Integrating Text Recognition for Overlapping Text Detection in Maps

Narges Honarvar Nazari*, Tianxiang Tan*, Yao-Yi Chiang**
****Department of Computer Science, University of Southern California, Los Angeles, CA, USA***
*****Spatial Sciences Institute, University of Southern California, Los Angeles, CA, USA***

ABSTRACT

Detecting overlapping text from map images is a challenging problem. Previous algorithms generally assume specific cartographic styles (e.g., road shapes and text format) and are difficult to adjust for handling different map types. In this paper, we build on our previous text recognition work, Strabo, to develop an algorithm for detecting overlapping characters from non-text symbols. We call this algorithm Overlapping Text Detection (OTD). OTD uses the recognition results and locations of detected text labels (from Strabo) to detect potential areas that contain overlapping text. Next, OTD classifies these areas as either text or non-text regions based on their shape descriptions (including the ratio of number of foreground pixels to area size, number of connected components, and number of holes). The average precision and recall of OTD in classifying text and non-text regions were 77% and 86%, respectively. We show that OTD improved the precision and recall of text detection in Strabo by 19% and 41%, respectively, and produced higher accuracy compared to a state-of-the-art text/graphic separation algorithm.

Keywords

Digital Map Processing, Spatial Databases, Optical Character Recognition, Geographic Information System

1. INTRODUCTION

Detecting overlapping text from map images is a challenging problem, and a great deal of algorithms has been proposed to extract the overlapping text from map images [5]. Although the previous methods can detect overlapping characters, their algorithms assume specific cartographic styles and cannot be easily generalized for handling various types of maps. In [7] and [9], the authors proposed algorithms to extract text from United States Geological Survey (USGS) maps. They exploit the cartographic style of USGS maps (e.g., text can only overlap with road lines) to remove the non-text symbols and group the remaining symbols as text labels. In [2], authors assume that the constituent strokes of characters are usually shorter segments in comparison with those of graphics. Yet, for maps that have many overlapping non-text symbols (e.g., linear objects), there are often non-text constituent strokes that have a similar size as the text constituent strokes. In addition, the length threshold that defines a short segment needs to be re-adjusted if a different scan resolution is used.

In our previous work [3], we developed an open source text recognition system, Strabo, to detect and recognize text labels in map images. Strabo does not handle overlapping characters well. Figure 1 shows two detected text labels that miss at least two characters due to the overlapping of text and non-text regions.

In this paper, we present an algorithm, denoted as Overlapping Text Detection (OTD). The main contribution of OTD is an approach that exploits the differences in shapes between alphabetical characters and non-text symbols to detect overlapping text. OTD receives the detected text labels from Strabo as input and searches the neighboring areas of the detected text labels to find and recover overlapping characters. Although in this paper we demonstrate OTD using Strabo, OTD assumes a generic input (the detected text locations and recognized characters) and can work with other text detection and recognition algorithms.

To detect the overlapping text areas, OTD works as follows. First, OTD creates the minimum enclosing box for each detected text label to represent each text label with its orientation. Second, OTD detects the potential overlapping text areas based on the locations and the recognition results of detected text labels. Third, OTD extracts the shape features of potential text areas and classifies the areas as text or non-text.

The remainder of this paper is organized as follows. Section 2 presents the overall approach for identifying the overlapping text regions. Section 3 describes the experimental results. Section 4 discusses the related work, and Section 5 concludes the paper by a discussion of the overall findings and future research directions.

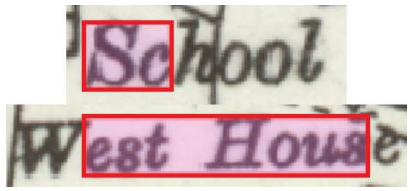

Figure 1. The rectangles show the bounding boxes of detected text labels. In the top and down images some characters have not been detected due to the overlap of text labels and non-text symbols.

2. Overall Approach for Identifying Overlapping Text Regions

In this section we present the technical details of OTD (Figure 2). We first provide a brief background on our previous text detection and recognition work (section 2.1). Then we describe how OTD detects overlapping text automatically (sections 2.2-2.4).

2.1. Background: Strabo

In our previous work [3], we developed a general approach to detect and recognize text labels in map images. Strabo first extracts text pixels from a map using text colors. Once the text pixels are extracted, the text detector of Strabo expands pixel areas when certain conditions are satisfied to group nearby pixels into characters and then nearby characters into strings. The conditions are based on cartographic labeling principles including the rules that characters in one map label are similar in size and are closer than the characters in two separate labels. Because map labels can be in various orientations, Strabo detects the label orientations automatically and rotates every label to the horizontal direction. Finally, Strabo uses an Optical Character Recognition (OCR) package (e.g., Tesseract-OCR) to convert the horizontal text labels to machine-readable data.

Strabo sends the bounding boxes, character recognition results, and orientations of the detected text labels to OTD.

