An OCR-Independent Character Segmentation Using Shortest-Path in Grayscale Document Images

Jia Tse *Member, IEEE*, Dean Curtis *Member, IEEE*, Christopher Jones *Member, IEEE*, and Evangelos Yfantis *Member, IEEE*
Digital Image Processing Laboratory
Department of Computer Science
University of Nevada, Las Vegas
4505 Maryland Parkway, Las Vegas, NV 89154
UNITED STATES OF AMERICA
Tse@unlv.nevada.edu

Abstract

An Optical Character Recognition (OCR) system with a high recognition rate is challenging to develop. One of the major contributors to OCR errors is smeared characters. Several factors lead to the smearing of characters such as bad scanning quality and a poor binarization technique. Typical approaches to character segmentation falls into three major categories: image-based, recognition-based, and holistic-based. Among these approaches, the segmentation path can be linear or non-linear. Our paper proposes a non-linear approach to segment characters on grayscale document images. Our method first determines whether characters are smeared together using general character features. The correct segmentation path is found using a shortest path approach. We achieved a segmentation accuracy of 95% over a set of about 2,000 smeared characters.

1. Introduction

An important aspect of document image analysis is recognizing the text of the document. An Optical Character Recognition (OCR) system with a high recognition rate is challenging to develop. According to a study conducted by ISRI, most OCR errors are ascribed to character segmentation errors [9]. Thus, segmentation is necessary to isolate the characters. The major types of strategies are categorized as: image-based, recognition-based, and holistic-based [2].

In image-based segmentation, general well known properties of characters are used. Some of the properties used are character width and height, the analysis of intraword and interword spaces, etc [4] [8].

In recognition-based segmentation, an OCR system is used to recognize the character and provide a confidence level to ensure the segmentation path is correct. This approach can be combined with an image-based approach to improve results [7] [8] [11].

In a holistic approach, the entire word is recognized instead of the characters of the word [3] [10]. The holistic type of approach is favorable in applications with a limitation of the number of possible words, such as the months of a year, or days of a week.

Segmentation algorithms can also be classified as linear and nonlinear. Linear approaches segment merged characters by linear cuts, mostly vertical cuts. Since not all connected characters can be separated with a linear cut, a more interesting approach is a nonlinear approach. Several works have been done in this direction, [1] [5] [6] [13].

In this paper, we present a non-linear character segmentation using shortest paths. For each word image, we find the connected components and determine whether any of the components are smeared. Then our shortest-path algorithm is applied on the grayscale image of smeared characters to determine the best segmentation path. We achieved an accuracy of 94.71%.

2. Character Segmentation

Our algorithm first finds the connected components of the binary word image. Next, the connected components are analyzed to determine whether or not each one of them consists of one or more characters. The binarization threshold during preprocessing is adjusted lower to favor characters being smeared rather than broken. If the connected component consists of one or more characters, we apply our segmentation algorithm on the grayscale image to dissect the

0-7695-3069-9/07 $25.00 © 2007 IEEE
DOI 10.1109/ICMLA.2007.21

(a) (b)

Figure 1. (a) a 'p' smeared together with an 'e'. (b) the pixels of intersection. Notice that intensity patterns are lighter among the borders of the characters.

characters one at a time.

By the study and experimentation of document images, we arrived at a conclusion relating to the nature of smeared characters. One observation is that some of the characters that appear smeared in the binary image are in fact not smeared in the original image. The cause of the smearing was introduced by a weak binarization algorithm. For example, a binarization algorithm that uses a global threshold can create unwanted artifacts on various parts of the image.

Another observation is that each character stroke has its darkest pixel values towards the heart of the stroke. The pixel intensities get lighter as it approaches the edge of the stroke. Typically, when characters are connected, only the tip of the edges touch. The overlapping pixels are significantly lighter than the pixels at the heart of the character. We use this property to our advantage in character segmentation. Figure 1 shows the difference in intensity levels between the center and the border of each stroke.

2.1. Smeared Character Detection

In order to segment characters properly, we first have to identify the class of characters that are merged and the class that is not. We perform this operation using the binary image. In the detection of merged characters, we apply a rule-based algorithm that takes advantage of inherent features of individual characters. We used connected component analysis as our preliminary step to segment characters.

Each connected component is analyzed to determine whether it consists of multiple characters and should be segmented based on the following heuristics:

1. If the component width is less than 1.1 times its height, then it is a single character and should not be segmented. It is a general character property for the majority of the English alphabet that the width of a character does not exceed its height, and therefore should not be segmented. A threshold of 1.1 is chosen to include the characters whose width and height are the same, such as the 'o'.

