2011 International Conference on Document Analysis and Recognition

Recognition of Multi-Oriented, Multi-Sized, and Curved Text

Yao-Yi Chiang
University of Southern California,
Information Sciences Institute
and Spatial Sciences Institute,
4676 Admiralty Way, Marina del Rey, CA 90292, USA
Email: yaoyichi@isi.edu

Craig A. Knoblock
University of Southern California,
Department of Computer Science
and Information Sciences Institute,
4676 Admiralty Way, Marina del Rey, CA 90292, USA
Email: knoblock@isi.edu

***Abstract*—Text recognition is difficult from documents that contain multi-oriented, curved text lines of various character sizes. This is because layout analysis techniques, which most optical character recognition (OCR) approaches rely on, do not work well on unstructured documents with non-homogeneous text. Previous work on recognizing non-homogeneous text typically handles specific cases, such as horizontal and/or straight text lines and single-sized characters. In this paper, we present a general text recognition technique to handle non-homogeneous text by exploiting dynamic character grouping criteria based on the character sizes and maximum desired string curvature. This technique can be easily integrated with classic OCR approaches to recognize non-homogeneous text. In our experiments, we compared our approach to a commercial OCR product using a variety of raster maps that contain multi-oriented, curved and straight text labels of multi-sized characters. Our evaluation showed that our approach produced accurate text recognition results and outperformed the commercial product at both the word and character level accuracy.**

I. Introduction

Text recognition, or optical character recognition (OCR), is an active area in both academic research and commercial software development. Effective text recognition techniques are widely used, such as for indexing and retrieval of document images and understanding of text in pictorial images or videos.

In classic text recognition systems, including most commercial OCR products, the first step is "zoning," which analyzes the layout of an input image for locating and ordering the text blocks (i.e., zones). Next, each of the identified text blocks containing homogeneous text lines of the same orientation is processed for text recognition. However, this zoning approach cannot handle documents that do not have homogeneous text lines, such as artistic documents, pictorial images with text, raster maps, and engineering drawings. For example, Figure 1 shows an example map that contains multi-oriented text lines of multi-sized characters and no zones of homogeneous text lines exist.

To process documents with non-homogeneous text, one approach is to recognize individual characters separately [1, 4, 9], such as utilizing rotation invariant features of specific character sets for character recognition [4]. However, this approach requires specific training work and hence cannot be easily integrated with the classic, well-developed OCR techniques that process homogeneous text. Moreover, recognizing individual characters separately fails to take the advantage of word context, such as utilizing a dictionary to help recognize grouped characters that represent meaningful words.

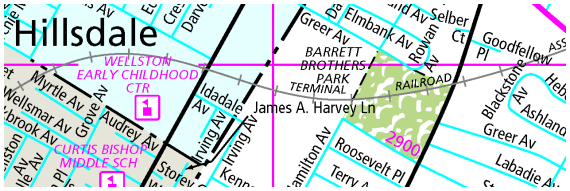


Figure 1. Multi-oriented and multi-sized characters in a raster map from Rand McNally maps

Instead of recognizing individual characters separately, previous work on extracting text lines from non-homogeneous text for text recognition typically handles specific cases, such as specific language scripts [8], straight text lines [5, 10], multi-oriented but similar-sized characters [5, 6]. In our previous work [3], we presented a text recognition approach that locates individual multi-oriented text labels in raster maps and detects the label orientations to then leverage the horizontal text recognition capability of commercial OCR software. Our previous work requires manually specified character spacing for identifying individual text labels and does not consider multi-sized characters.

In this paper, we build on our previous work [3] and present a text recognition technique to dynamically group characters from non-homogeneous text into text strings based on the character sizes and maximum desired string curvature. The hypothesis is that characters in a text string are similar in size and are spatially closer than the characters in two separated strings. Our text recognition technique does not require training for specific fonts and can be easily integrated with a commercial OCR product for processing documents that contain non-homogeneous text.