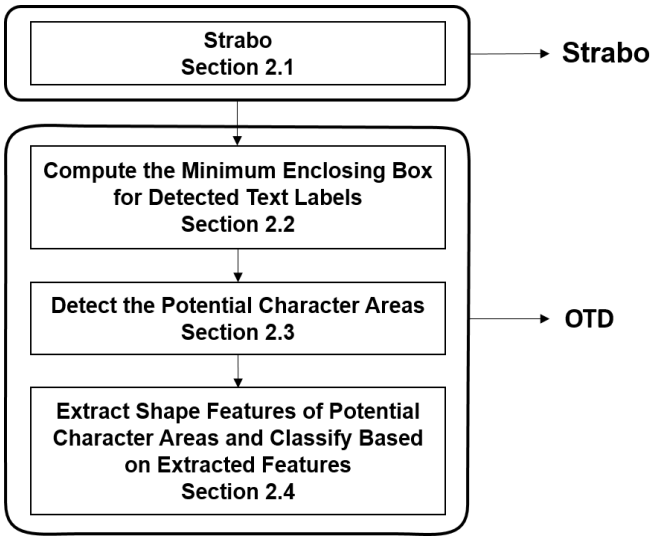

Figure 2. The Workflow of OTD.

2.2. Compute the Minimum Enclosing Box for Detected Text Labels

To explore the neighborhood of the detected text labels for recovering overlapping characters, OTD needs to have a bounding box that represent each detected text label with its orientation and location. OTD calculates the minimum enclosing box (MEB) to represent the detected text labels and their orientations. A MEB for an object is the smallest bounding box within which all points of object lie. The orientation of a MEB represents the orientation of the detected text label. The left image in Figure 3 shows the MEB of the label "School". Since "School" is horizontal, its MEB is a horizontal rectangle. The right image in Figure 3 shows MEB of the label "STREE". Since "STREE" is non-horizontal, its corresponding MEB is a rotated (from the horizontal direction) rectangle.

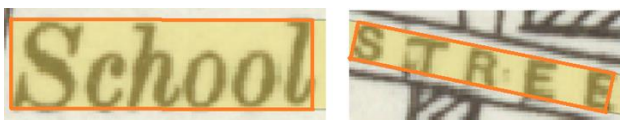

Figure 3. Detected text labels (from Strabo) "School" and "STREE" and their corresponding minimum enclosing boxes (from OTD).

2.3 Detect the Potential Character Areas

In this section we explain how OTD detects the potential text areas in the neighborhood of the detected text labels. The dashed areas in Figure 4 are potential text areas for the labels "Allot". The potential text areas may contain one or more overlapping characters. Since OTD does not make a prior assumption about the number of overlapping characters, it iteratively explores whether or not there is a character in the potential text areas.

In every iteration OTD creates a rectangle that has the same dimension (e.g., width, height, and orientation) as the minimum enclosing box of one character. We call this rectangle as the potential character area (i.e., one potential text area could contain one or more potential character areas). In Figure 5 the blue rectangle is the minimum enclosing box of the label "ALL", the red rectangle is the minimum enclosing box of character "A", and the purple rectangles are the first potential character areas on either side of "ALL".

OTD classifies the potential character areas as text or non-text based on their shape features. If OTD classifies the potential character area as text, it continues to explore the rest of areas. This classification process has two stopping criteria: (1) OTD stops if a potential character area has too few foreground pixels, or (2) if two successive potential character areas are classified as non-text (the classification process is discussed in section 2.4). The reason for the second stopping criterion is that the probability of finding a character after two successive non-text areas is low. For instance, assuming OTD classifies the potential character area correctly with a success rate of 70%, the probability that OTD classifies two successive potential character areas incorrectly is then 9% (30% * 30%).

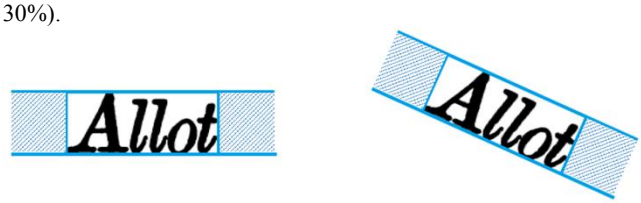

Figure 4. Blue bounding boxes in both images are minimum enclosing boxes of labels "Allot". The dashed areas are potential text areas.

Figure 5. The blue rectangle is minimum enclosing box of label "All", the red rectangle is minimum enclosing box of character "A", and the purple rectangles are potential character areas.

In the following paragraphs, we explain how OTD computes the orientation, width, and height of a potential character area. The orientation of potential character area is the same as the orientation of the minimum enclosing box of a detected text label. The width and height of a potential character area depend on the recognized characters (of the detected label) and whether the potential character area is a prefix or postfix to the detected label.

For each detected text label, OTD first computes its width and the number of characters inside it from the text recognition results (of Strabo). By dividing the bounding box width by the total number of recognized characters, OTD computes the average character width for that detected text label. Because uppercase and lowercase characters can have significantly different character width, the average character width cannot be used directly for generating the potential character areas.