(a) (b) (c)

Figure 2. Examples of characters that should be segmented. Case (a) and (c) fails the loop analysis test, and (b) fails the 'm' and 'w' detection test.

2. If the component width is more than twice its height, then it should be segmented. There is no letter of the alphabet whose width is more than twice its height. The biggest width to height ratio are the 'm' and the 'w' written in certain fonts. This ratio is still lower than 2.

3. If a component has a width between 1.1 and 2 times its height, it is unknown whether it can be segmented. We apply an alternate set of rules for components that belong to this set: loop detection & analysis and 'm' & 'w' detection.

We take into consideration the number and location of loops of the connected component. A typical characteristic of the English alphabet is that no character consists of more than 2 loops. Two-loop characters have loops that are oriented vertically among each other. Characters with one loop consist of a loop that stretches horizontally across the center of the component. Examples of characters that can be segmented as determined by loop analysis are shown in Figure 2. Our smeared character detection algorithm using loop analysis is shown in Algorithm 1.

To do loop detection, we apply the following algorithm:

1. Pad the border of the binary image connected component with the background color.

2. Invert the binary image component.

3. Find the connected components.

4. The number of loops of the component equals the number of connected components-1 (Figure 3).

The only characters that have width between 1.1 and 2 times greater than their height are the 'm' and the 'w'. We detect 'm' and 'w' by using their unique characteristics rather than an OCR engine. One way to determine whether a character is a 'm' or a 'w' is to find the character's vertical symmetry [12]. If the character does not have vertical symmetry, the character can be segmented. However, if vertical symmetry holds, then the nature of the character is inconclusive. Figure 2 shows examples of characters that exhibit vertical symmetry but should still be segmented.

Algorithm 1: Smeared Character Detection

Input: $image$
Output: **boolean** segmentable/notSegmentable

```
loops = LoopDetection.apply(image)
if |loops| > 2 then
    return segmentable
end
if |loops| == 2 then
    if areHorizontal(loops) then
        return segmentable
    end
end
if |loops| == 1 then
    if leftOfCenter(loops) then
        return segmentable
    end
    if rightOfCenter(loops) then
        return segmentable
    end
end
return notSegmentable
```

(a) (b)

Figure 3. Loop Detection. (a) shows the original image, and (b) shows the inverted image. The number of loops is equal to one less than the total number of connected components.

Another feature for 'm' and 'w' detection is to find the number of horizontal and vertical crossings. The number of horizontal crossings H_c is defined by:

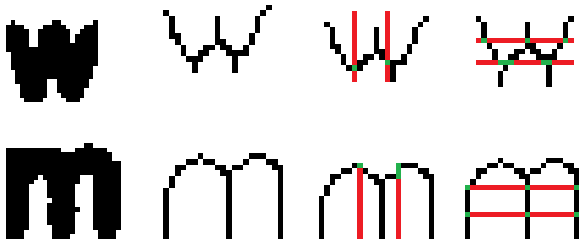
$$H_c = \sum_{x=0}^{w-1} cross(x, x+1)$$

where

$$cross(x, x+1) = \begin{cases} 1 & \text{, if } I(x,y) = 1 \text{ and } I(x+1,y) = 0 \\ 0 & \text{, otherwise} \end{cases}$$

w is the width of the image, y is the height at which the horizontal crossings are analyzed, x is the x coordinate of the pixel of interest, and $I(x, y)$ is the binary value at pixel location (x,y).

This method works in most cases; however, we noticed several instances that required us to alter our approach.



(a) (b) (c) (d)

Figure 4. The process used in 'm' and 'w' detection. (a) The binary image. (b) The thinned binary image. (c) Vertical crossings. (d) Horizontal crossings.

Since document images may be of poor quality, the smeared characters can be smeared to an extreme such that the 'm' and the 'w' exhibit far less horizontal crossings than expected. As shown in Figure 4 (a), the 'w' has only one horizontal crossing at its center. To counter this effect, we apply a thinning algorithm on the character image.

In the vertical crossings test, a vertical line is drawn from the top to the bottom of the image at increments of approximately 1/3 the image width. Exactly one crossing is allowed for both the 'm' and the 'w'. We apply a similar idea in the horizontal crossings test. Figure 4 shows the detection process.

2.2. Character Segmentation Using Shortest Path

Once we have established the set of smeared characters, the next step is to segment them. When the connected component is determined that it can be segmented, as in section 2.1, the corresponding component of the grayscale image is extracted. The grayscale intensity values are ranged from 0 - 255, with 0 as black and 255 as white. In order to incorporate grayscale intensities directly into our algorithm, we have to invert the image so that 0 is white and 255 is black. The motivation is that a path including a white pixel have cost 0 and the cost of a black pixel is the highest: 255.

