
Text recognition from raster maps has been an active research area. One type of research builds specific character recognition components for handling multi-oriented text labels [1, 5], which differs from classic OCR research that assumes the documents containing text lines all in a single orientation. Deseilligny et al. [5] use rotation-invariant features and Adam et al. [1] use image features based on the Fourier-Mellin Transformation to compare the target characters with the trained character samples for recognizing text labels in maps. These methods require intensive training, such as providing sample characters for maps using different fonts to generate distinct feature sets for the classification.

For the techniques that employ classic OCR methods as their character recognition components, Li et al. [7] identify the graphics layer and text labels using the connected-component analysis and extrapolate the graphics layer to remove the lines that overlap with characters. Then, a template-matching-based OCR component is used to recognize characters from the text labels. Cao and Tan [2] analyze the geometry properties of the connected components 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 application from HP is used to recognize the text labels. In both [7] and [2], the identified text labels are manually rotated to the horizontal direction for the final character recognition task.

Pouderoux et al. [9] use dynamic parameters generated from the geometry of the connected components to identify strings in raster maps; however, they do not consider overlapping labels, which commonly exist in maps. The identified strings are then rendered horizontally for character recognition using the average angle connecting the centroid points of the components in a string, which can be inaccurate when the characters have very different heights or widths. For example, consider 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’. In our approach, we use a robust skew detection method derived from a morphological-operator based skew detection technique [8] to identify the orientation of each string automatically.

3. Extracting Text Pixels

Raster maps usually contain numerous colors due to the scanning or compression processes. To extract the text pixels, we apply color segmentation techniques to reduce the number of colors in the maps for generating a color palette with a limited number of colors. We first utilize the Mean-shift filtering algorithm [4] to smooth the image and reduce noise. The Mean-shift filtering algorithm merges two colors into one by considering their distance in the color and image spaces, which preserves the object edges while performing the segmentation. Next, we utilize a color quantization method called the Median-cut [6] to generate a image with at most 1,024 colors. We present the quantized image to the user for selecting a set of colors that represents text in the map; however, if a set of colors is used on both text and other features, text separation techniques, such as the one developed by Cao and Tan [2] or the connected-component analysis can be used to remove graphics. Figure 1(a) shows an example of the extracted text layer.


(a) An example text layer (b) Dilated characters

(c) String groups (d) Individual strings

Figure 1. Locating strings in the map

4. Identifying Strings

With the extracted text layer, the user provides a sample string for each of the font sizes used in the text layer and indicates how many characters are in the sample string. We then compute the character width and character spacing for each font size using the sample strings. If the text layer has more than one font size, we separate the text layer into sub-layers by performing the connected-component analysis with a size threshold on every connected component. Therefore, every sub-layer contains characters of a similar size.

To group characters in the text layer into strings, we use the dilation operator to merge nearby characters. We determine the iteration of the dilation operator using the character spacing. Figure 1(b) shows the results of the merged characters and Figure 1(c) shows the identified strings using the rectangles. The dilation operator has the advantage that it can group characters in curved strings, but it can also merge nearby strings [2].

Figure 2 shows a merged string, where ‘W’ overlaps with ‘n’ and ‘T’ overlaps with ‘y’. To separate the merged string, we first use the distance transformation to calculate the pixel distance between each connected component in the string. Two connected components are linked if the distance in pixels between them is smaller than a threshold, which is also determined by the character spacing. Next, we start to trace the connected components following the links in each merged

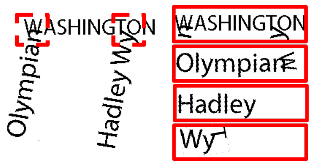


Figure 2. Splitting merged strings

string and identify individual strings. Finally, we identify two types of connected components as the splitting points: The first type is the connected components that have more than three links, such as 'Ty'. The second type is the connected components that constitute an angle smaller than a threshold with the two neighbors, such as 'Wn' and its neighbors 'a' and 'A'. We calculate the angle between three connected components using the centroid of each connected component. In the case where three connected components of the same height constitute a straight string, the angle is 180 degrees. However, since the connected components have various heights, we use an angle threshold of 145 degrees to prevent breaking a continuous string. For the maps with curved labels, we use an angle threshold of 125 degrees to preserve them. After we identify the splitting points, we can produce individual strings as shown on the right side of Figure 2.

5. Detecting String Orientation

Skew correction is well developed in modern OCR techniques; however, classic skew correction can only be applied to multi-line documents since the line spacing is exploited to detect the tilt angle, such as the morphological-operator based method [8]. We modify the morphological-operator based skew correction to detect the orientation for a single string by automatically selecting different sizes of structure elements of the morphological operators (i.e., the closing and erosion operators) for each string image.

We first rotate the string image by 0 degree to 179 degrees and apply the closing operator on the rotated images using a structure element of height equal to one pixel and width equal to the character width plus character spacing to grow string blobs, as shown in the middle row in Figure 3. Then, we use the maximum horizontal width among the rotated strings to determine the width of the structure element of the erosion operator.