To compute the width of a potential character area, OTD considers three scenarios for detected text labels (Figure 6). In Figure 6 and 7 the purple rectangles show bounding boxes of the detected text labels, and the blue rectangles show the potential character areas. In Figure 6 "Width" is the width of detected text label, "N" is number of characters of the detected text label, "Width/N" is the average character width of detected text label, and ULWR is the ratio of uppercase character width to lowercase character width (Uppercase-to-Lowercase-Width-Ratio).

The three scenarios for computing the width of detected text labels are as follows. The first scenario represents the detected text labels that have their first recognized character in uppercase and the rest of characters in lowercase (the top image in Figure 6). In this

case since the first uppercase character of text label has been detected, OTD does not explore the left potential character areas. The width of the right potential character area (for finding lowercase characters) is the same as the average character width, which is Width divided by N. Here we assume that the number of lowercase characters in the detected label is larger than uppercase characters and hence the average character width is similar to the width of a lowercase character.

The second scenario represents the detected text labels that have first character in lowercase (the middle image in Figure 6). In this case OTD expects to detect an uppercase character in one of the potential character areas to the left of the detected label. Since in general the average width of uppercase characters is greater than the average width of lowercase characters, the width of the left potential character area is ULWR times greater than the average character width. The width of right potential character area is the same as average character width (Width divided by N).

The third scenario represents the detected text labels that have all characters in uppercase (the bottom image in Figure 6**).** In this case, if there are overlapping characters in the left or right potential character areas, they must be in uppercase. The width of the left and the right potential character areas are the same as the average character width (Width divided by N).

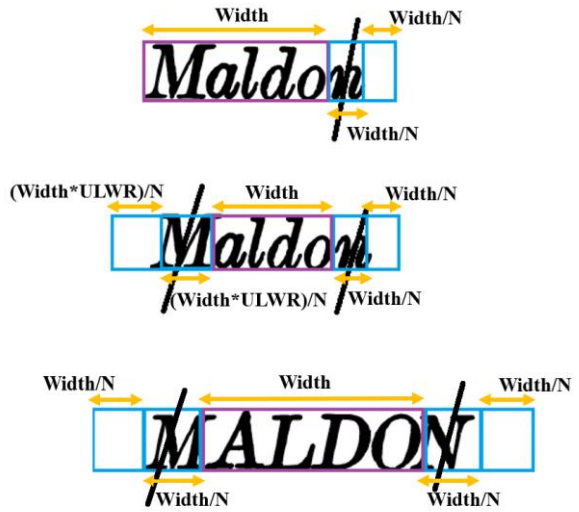

Figure 6. Three scenarios for computing width of potential text areas.

To compute the height of the potential character areas, OTD also considers three scenarios of the recognized text (Figure 7). In English scripts, some characters have larger heights than the other characters. Typically, all uppercase characters have an equal height. Characters “b”, “d”, “f”, “h”, “k”, “l”, and “t” have the same height as the uppercase characters, and they are right on top of the writing line. Characters “g”, “j”, “p”, and “q” have the same height as the uppercase characters but more than half of their shape is under the writing line. The rest of the English characters have smaller heights and they are located on top of the writing line. We consider all uppercase characters and “b”, “d”, “f”, “h”, “k”, “l”, “t” as the first group, characters “g”, “j”, “p”, and “q” as the second group, and the rest as the third group.

Figure 7 shows the three scenarios for computing the height of detected text labels where “Height” is the height of detected text label, “FTHR” is First-to-Third-Group-Height-Ratio, and FSHR is First-to-Second-Group-Height-Ratio (explained in the following paragraphs). The first scenario represents the text labels that have all characters from the third group (the top image in Figure 7). In this case, the height of the potential character area is FTHR times the height of the minimum enclosing box of the detected label. The second category consists of text labels with characters from the second and third groups (the middle image in Figure 7). In this case, the height of the potential character area is FSHR times the height of the minimum enclosing box of the detected label. The third category consists of the text labels that have at least one character from the first group (the bottom image in Figure 7). In this case, the height of the potential text areas is the same as the height of the minimum enclosing box of the detected labels.

As explained in the previous paragraphs, we use the ULWR, FTHR, and FSHR ratios to compute the height and the width of potential character areas. ULWR is the ratio of the average widths of uppercase characters to the average widths of the lowercase characters. FTHR is the ratio of the average heights of characters in the first group to the average heights of the characters in the third group. FSHR is ratio of the average heights of characters in the first group to the average heights of characters in the second group. The measure of these three aforementioned ratios depend on the font type and do not change with the scanning resolution. To estimate the three aforementioned ratios, we use the Arial font which has common shape features of the font types in our test maps (section 3). We empirically compute the measures of ULWR, FTHR, and FSHR for Arial font, which are 1.3, 1.5, and 1.36, respectively.