In our shortest path algorithm, the goal is to find a minimum cost path that will start from the top of the component and end at the bottom of the component. We enforce the rule that the only valid next move of the path is in the forward direction. This ensures simple paths and avoids cycles. The shortest path algorithm is only applied to connected components that are classified as "smeared" from section 2.1. One segmentation path is constructed for each execution. To handle multiple smeared characters, the connected component will be analyzed and segmented in a recursive fashion.

Let the image be represented by a directed acyclic graph

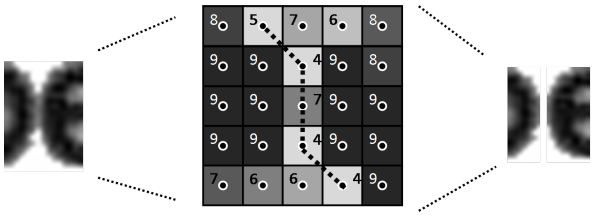


Figure 5. An image can be interpreted as a weighted directed acyclic graph. The weights are specified by the pixel intensity value, which for illustration purposes are scaled between 1 - 10 with 10 being the highest.

$G(V, E)$ where V is the set of all vertices and E is the set of all edges that connect the vertices. Each $v \in V$ can be represented as $v_{(x,y)}$ where (x, y) denote its x, and y coordinates relative to the top left corner of the image. Each edge $e \in E$ can be represented by the pair (u, v) where u and v are being connected by e. A vertex of the graph (also a node) represents a pixel of the image. Each vertex has a weight $W_{(x,y)} = I_{(x,y)}$, where $I_{(x,y)}$ is the intensity value for pixel (x,y) of the grayscale image. A vertex $v_{(x,y)}$ has outgoing edges to $v_{(x-1,y+1)}, v_{(x,y+1)}$ and , $v_{(x+1,y+1)}$ and incoming edges from $v_{(x-1,y-1)}, v_{(x,y-1)}$ and , $v_{(x+1,y-1)}$. Let the set $V_{(x,*)} = v_{(x,0)}, v_{(x,1)}, ..., v_{(x,h)}$, where h is the height of the image, and let $V_{(*,y)} = v_{(0,y)}, v_{(1,y)}, ..., v_{(w,y)}$, where w is the width of the image. Let $P(x, y)$ be the minimum cost from a $v \in V_{(*,0)}$ to $v_{(x,y)}$. Figure 5 shows a higher level overview of our shortest path algorithm.

Our shortest path algorithm takes a greedy approach similar to Dijkstra's shortest-path algorithm. Initially:

$$\forall\, v_{(x,y)} \in V_{(*,0)}, \qquad P_{(x,y)} = W_{(x,y)}$$

Then,

$$\forall\, v_{(x,y)} \in G, \qquad P_{(x,y)} = MIN(l_{x,y}, m_{x,y}, r_{x,y}) + W_{(x,y)})$$

where

$$l_{x,y} = \begin{cases} d * P_{(x-1,y-1)} & \text{, if } (x-1, y-1) \text{ is near center} \\ c * d * P_{(x-1,y-1)} & \text{, otherwise} \end{cases}$$

$$m_{x,y} = \begin{cases} P_{(x,y-1)} & \text{, if } (x, y-1) \text{ is near center} \\ c * P_{(x,y-1)} & \text{, otherwise} \end{cases}$$

$$r_{x,y} = \begin{cases} d * P_{(x+1,y-1)} & \text{, if } (x+1, y-1) \text{ is near center} \\ c * d * P_{(x+1,y-1)} & \text{, otherwise} \end{cases}$$

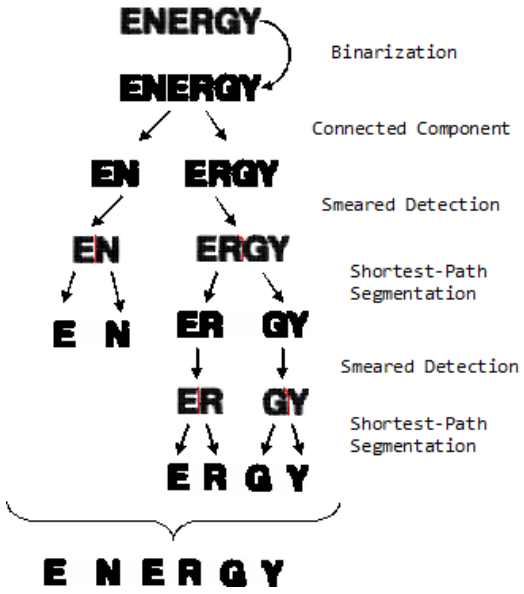


Figure 6. Our shortest path segmentation process for the word "ENERGY".

c and d are our penalty constants for off-centered and diagonals, respectively. Figure 6 shows the shortest path segmentation process.