We identify the horizontal string among the rotated strings using the number of remaining pixels after we apply the erosion operator. The bottom row in Figure 3 shows example results after we apply the erosion operator where the horizontal string has more remaining pixels than the tilted string. If we use a fixed erosion operator on all strings, a larger-sized erosion operator eliminates all foreground pixels of shorter strings even if the strings are in the horizontal direction. If we use

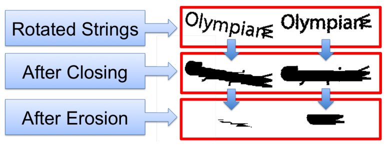


Figure 3. Detecting string orientation using morphological operators

a smaller-sized erosion operator, it can only eliminate a small percentage of the foreground pixels so that after the erosion the horizontal string might not be the string with the most remaining pixels among the rotated strings.

We only apply the morphological-operator based orientation detection on the strings having more than three connected components. This is because the detected orientation of a short string can be dominated by the heights of the short strings connected components using our method. Since short strings in a raster map are usually part of a longer label, we search from the centroid of a short string for nearby strings and use the orientations of the nearby strings as the short string's possible orientations. For example, the most common short strings in our test maps are 'Av' as avenue, 'Dr' as drive, and 'Cir' as circle, which are all part of a road name. We used a dynamic distance-threshold derived from the size of the bounding box of each string to limit the search space.

Once we have the string orientations, for each string, we rotate the string clockwise and counterclockwise to the horizontal direction according to its possible orientations (short strings might have more than one detected orientation depending on the number of its neighboring strings) to generate a set of rotated strings. Then, to identify the correctly oriented horizontal string (i.e., not the upside-down one), we send all rotated strings to a commercial OCR product called ABBYY FineReader 10 and automatically select the rotated string with the highest returned recognition confidence.

6. Experimental Setup and Results

We tested our approach on maps from two sources. One test map is a digital map (850x850 pixels) published by Rand McNally (RM map) covering St. Louis, MO. A second test map is a map tile (2750x2372 pixels) cropped from a scanned map (350 dot-per-inch) published by International Travel Maps (ITM map) covering Baghdad, Iraq. The two maps contain a total of 1,656 characters and 296 words (four words are curved strings).

We first detected strings and their orientation(s) (a detected string is a group of characters, which can be a sub-string of a word). We detected 308 strings from the two maps, and we applied our orientation detection on 225 of the detected strings that have more than three connected components. After manual verification, 219 strings have their detected orientations with a 0 degree offset to the ground truth and the average orientation offset for the six inaccurate strings is 4.8 degrees. The inaccurate orientations come from the shorter strings that have at least one character having very different size than the others, such as 'Talaa'. For the 83 strings that have fewer than four connected components, we searched nearby strings to assign their orientations, and eight of them we could not find a nearby string for the given search threshold. This is because five of the eight strings are near the boarders of the maps and the other three are isolated characters. For the short strings where we found at least one nearby string (the number of nearby strings we found ranges from one to three), three strings do not have the correct orientations passed from their nearby strings. This is because the short strings are near the boarders of the maps and hence the short strings do not follow any of the string orientations near them.

Table 1 shows the OCR results. A portion of the unrecognized characters is due to the fact that we did not find the orientations for some of the short strings, so they were not sent to the character recognition task. The problem of the missing orientations can be resolved by performing OCR on multiple degrees to recognize the characters since there are only a few strings where we did not detect the correct orientations. The other part of the unrecognized characters and false-positives come from some of the overlapping characters, such as the 'n ' and 'w' shown in Figure 2.

For the word level accuracy, we successfully recognized 249 words among 296 words in the two maps. For 37 of the words, we did not successfully recognize all of their characters. In addition, we mis-split 10 words into 21 substrings. The problem of the mis-split words is due to the fact that the neighboring characters in these words have very different sizes and some characters can be misidentified as the splitting points. For example, we mis-split the word 'Haifa' into 'Haif' and 'fa' since 'f' was misidentified as a splitting point. In the cases of the overlapping characters and mis-split words, additional knowledge such as a geographical name database, can help as a dictionary to improve the results. For comparison, we ran the OCR application directly on the two maps. The OCR application could recognize only non-overlapping labels in the horizontal direction, which are about two-thirds of the words in the two maps.

7. Discussion and Future Work

In this paper, we presented an approach to automatically recognize text labels in raster maps. Our approach focuses on locating individual strings and detecting the string orientations to then leverage the horizontal text recognition capability of commercial OCR software. By doing so, our approach requires little user training and benefits from future improvements on commercial OCR software. Our experiments show accurate results on detecting string orientations and recognizing text labels. In the future, we plan to include additional knowledge of the map region to build a dictionary for improving the OCR accuracy on overlapping characters.

Table 1. OCR results (P. is precision and R. is recall)

Map	Char. P.	Char. R.	Word P.	Word R.
RM	95.6%	93.1%	76.3%	79.3%
ITM	96.3%	95%	81.5%	85.2%

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.

[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] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on PAMI*, 24(5):603–619, 2002.

[5] 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.

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

[7] 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.

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

[9] 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.