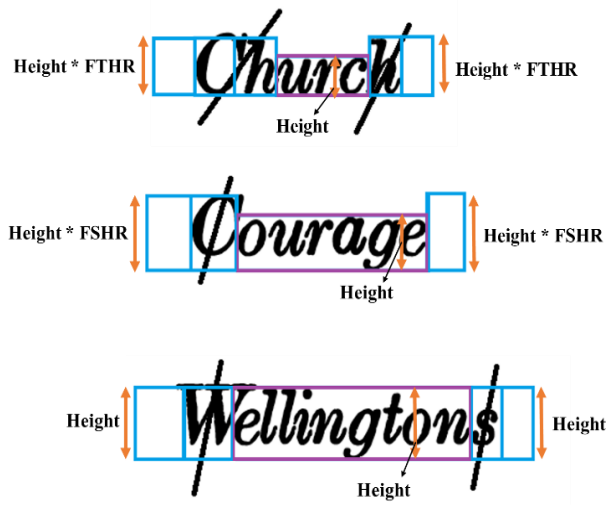

Figure 7. Three scenarios for computing height of potential text areas.

2.4 Extract Shape Features

After computing the size and location of potential character areas for every detected text label, OTD scans the foreground pixels of these detected areas (the foreground pixels can contain text and/or non-text objects). OTD uses four shape features to classify the potential character areas as text or non-text label. These features are: (1) the ratio of the number of foreground pixels to the size of a potential character area, (2) the number of connected components, (3) the number of holes, and (4) the presence of linear objects in a potential character area.

The first feature uses the ratio of number of foreground pixels to the size of potential character area. In the minimum enclosing box of a character, the proportion of the foreground pixels to the size of the bounding box cannot be less than a specific threshold (Figure 8). We empirically define the pixel-to-size-ratio threshold as 0.2.

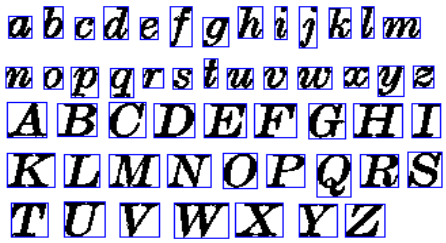

Figure 8. English lowercase and uppercase characters and their corresponding bounding boxes.

The second feature is based on the number of connected components. Except "i" and "j", the rest of the English characters has only one connected component. Although the number of connected components of English characters is at most two, the potential character areas do not always completely encompass every pixel of a character and can contain pixels from other symbols (e.g., an overlapping road). The left image in Figure 9 shows the text label "Grisz". Since characters "G", "r", "s", and "z" overlap with lines, the only detected character is "i". Since "i" has a smaller width than the other characters of text label "Grisz", OTD cannot create potential character areas with the exact width as the overlapping characters. In the result (the right image of Figure 9), the majority of the potential character areas have more than one connected component. Since such cases could appear, OTD considers the potential character areas that contain more than five connected components as non-character areas.

Figure 9. The left image shows potential character areas for the detected text label "Grisz". In the right image, each color shows a connected component in each potential character area.

The third feature limits the number of holes in a character. The majority of English characters do not have any hole in their structure. The rest of the characters have either one hole or two holes. Although the maximum number of holes in a character is two, the potential character areas could contain partial characters (e.g., Figure 9). OTD considers the potential character areas that contain more than five holes as non-character areas.

Maps usually contain many linear objects such as roads and boundaries. These linear objects can overlap with characters. The forth feature helps OTD to find the potential character areas that contain only non-overlapping linear objects. To detect non-overlapping linear objects, OTD explores the intersections of the connected components in a potential character area and the area edges. If a potential character area contains a connected component that intersects two parallel edges of the area, the connected component can be a linear object. Figure 10 and 11 show four potential character areas that contain a connected component intersecting two parallel edges.

Depending on the intersecting edges, in Figure 10 OTD considers the height of the area as the length of the connected component while in Figure 11 uses the width. In both cases, OTD divides the number of foreground pixels of the connected component by the length to calculate the thickness of the component. A linear object has a smaller average thickness than an overlapping linear object (with a character). Therefore, if the average thickness is smaller than a threshold OTD classifies the connected component as a linear object.

Figure 10. The left image shows a character overlapping with a linear object and the right image shows a linear object.

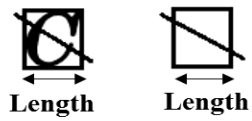

Figure 11. The left image shows a character overlapping with a linear object and the right image shows a linear object.

3. Experiments

We tested the performance of OTD on 1920 6-inch Historical Ordnance Survey maps of the United Kingdom (UK). We tested on 10 tiles within these scanned maps, each covering 1,000 x 1,000 square meters in the TQ grid (near London) of the British National Grid (equal to 1,512 x 1,512 pixels).

We performed two experiments to evaluate the OTD algorithm. First, we evaluated OTD in detecting and classifying overlapping character areas. Next, we evaluated the overall performance of text detection in Strabo before using OTD, Strabo after using OTD, and compared the result with a baseline approach from Cao et al.'s algorithm [2].