There is an advantage for using the actual grayscale intensities as the weight of the points of the path. They are ideal because they are an integral part of the document itself. The intensity level is an indication of the pixel's significance to the character. Lower intensities have lower significance and higher intensities have a higher significance. The intensity values are lower as it approaches the edge of a stroke, therefore it is a preferable segmentation point.

We noted that in the majority of smeared characters the segmentation path lies mostly towards the center. We dictate the type of path and behavior of the search pattern by imposing fines on certain actions that deviate from the typical path pattern. To favor straight lines we impose a 10% fine on all diagonal path points. To ensure that each move is in a direction towards the center of the image, we impose a 20% fine for any path points going outside of a specified boundary. The penalty constant c and d are adjusted accordingly.

3. Experiments and Results

We have tested our method on 555 words with nearly 2,000 smeared characters spanning from a corpus of 13 document images. Each word may consists of two or more smeared characters. The set of images are diverse in style, but with similar fonts. The forms may contain check boxes,

blank lines, logos and horizontal and vertical lines. The quality level of each form varies, so the testing set provides several different levels of smearedness. We only conduct our experiment on typewritten words.

Each form contains both words and numbers. Our algorithm primarily was designed to segment letters of a word, but it was found to be quite effective in segmenting digits as well. Our algorithm is also competent in segmenting thin characters. Since characters with thin strokes have less disparity between gray level intensities, the shortest path may sometimes not be identified correctly due to the existence of various different paths of similar costs. Adjustments made to the penalty constant allowed us to correctly handle this issue.

Our metric for statistical analysis is based on the following: The total number of splits (T) required for each word is determined by visual inspection. The number of splits (S) found by our algorithm is recorded. The number of correct splits (C) is then determined visually. Overall, our accuracy per split (C/T) is 94.71%. Our accuracy of detecting merge characters is (S/T) 77.98% (Table 1).

Table 1. Overall Statistics

Form Name	Smeared Char's (T)	Total Split (S)	Correct Splits (C)	Seg. Acc. (C/S)	Smeared Det. (S/T)
1000	121	100	96	96.00%	82.64%
2000	2	3	1	33.33%	150.00%
3000	5	6	3	50.00%	120.00%
4000	1	1	0	0.00%	100.00%
5000	1	1	1	100.00%	100.00%
7000	5	4	4	100.00%	80.00%
8000	3	3	3	100.00%	100.00%
9000	6	6	6	100.00%	100.00%
10000	430	350	343	98.00%	81.40%
11000	316	259	255	98.46%	81.96%
12000	35	30	30	100.00%	85.71%
13000	869	636	583	91.67%	73.19%
TOTAL	**1,794**	**1,399**	**1,325**	**94.71%**	**77.98%**

Our algorithm deteriorates as it is applied to poorer quality images which consist of heavily smeared characters. Analysis of the grayscale image provides little to no advantage over its binary counter part. This defeats the purpose of our algorithm. For this set of smeared words, our C/T is 91.67%, and our S/T is 73.19%. As expected, the poor quality image achieved statistical results below average. On light to moderately smeared characters, our algorithm was able to segment characters with multiple connections in a recursive fashion. The results were significantly better than the overall average, and the heavily smeared characters. The C/T achieved here is 97.25%, and S/T is 82.49%. Figure 7 shows the result of our segmentation algorithm on different types of smeared characters.

Our results clearly indicate that most of our segmentation

Figure 7. Different types of smeared characters. (a)-(c) are heavily smeared. (d)-(f) are moderately smeared. (g)-(i) are eroded characters. (j)-(l) are smeared digits.

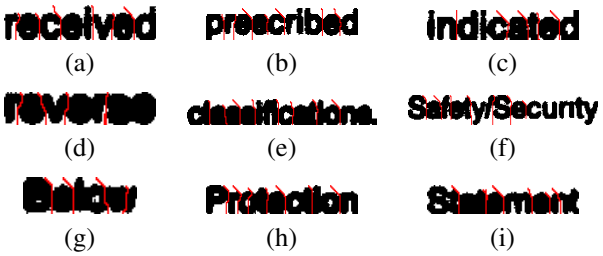
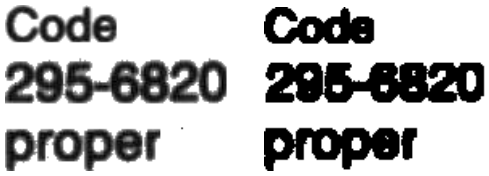


Figure 8. Different types of errors of our segmentation algorithm. (a)-(c) are undetected smeared connections with thin characters such as 'i's and 'l's. (d)-(f) are undetected smeared characters due to its poor quality and over-smearedness. (g)-(i) are missegmentations due to overly smeared characters.

errors arise from characters that are smeared but were not detected to be smeared. Our smeared detection algorithm is particularly weak at deciding smeared connections whose width and height ratio is consistent with general character properties. Figure 8 shows the different types of errors we encountered.