II. Related Work

Text recognition from documents that contain non-homogeneous text, such as from raster maps [7], is a difficult task, and hence much of the previous research only works

1520-5363/11 $26.00 © 2011 IEEE
DOI 10.1109/ICDAR.2011.281

on specific cases. Fletcher and Kasturi [5] utilize the Hough transformation to group characters and identify text strings. Since the Hough transformation detects straight lines, their method cannot be applied on curved strings. Moreover, their work does not handle multi-sized characters.

Goto and Aso [6] present a text recognition technique to handle multi-oriented and curved text strings, which can have touching characters. Their technique first divides the input document into columns of equal sizes and then detects connected components within each column for further dividing the columns into blocks. Then the connected components in each block are expanded in various orientations to compute the local linearity for extracting text strings. This block-based approach works on touching characters but requires characters of similar sizes.

Velázquez and Levachkine [13] and Pal et al. [8] present text recognition techniques to handle characters in various font sizes, font types, and orientations. Their techniques are based on detecting straight string baselines for identifying individual text strings. These techniques cannot work on curved strings.

Pouderoux et al. [10] present a text recognition technique for raster maps. They identify text strings in a map by analyzing the geometry properties of individual connected components in the map and then rotate the identified strings horizontally for OCR. Roy et al. [11] detect text lines from multi-oriented, straight or curved strings. Their algorithm handles curved strings by applying a fixed threshold on the connecting angle between the centers of three nearby characters. Their orientation detection method only allows a string to be classified into 1 of the 4 directions. In both [10, 11], their methods are based on the assumption that the string curvature can be accurately estimated from the line segments connecting each character center in a string. However, this assumption does not hold when the string characters have very different heights or widths. In contrast, we present a robust technique to estimate the curvature and orientation of a text string and our technique is independent from the character size.

III. Overview of Our Text Recognition Approach

Given a document image, there are three major steps in our approach for text recognition. First, we extract the text pixels from the input document. For an input image, the user provides example text areas where each text area is a rectangle that contains a horizontal string. The user can rotate the rectangle to select a text string that is not horizontally placed in the image. Since each rectangle contains a horizontal string, we exploit the fact that the text pixels are horizontally near each other to identify the colors that represent text in the image and use the identified colors to extract the text pixels [2]. Second, we dynamically group the extracted text pixels into text strings, which is the main focus of this paper. Third, with the identified text strings, we employ our previous work [3] to detect the orientation of each string and rotate the stings to the horizontal direction for text recognition using a commercial OCR product.

This paper focuses on the second step of string identification, which is described in the next section. The details of the other steps are described in our previous work [2, 3].

IV. Identifying Individual Text Strings

Once we extract the text pixels, we have a binary image where each connected component (CC) in the foreground is a single character or a part of a character, such as the top dot of the 'i'. To group the CCs into strings, we present the conditional dilation algorithm (CDA) and Figure 2 shows the pseudo-code of the CDA.

The CDA performs multiple iterations to expand and connect the CCs and then uses the connectivity of the expanded CCs to identify individual text strings. As shown in the ConditionalDilation function in Figure 2, before the first CDA iteration, the CDA sets every CC as expandable. Next, in an iteration, the CDA tests a set of conditions on every background pixel (the TestConditions sub-function) to determine if the pixel is a valid expansion pixel: a background pixel that can be converted to the foreground for expanding a CC. After an iteration, the CDA evaluates each expanded CC (the CountExpandableCC sub-function) to determine whether the CC can be further expanded in the next iteration and stops when there is no expandable CC. We describe the test conditions to determine an expansion pixel and an expandable CC in the remainder of this section.

Character Connectivity Condition An expansion pixel needs to connect to at least one and at most two characters. This is because the maximum neighboring characters that any character in a text string can have is two.