For the first experiment, OTD received the bounding boxes, character recognition results, and orientations of the detected text labels from Strabo. OTD explored the potential character areas of text labels to detect and classify overlapping characters. For each detected text label from Strabo, we manually identified the missing characters as the ground truth (i.e., the text areas near the detected text labels that were not detected by Strabo).

Table 1 shows the precision and recall of the first experiment. The last column is the number of ground truth. The average precision of OTD for detecting the overlapping characters in 10 map images was 77% and the average recall was 86%. The image with the lowest recall was image 4. The reason that image 4 had low recall was that it had only two false negative and two true positive. In other words, there were not enough ground truth in image 4. Figures 12-15 show example results of OTD.

Figure 12. In the left image, OTD recovered two overlapping characters ("No") of the detected text label "rthend" from the potential character area to the left of label. In the right image, OTD recovered two overlapping characters ("ts") of the detected text label "Smyat" from the potential character area to the right of label.

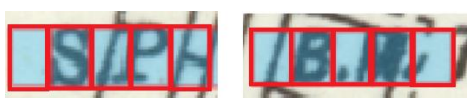

Figure 13. In the left image, the label "S" was detected by Strabo. By exploring the potential character areas to the right, OTD detected the overlapping character "P". In the right image, the label "B" was detected by Strabo, and OTD recovered the overlapping character "M" by exploring the potential character areas to the right of label.

Figure 14. Top image: OTD detected four missing characters ("hool") of the detected text label "Sc" (from Strabo). Bottom image: OTD detected two missing characters ("W" and "e") of the detected text label "est Hous".

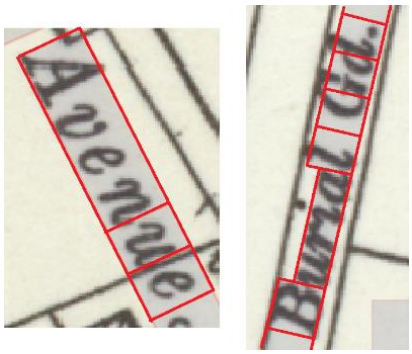

Figure 15. In the left and right images, OTD recovered the overlapping characters for the non-horizontal labels "Aven" and "uria" detected by Strabo. In the left image, OTD recovered two characters "ue". In the right image, OTD recovered the uppercase character "B" to the left and characters "l Gd" to the right.

Table 1. Precision and Recall of OTD in 10 test map images.

	Precision	Recall	Number of Ground Truth
Image 1	86%	100%	18
Image 2	73%	95%	31
Image 3	78%	92%	64
Image 4	100%	50%	4
Image 5	72%	86%	108
Image 6	62%	89%	15
Image 7	88%	80%	53
Image 8	74%	78%	16
Image 9	81%	81%	65
Image 10	77%	80%	31
Average	77%	86%	405

The errors in OTD were from three main categories. First, OTD could not correctly classify the text labels that were located in the neighborhood of many non-text symbols (Figure 16). In this case either the number of connected components or number of holes in the potential character area were more than the thresholds. Second, OTD could not correctly classify non-text symbols that had similar shape features as characters (Figure 17). In this case, the number of connected components and number of holes in the potential character area are usually less than specified thresholds. To solve the two aforementioned issues in our future work, we could explore whether or not each of connected components or holes in the potential character area could be a part of a character based on their shape features. Third, incorrect results from Strabo lowered the accuracy of OTD (Figure 18 and 19). If Strabo computed incorrect orientations or recognition results for the detected text labels, OTD could not find the accurate potential character areas.

Figure 16. OTD explored the potential character areas of the detected label "MA". OTD classified the overlapping characters "L" and "O" as text. Since characters "D" and "N" overlapped with many non-text symbols, the number of holes was more than the specified threshold and OTD misclassified "D" and "N" as non-text.

Figure 17. In the left image, OTD explored the potential character areas of the detected label "urch". OTD explored the left potential character areas of detected label and classified them as text ("Ch"). The right potential character areas are non-text labels with similar shape features as text. OTD misclassified the right potential areas as text. In the right image, OTD explored the right potential areas of the detected label "Hous". OTD classified the overlapping character "e" as text and also misclassified its neighboring non-text symbol as text.

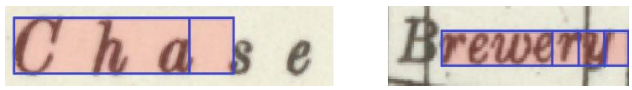

Figure 18. The left image shows the detected labels "Cha". Since Strabo recognized the label as "Cfza" and OTD used the number of recognized characters for computing the average character width, OTD could not detect the exact potential character areas. The right image shows detected label "rewe". Since Strabo recognized the first character of the detected labels as uppercase ("R"), OTD did not explore the left potential character areas and the uppercase character "B" could not be retrieved.