Of the set of characters that were detected to be smeared but were segmented improperly, most of them were heavily smeared. Figure 8 (j)-(l) show examples of segmentations in this category. Segmentation of characters smeared to the degree as in figure 8 is extremely difficult. A holistic approach is more suitable to handle smeared characters in this category.



(a) (b)

Figure 9. (a) words that are in grayscale. (b) the corresponding words in binary. The conversion from grayscale to binary may introduce extra smeared connections.

4. Conclusion

This paper proposed a non-linear technique to perform character segmentation. Our method first determines whether a connected component is a smeared character. It utilizes data from grayscale pixel values to determine a reliable segmentation path.

Our algorithm operates on the grayscale image, where, in comparison with the binary image, provides a more accurate representation of the original image (Figure 9). The majority of existing segmentation algorithms are performed in the binary domain ([4], [8], [10] [11], [13]).

Another advantage is that it does not require an OCR to verify its segments, and a segmentation path can be found in one iteration. Using an OCR slows the segmentation process. Many existing works elicit the help of a recognizer to verify the segmentation paths ([7], [8], [11], [13]).

In our future work, we will explore simple methods of character segmentation in words that are heavily smeared. We will also investigate improving smeared character detection using methods that are independent of the font.

5. Acknowledgment

Special thanks extended to Dr. Stephen Rice, Vice President of Research at UNLV. This project is supported by the Department of Energy, #2362-272-76FJ. This work performed in collaboration with Apogen Technologies.

References

[1] N. Arica and F. Yarman-Vural, "Optical Character Recognition for Cursive Handwriting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v.24, n.6, June 2002.

[2] R. Casey and E. Lecolinet, "A Survey of Methods and Strategies in Character Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v.18, n.7, 1996.

[3] C. Chen, and J. DeCurtins, "Word Recognition in a Segmentation-Free Approach to OCR," *Proceedings of the Second International Conference on Document Analysis and Recognition*, 1993.

[4] R. Hoffman and J. McCullough, "Segmentation Methods for Recognition of Machine-Printed Characters," *IBM Journal of Research and Development*, 1971.

[5] C. Jones, V. Kubushyn, J. Bunch, D. Curtis, E.A. Yfantis, "Grouping, Segmentation and Recognition of Handwritten Social Security Images," *International Conference on Image Processing, Computer Vision, and Pattern Recognition*, 2007.

[6] S. Lee, D. Lee, and H. Park, "A New Methodology for Gray-Scale Character Segmentation and Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v.18, n.10, October 1996.

[7] S. Liang, M. Shridhar, and M. Ahmadi, "Efficient Algorithms For Segmentation and Recognition of Printed Characters in Document Processing," *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, 1993.

[8] S. Messelodi and C. Modena, "Context Driven Text Segmentation and Recognition," *Pattern Recognition Letters*, v.17, n.1, 1996.

[9] T. Nartker, *ISRI 1992 Annual Report*, Univ. of Nevada, Las Vegas, 1992.

[10] B. Plessis, A. Sicsu, L. Heutte, E. Menu, E. Lecolinet, O. Debon, and J. Moreau, "A Multi-Classifier Combination Strategy for the Recognition of Handwritten Cursive Words," *Proceedings of the Second International Conference on Document Analysis and Recognition*, 1993.

[11] J. Song, Z. Li, M. Lyu, and S. Cai, "Recognition of Merged Characters Based on Forepart Prediction, Necessity-Sufficiency Matching, and Character-Adaptive Masking," *IEEE Transactions on Systems, Man, and Cybernetics: Part B*, v.35, n.1, 2005.

[12] J. Tse, D. Curtis, J. Bunch, C. Jones, E.A. Yfantis, and A. Thomas, "Handwritten and Typewritten Word and Character Separation in Unconstrained Document Images," *International Conference on Image Processing, Computer Vision, and Pattern Recognition*, 2007.

[13] J. Wang and J. Jean, "Segmentation of Merged Characters by Neural Networks and Shortest-Path," *Pattern Recognition*, v.27, n.5, 1994.