Character Size Condition If an expansion pixel connects to two characters, the sizes of the two characters must be similar. For a character, *A*, and its bounding box, *Abx*, the size of *A* is defined as:

$$Size = Max(Abx.Height, Abx.Width) \quad (1)$$

For the characters connected by expansion pixels, the size ratio between the characters must be smaller than a predefined parameter (the max_size_ratio parameter). For two characters, *A* and *B*, their bounding boxes are *Abx* and *Bbx*, their size ratio is defined as:

$$SizeRatio = \frac{Max(Size(A), Size(B))}{Min(Size(A), Size(B))} \quad (2)$$

This character size condition guarantees that every character in an identified text string has a similar size. We use the size ratio equal to two because some letters, such as the English letter 'l' and 'e', do not necessarily have the exact same size, even when the same font is used.

```
// The number of processed iterations
  IterationCounter = 0;
// The number of expandable connected components
  Expandable_CC_Counter;
// CDA parameters
  double max_size_ratio, max_distance_ratio,
         max_curvature_ratio;

MainFunction void ConditionalDilation(int[,] image)
  FOR EACH connected component CC in image
    CC.expandable = TRUE;
  DO{ TestConditions(image);
      CountExpandableCC(image);
      IterationCounter = IterationCounter+1;
  } WHILE(Expandiable_CC_Counter > 0)
EndMainFunction

SubFunction void TestConditions(int[,] image)
  FOR EACH background pixel BG in image
    IF(PassConnectivityTest(BG)&&PassSizeTest(BG)&&
       PassExpandabilityTest(BG)&&
       PassStringCurvatureTest(BG))
      Set BG to Foreground;
EndSubFunction

SubFunction void CountExpandableCC(int[,] image)
   FOR EACH expanded connected component ECC in image
     IF(HasConnectedToTwoECCs(ECC) ||
       IterationCounter > max_distance_ratio*ECC.char_size)
       ECC.expandable = FALSE;
     ELSE
       Expandable_CC_Counter = Expandable_CC_Counter+1;
EndSubFunction
```

Figure 2. The pseudo-code for the conditional dilation algorithm (CDA)

Character Expandability Condition An expansion pixel needs to connect to at least one expandable CC and the expandability of a CC is determined as follows: before the first CDA iteration, every CC is expandable. After each iteration, the CDA checks the connectivity of each expanded CC and if the expanded CC has already connected to two other CCs, the CC is not expandable.

Next, for the remaining expanded CCs (i.e., the ones with connectivity less than two), the CDA determines the expandability of each CC by comparing the number of iterations that have been done and the original size of each CC before any expansion. This is to control the longest distance between any two characters that the CDA can connect so that the characters in two separated strings will not be connected. For example, in our experiments, we empirically set the longest distance between two characters to 1/5 of the character size (the max_distance_ratio parameter). As a result, for a character of size equal to 20 pixels, the character will not be expandable after 4 iterations, which means this character can only find a connecting neighbor within the distance of 4 pixels plus 1/5 of the size of a neighboring CC.

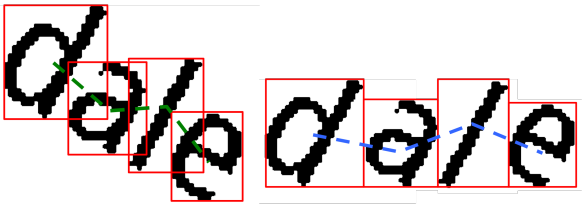
String Curvature Condition If an expansion pixel connects two CCs and at least one of the two CCs has a connected neighbor (i.e., together as a string with at least three characters), the curvature of the set of CCs should be less than the maximum desired curvature. This condition allows the CDA to identify curved strings and guarantees that the characters of the text strings in different orientations will not be connected. However, determining the string curvature without knowing how the characters are aligned is unreliable. For example, considering the text string "Wellington", if we link the mass centers or bounding-box centers of each character to represent the string curvature, the line segments linking any two neighboring characters can have very different orientations since the characters have various heights, such as the links between "We" and the one between "el".