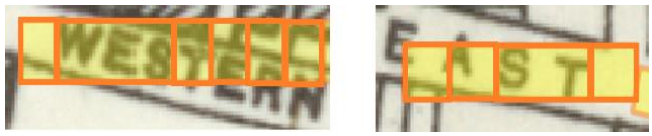

Figure 19. Due to the incorrect orientations detected by Strabo, OTD could not detect the correct potential text areas of detected labels.

For the second experiment, we evaluated the performance of detecting individual text labels using the Cao et al.'s algorithm [2], using Strabo without OTD, and using Strabo with OTD. Table 2 and 3 show the precision, recall, and F-measure of experiment results. As shown in table 2 and 3, OTD improved the precision, recall, and F-measure of text detection in Strabo by 19%, 41%, and 29%, respectively. OTD produced more accurate results compared to Cao et al.'s algorithm and improved the precision, recall, and F-measure by 538%, 83%, and 231%, respectively. The ground truth in this experiment was every text label in the test map. Since the output of all tested algorithms were bounding boxes that contain text labels, a true positive for this experiment was a bounding box with the following features: (1) a bounding box that were completely encompassed the minimum enclosing box of a text label, (2) a bounding box with a width less than 1.5 times the width of the minimum enclosing box of detected text label, and (3) a bounding box with a height less than 1.5 times the height of the minimum enclosing box of detected text label. In Figure 20, the blue rectangles are the minimum enclosing boxes of the ground truth "Allot" and the orange rectangles are bounding boxes of detected text labels. Since the width and height of detected text label are less than 1.5 x the width of text label or 1.5 x the height of text label, we considered the detected text label as a true positive. This orange rectangle served as a buffer for which OCR software could still treat the detected label as a single line text.

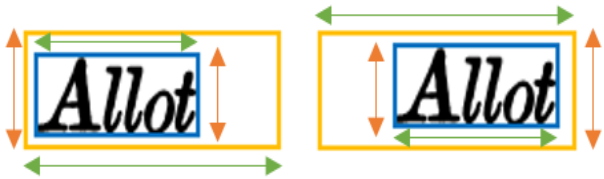

Figure 20. The blue rectangles show the minimum enclosing boxes of the text label "Allot", and orange rectangles show the bounding box of the detected text label.

Table 2. Precision and recall of text detection for Cao et al.'s algorithm, Strabo without OTD, and Strabo with OTD in 10 test maps. "P" in table is precision, "R" is recall, "GT" is the number of Ground Truth, and Avg is Average.

	Cao et al.		Strabo		OTD		GT
	P	R	P	R	P	R	
1	20%	61%	77%	63%	97%	84%	38
2	10%	27%	50%	52%	66%	61%	83
3	13%	26%	53%	38%	73%	60%	72
4	2%	31%	42%	43%	39%	49%	35
5	13%	27%	34%	29%	54%	58%	91
6	7%	37%	42%	37%	48%	39%	38
7	15%	32%	75%	38%	69%	52%	71
8	5%	38%	67%	54%	58%	69%	26
9	5%	21%	30%	22%	42%	40%	73
10	13%	29%	26%	39%	25%	46%	93
Avg	8%	30%	43%	39%	51%	55%	620

The Cao et al.'s algorithm received three parameters as input. We tested the Cao et al.'s algorithm on 10 different sets of parameters to have the best results for each test map. OTD outperformed the precision and recall of Cao et al.'s algorithm. Figures 21 and 22 show the results of the three aforementioned experimental settings. The top images of Figures 21 and 22 (results of Strabo using OTD) show the bounding boxes of the detected text labels (from Strabo) and potential character areas. As shows in the top image of Figure 21, OTD explored the potential character areas of text labels on either side of a detected label. If OTD classified the detected potential character area as non-text, it stopped to explore the rest of potential areas (e.g., the text label "PO" in top image of Figure 21). If OTD classified a detected potential character area as text, it continued to explore the rest of potential areas (e.g., the right side of text label "Sc" in top image of Figure 21). The middle and bottom images of Figure 21 and 22 show the results of Strabo without using OTD and Cao et al.'s algorithm, respectively. In Cao et al.'s algorithm, due to the assumption that non-text constituent strokes have similar size as text constituent strokes, many non-text symbols were detected as text. Cao et al.'s algorithm connected some of text labels and non-text symbols (which have been detected as text) and hence their results contain large bounding boxes which are as big as half of map size.

Table 3. F-Measure of text detection results for Cao et al.'s algorithm, Strabo before using OTD, and Strabo after using OTD.