To accurately estimate the curvature of a string, the CDA first establishes a curvature baseline for the string. For example, the left image in Figure 3(a) shows an example string, and the right image shows the rearranged string as if the example string is straight and in the horizontal direction. The CDA generates the rearranged string by first aligning each of the characters vertically and rearranging the characters' positions in the horizontal direction so that the characters are not overlapped. The dashed line in the right image shows the curvature baseline of "dale". This curvature baseline contains two connecting angles: the ones between"dal" and "ale".

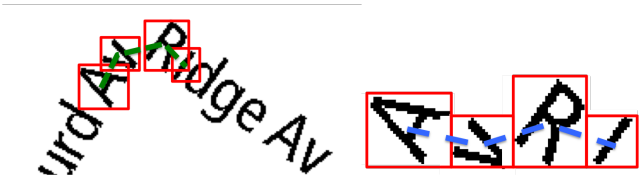
With the curvature baseline, the CDA determines the string curvature by comparing the connecting angles in the original string to the ones in the curvature baseline. For example, Figure 3(c) shows that θ_1 is similar to θ_1' and θ_2 is similar to θ_2' and hence the CDA considers the string "dale" as a straight string (i.e., every original connecting angle is similar to its corresponding one). Figure 3(d) shows an example where θ_1 is very different from θ_1' and hence the CDA considers the string "AvRi" as a curved string.

The CDA uses a curvature parameter to control the maximum desired curvature of a text string (the max_curvature_ratio parameter). If the difference between one connecting angle of a string and the corresponding angle in the string's curvature baseline is larger than the curvature parameter, the string violates the string curvature condition. For example, with the curvature parameter set to 30% from the curvature baseline, any string with curvature within 138° (180° divided by 130%), to 234° (180° multiplied by 130%) will be preserved.

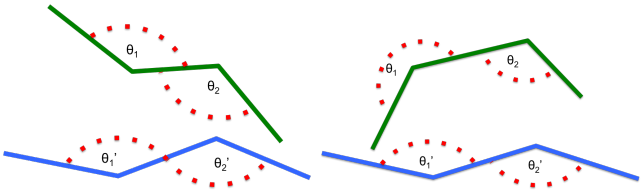
The CDA Output After the CDA stops when there is no expansion pixel, each connected component of the expansion results is an identified text string. For example, in Figure 4, the set of color blobs are the expansion results (each color represents a connected component) and the black pixels overlapped with a color blob belong to an identified string. In Figure 4, the CDA does not group small CCs correctly, such as the dot on top of the character 'i' . This is because these small CCs violate the character size condition. The OCR system will recover these missing small parts in the character recognition step, which is more robust than adopting special rules for handling small CCs in the CDA.



(a) The original string (left) and curvature baseline (right) of "dale"



(b) The original string (left) and curvature baseline (right) of "AvRi"



(c) θ_1/θ_2 is similar to θ_1'/θ_2' (d) θ_1 is very different from θ_1'

Figure 3. Testing the string curvature condition

Figure 4. The CDA output

V. Experiments

We have implemented the techniques described in this paper in our map processing system called Strabo. To evaluate our technique, we tested Strabo on 15 maps from 10 sources, including 3 scanned maps and 12 computer-generated maps (directly generated from vector data).[1] These maps contain non-homogeneous text of numeric characters and the English alphabet. Table I shows the information of the test maps and their abbreviations used in this section. Figure 5 shows one example area in a test map.

We utilized Strabo together with a commercial OCR product called ABBYY FineReader 10 to recognize the text labels in the test maps. For comparison, ABBYY FineReader 10 was also tested alone without Strabo. For evaluating the recognized text labels, we report the precision and recall at both the character and word levels.

Table II shows the numeric results of our experiments. Strabo produced higher numbers compared to using only

[1]The information for obtaining the test maps can be found on: http://www.isi.edu/integration/data/maps/prj_map_extract_data.html

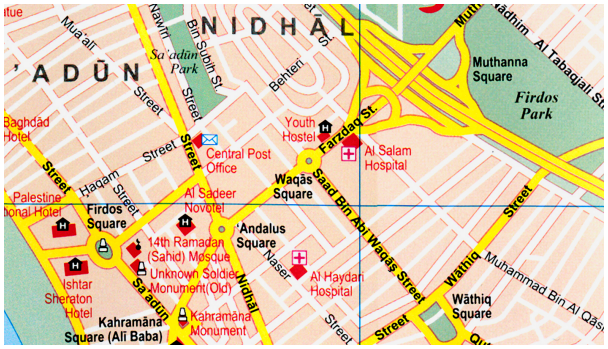


Figure 5. A portion of the GIZI map

Table I
Test maps for experiment

Map Source (abbr.)	Map Type	# Char/Word
International Travel Maps (ITM)	Scanned	1358/242
Gecko Maps (GECKO)	Scanned	874/153
Gizi Map (GIZI)	Scanned	831/165
Rand McNally (RM)	Computer Generated	1154/266
UN Afghanistan (UNAfg)	Computer Generated	1607/309
Google Maps (Google)	Computer Generated	401/106
Live Maps (Live)	Computer Generated	233/64
OpenStreetMap (OSM)	Computer Generated	162/42
MapQuest Maps (MapQuest)	Computer Generated	238/62
Yahoo Maps (Yahoo)	Computer Generated	214/54

ABBYY FineReader 10 in all metrics, especially the recall. ABBYY FineReader 10 did not do well on identifying text regions from the test maps because of the multi-oriented text strings in the maps. ABBYY FineReader 10 alone could only recognize the stings that are in the horizontal or vertical directions. Moreover, ABBYY FineReader 10 could not detect any text region from the Google, OSM, MapQuest, and Yahoo maps and hence the precision and recall are 0 at both the character and word levels.

Overall Strabo achieved accurate text recognition results at both the character and word levels. This is because the CDA successfully grouped the multi-oriented and multi-sized characters into individual text strings for OCR. Moreover, the CDA correctly identified curved strings that have their curvature within the desired curvature ratio (30%), such as the example shown in Figure 6.

The errors in Strabo's results came from several aspects: (i) The poor image quality of the test maps, especially scanned maps, could result in poor quality of text pixels, such as broken characters or the existence of non-text objects in the extracted text pixels. (ii) The CDA might not correctly identify strings with significant wide character spacing. For example, Figure 7 the string "Hindu Kush" in the UNAfg map was not identified correctly. (iii) The CDA might group characters with non-text objects. If there exist non-text objects in the CDA input and a non-text object was close to one end of a string and has a similar size as the ending character, the CDA would connect the end character to the non-text object. A connected-component filter can be used to post-process the extracted text pixel

Table II
TEXT RECOGNITION RESULTS (P. IS PRECISION AND R. IS RECALL)

Source	System	Ch. P.	Ch. R.	Wd. P.	Wd. R.
ITM	Strabo	93.6%	93.3%	83.3%	82.6%
	ABBYY	86.4%	45.6%	57.5%	33%
GECKO	Strabo	93.4%	86.3%	83.1%	77.1%
	ABBYY	77.8%	41%	66.2%	37.2%
GIZI	Strabo	95.1%	77.3%	82%	63.6%
	ABBYY	71.3%	16%	51.4%	10.9%
RM	Strabo	93.4%	94%	87.9%	84.9%
	ABBYY	71.8%	10.4%	23.5%	3%
UNAfg	Strabo	91.5%	88%	82.3%	80.2%
	ABBYY	65.6%	56%	34.8%	36.5%
Google	Strabo	97.3%	91.7%	89.2%	85.8%
	ABBYY	0%	0%	0%	0%
Live	Strabo	94.7%	93.5%	75.3%	76.5%
	ABBYY	51.8%	47.6%	47.8%	53.1%
OSM	Strabo	95.4%	77.7%	74.3%	69%
	ABBYY	0%	0%	0%	0%
MapQuest	Strabo	91.3%	84%	81%	75.8%
	ABBYY	0%	0%	0%	0%
Yahoo	Strabo	69.7%	63.5%	43.1%	40.7%
	ABBYY	0%	0%	0%	0%
Avg.	Strabo	92.7%	87.9%	82%	77.5%
Avg.	ABBYY	71.9%	30%	46.1%	20.6%