	Cao et al. F-measure	Strabo F-measure	OTD F-Measure
Image 1	30%	70%	90%
Image 2	14%	51%	64%
Image 3	17%	44%	66%
Image 4	4%	42%	43%
Image 5	18%	31%	56%
Image 6	11%	39%	43%
Image 7	21%	50%	59%
Image 8	9%	60%	63%
Image 9	8%	25%	41%
Image 10	18%	31%	33%
Average	16%	41%	53%

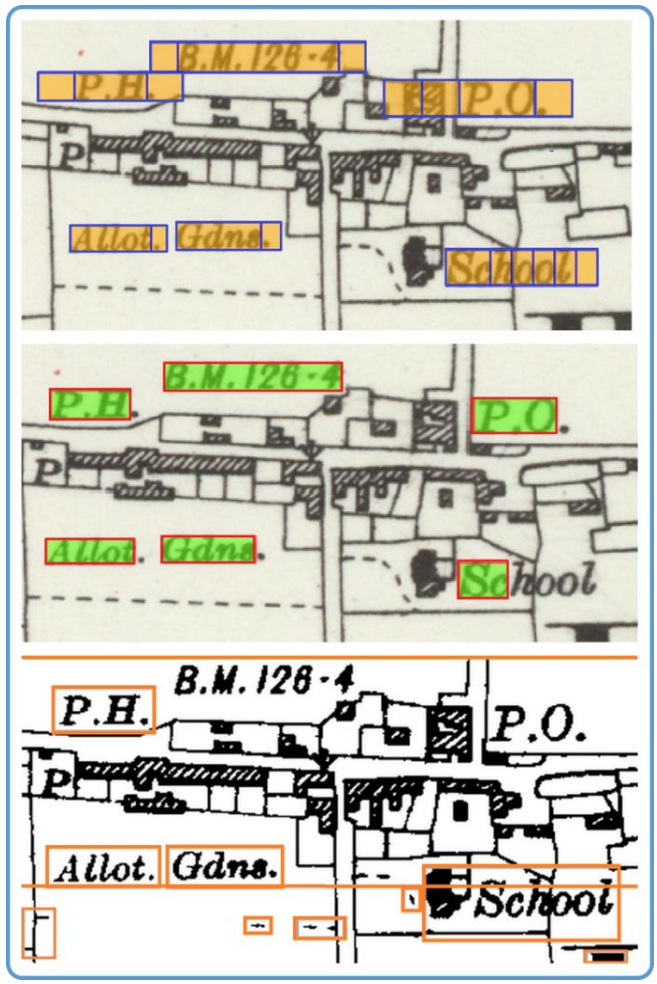

Figure 21. The top image is the result of text detection from Strabo after using OTD. The rectangles show the minimum enclosing boxes of the detected text labels and the potential character areas. The middle image shows the results of Strabo before using OTD. The rectangles show the detected text labels. The bottom image shows the results from Cao et al.'s algorithm. The rectangles show the results of text detection.

Figure 22. Example text detection results. The top images are the result of text detection from Strabo after using OTD. The middle images show the results of Strabo before using OTD. The bottom images show the results from Cao et al.'s algorithm.

4. Related Work

There exists abundant work in the area of text detection and recognition, specifically for processing map documents. Yet, recognizing map labels in scanned maps in general is still not a solved problem [4, 8]. This is because most of the previous work assume specific cartographic styles (e.g. road shapes and text format), which make their algorithms difficult to adjust for handling various map types and scan resolutions.

Fletcher et al. [6] present an algorithm to extract strings in various orientations and font styles from mixed text/graphics images. Their algorithm uses the Hough transformation, widths, heights, and distances between individual connected components to group characters for extracting text labels. Their method can only extract non-overlapping characters.

Cao et al. [2] present an algorithm to detect overlapping text in document images. Their algorithm first removes dashed lines, applies a thinning operator, and then searches for line intersections to identify line segments. To recover the overlapping text, their algorithm assumes that lines of characters are shorter than lines of graphics. Therefore, the thinned foreground objects are decomposed into line segments and shorter lines are grouped into characters. This assumption only holds for specific map types and the length threshold varies significantly for the same map with different scan resolution.

Tombre et al. [10] extend the work of Fletcher et al. [6] to handle overlapping characters in graphic-rich documents. Their algorithm for identifying overlapping characters is similar to Cao et al. [2]. Their algorithm assumes that the overlapping characters intersect non-text objects at only one location (i.e., points that text and graphic meet). In maps, especially historical maps, characters

very often overlap with non-text objects significantly and the aforementioned assumption does not work.

Li et al. [7] present an algorithm to extract street labels from the USGS topographic maps. The street labels in USGS maps very often only overlap with street lines. Their algorithm first detects non-overlapping characters, searches for nearby parallel lines (streets), and then follows the line direction to detect overlapping characters. Their algorithm cannot extract overlapping characters that intersect with other non-text symbols (e.g., buildings, trees).

Pezeshk et al. [9] present an algorithm to remove linear objects and some specific types of graphic objects (e.g., solid objects and dashed lines) in maps to extract text from images. Similar to Li et al. [7], their algorithm assumes a specific cartographic style and can only apply to maps that are similar to the USGS topographic maps (i.e., with a similar road symbology and label placement).

All of these algorithms [2, 6, 7, 9, and 10] assume that the characters only touch straight roads or linear objects, they do not consider the case that the characters overlapping curved lines or symbols (e.g., trees and buildings). In contrast, our approach utilizes shape features based on font types and are independent from the (scan) resolution and other cartographic styles.