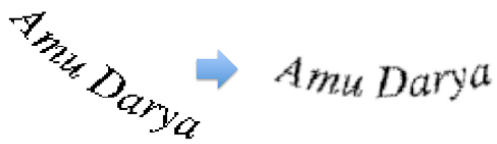


Figure 6. An identified curved string with its rotated image containing the horizontal string for OCR

for removing this type of error. However, the connected-component filter would need careful parameter settings and might also remove characters.

VI. DISCUSSION AND FUTURE WORK

We presented a general text recognition technique for processing documents that contain non-homogeneous text lines. This technique handles multi-oriented, curved and straight text lines of multi-sized characters and requires only three parameter settings. We show that our technique can be easily integrated with a commercial OCR product to support text recognition from documents for which classic layout analysis techniques do not work. In the future, we plan to test this text recognition technique on non-English scripts. We also plan to broaden the coverage of our technique to handle documents with mostly touching characters, such as by incorporating a character segmentation method [12].

ACKNOWLEDGMENT

This research is based upon work supported in part by the University of Southern California under the Viterbi School of Engineering Doctoral Fellowship.

REFERENCES

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

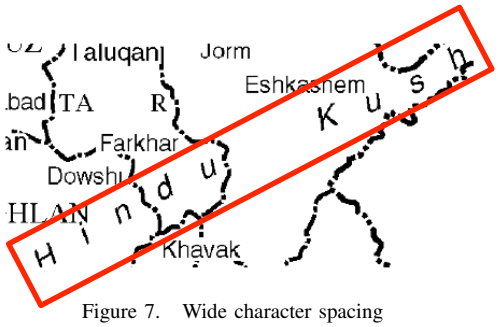


Figure 7. Wide character spacing

[2] Chiang, Y.-Y. (2010). Harvesting Geographic Features from Heterogeneous Raster Maps *Ph.D. Dissertation*, University of Southern California.

[3] Chiang, Y.-Y. and Knoblock, C. A. (2010). An approach for recognizing text labels in raster maps. In *Proceedings of the 20th ICPR*, pages 3199–3202.

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

[5] Fletcher, L. A. and Kasturi, R. (1988). A robust algorithm for text string separation from mixed text/graphics images. *IEEE TPAMI*, 10(6):910–918.

[6] Goto, H. and Aso, H. (1998). Extracting curved text lines using local linearity of the text line. *IJDAR*, 2(2–3):111–119.

[7] Nagy, G., Samal, A., Seth, S., Fisher, T., Guthmann, E., Kalafala, K., Li, L., Sivasubramaniam, S., and Xu, Y. (1997). Reading street names from maps - technical challenges. In *GIS/LIS conference*, pages 89–97.

[8] Pal, U., Sinha, S., and Chaudhuri, B. B. (2003). Multi-oriented english text line identification. In *Proceedings of the 13th Scandinavian conference on Image analysis*, pages 1146–1153.

[9] Pezeshk, A. and Tutwiler, R. (2010). Extended character defect model for recognition of text from maps. In *Proceedings of the IEEE Southwest Symposium on Image Analysis Interpretation*, pages 85–88.

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

[11] Roy, P. P., Pal, U., Lladós, J., and Kimura, F. (2008). Multi-oriented english text line extraction using background and foreground information. *IAPR International Workshop on DAS*, 0:315–322.

[12] Roy, P. P, Pal, U., Lladós, J., and Delalandre, M. (2009). Multi-oriented and multi-sized touching character segmentation using dynamic programming. In *the Proceedings of the 10th ICDAR*, pages 11–15.

[13] Velázquez, A. and Levachkine, S. (2004). Text/graphics separation and recognition in raster-scanned color cartographic maps. In *GREC*, vol 3088 of *LNCS*, pages 63–74.