Velazquez et al. [11] compute imaginary lines following the orientation of non-overlapping characters to create a rectangle for detecting overlapping characters along the same orientation. However, this algorithm can only detect overlapping characters between non-overlapping characters.

In addition, there is a great deal of work for recognizing text in photos of natural scenes [12], and a number of them reports high recognition accuracy, such as the Google PhotoOCR using deep learning and large language models [1]. This type of text recognition work generally assumes that text regions have a significant difference in response to a certain image filter (e.g., edge detectors and corner detectors) to first localize text regions in natural scenes. This assumption does not apply to document images (e.g., maps) where graphical features in documents can have similar response to these image filters as text.

5. Discussions and Future Work

The contribution of this paper is an approach that exploits the differences in shapes between characters and non-text symbols to detect overlapping text. OTD receives the detected text labels from a text detector/recognizer, creates the minimum enclosing boxes for text labels, detects the potential character areas, and classifies the areas as text or non-text based on their shape features. We showed that OTD outperformed two existing approaches (Strabo and Cao et al.'s algorithm) in text detection. In this paper we use empirical thresholds to classify potential character areas to text and non-text. We plan to test a broader range of map types and use machine learning approaches like the artificial neural networks to automatically classify potential text areas.

REFERENCES

[1] A. Bissacco, M. Cummins, Y. Netzer and H. Neven, "PhotoOCR: Reading Text in Uncontrolled Conditions," in *International Conference on Computer Vision*, 2013.

[2] R. Cao and C. L. Tan, "Text/Graphics Separation in Maps," in *Graphics Recognition Algorithms and Applications, 4th International Workshop*, 2001.

[3] Y.-Y. Chiang and C. A. Knoblock, "A general approach for extracting road vector data from raster map," *International Journal on Document Analysis and Recognition,* vol. 16, no. 1, pp. 55-81, 2013.

[4] Y.-Y. Chiang and C. A. Knoblock, "Recognition of Multi-Oriented, Multi-Sized, and Curved Text," in *International Conference on Document Analysis and Recognition*, pp. 3199–3202, 2011.

[5] Y.-Y. Chiang, S. Leyk and C. A. Knoblock, "A Survey of Digital Map Processing Techniques," *ACM Computing Surveys,* vol. 47, no. 1, pp. 1-44, 2014.

[6] L. A. Fletcher and R. Kasturi, "A robust algorithm for text string separation from mixed text/graphics images," *Pattern Analysis and Machine Intelligence,* vol. 10, no. 6, pp. 910-918, 1988.

[7] L. Li, G. Nagy, A. Samal, S. Seth and Y. Xu, "Integrated text and line-art extraction from a topographic map," *International Journal on Document Analysis and Recognition,* vol. 2, no. 4, pp. 177-185, 2000.

[8] G. Nagy, A. Samal, S. Seth, T. Fisher, E. Guthmann, K. Kalafala, L. Li, S. Sivasubramaniam and Y. Xu, "Reading Street Names from Maps - Technical Challenges," in *The GIS/LIS Conference*, 1997.

[9] A. Pezeshk and R. Tutwiler, "Automatic Feature Extraction and Text Recognition From Scanned Topographic Maps," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 49, no. 12, pp. 5047-5063, 2011.

[10] K. Tombre, S. Tabbone, L. Pélissier, B. Lamiroy and P. Dosch, "Text/Graphics Separation Revisited," in *DAS '02 Proceedings of the 5th International Workshop on Document Analysis Systems V*, 2002.

[11] A. Velázquez and S. Levachkine, "Text/Graphics Separation and Recognition in Raster-Scanned Color Cartographic Maps," in *Graphics Recognition, Recent Advances and Perspectives, 5th InternationalWorkshop*, 2003.

[12] Q. Ye and D. Doermann, "Text Detection and Recognition in Imagery: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 37, no. 7, pp. 1480-1500, 2014.

Author Biography

Narges Honarvar Nazari is a master student in Computer Science at the University of Southern California. She received her Bachelor degree in Computer Software Engineering from Shahid Beheshti University. Her research focuses on image processing, document analysis, computer vision, and geospatial information sciences.

Tianxiang Tan received his Bachelor degree in Software Engineering at Sun Yat-sen University in 2014. He is currently working toward his master degree in Computer Science at the University of Southern California. His research interests include image processing, machine learning and computer vision.

Yao-Yi Chiang is an Assistant Professor (Research) at the University of Southern California (USC), Spatial Sciences institute. He received his Ph.D. degree in Computer Science from USC in 2010; his Bachelor degree in Information Management from National Taiwan University in 2000. His general area of research is information integration, with a focus on acquiring, modeling, fusion, and visualization of geographic data from heterogeneous sources. In particular, Dr. Chiang is an expert in digital map processing and geospatial information system (GIS). He has published a number of articles on automatic techniques for geospatial data extraction and integration. Prior to USC, Dr. Chiang worked as a research scientist for Geosemble Technologies, which was founded based on a patent on geospatial data fusion techniques and he was a co-inventor.