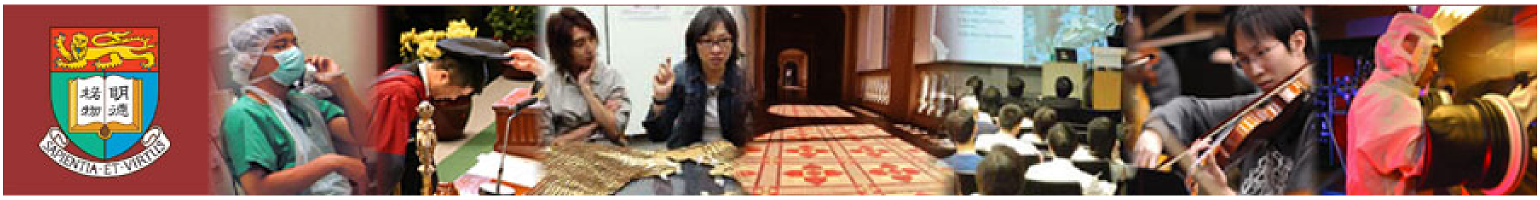


Underline Detection and Removal in a Document Image Using Multiple Strategies

Zhen-Long BAI and Qiang HUO
Department of Computer Science and Information Systems,
The University of Hong Kong, Pokfulam Road, Hong Kong, China
(Email: zlbai@csis.hku.hk, qhuo@csis.hku.hk)

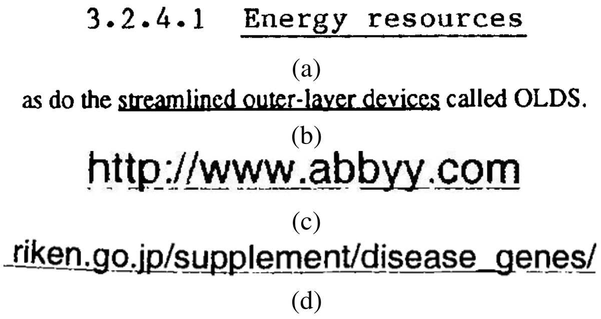
Abstract

This paper presents a novel three-module approach for underline detection and removal in Chinese/English OCR. The detection module uses strategies of connected component analysis and bottom edge analysis. The removal module uses different methods for different kinds of underlines. The disambiguation module is effected via recognition confidence comparison for reducing the risk of removing wrongly doubtful underlines. Our approach can deal with untouched, touched, broken and slightly curved underlines. In a benchmark test using single text line images extracted from UW-I database and images captured by C-Pen, we demonstrate that our approach has little negative effect on pure-text images, and can detect and remove reliably underlines in text line images with underlines.

1. Introduction

Underlines are frequently used in many documents. The work reported in this paper deals with the problem of how to detect and remove possible underlines in a printed document image for improving OCR performance. Some examples of single text line images with underlines are illustrated in Fig.1, where the images in Figs.1(a)(b) are from UW-I database [3], and images in Figs.1(c)(d) are captured by a pen scanner product called C-Pen 10 [5]. In the simplest case, the underline in Fig.1(a) is not touched with any text part, and is called an *untouched underline*. The underlines in Figs.1(b)(c)(d) are touched with some characters in text lines, thus are named as *touched underlines*. The underlines in Figs.1(c)(d) are broken into several parts, thus are called *broken underlines*. The underline in Fig.1(d) is slightly curved, and is referred to as a *curved underline*.

By playing with one of the best commercial OCR products on the market, namely FineReader 7.0 [4], we observed that 1) it can remove most of the touched, broken and curved underlines; and 2) it does not work very well for difficult cases as in Figs.1(b)(d), where there exist some black pixels under the underline, or an underline is both curved and broken. Although good techniques obviously have been developed for underline removal in OCR by some leading companies, unfortunately, they have never been published in open literature. We can only find a few works in literature that are somehow related to the topic. One example is the work reported in [6], in which a technique for removing straight frame lines from a form is described. As part of efforts to develop a pen scanner based Chinese/English OCR system (e.g., [2]), we developed our own solution for underline removal. In the following, we describe in detail how our approach works, and report the result of a benchmark test to demonstrate the effectiveness of our approach.



3.2.4.1 Energy resources

(a)

as do the streamlined outer-layer devices called OLDS.

(b)

http://www.abbyy.com

(c)

riken.go.jp/supplement/disease_genes/

(d)

Figure 1. Examples of single text line images with underlines

2. Our Approach

The overall architecture of our approach is shown as in Fig.2. Given a binary image of a text line hypothesized by the layout analysis module, our approach works as follows: First, an underline detection module is applied to tell whether there exists any underline; Then, an underline removal module is used to remove different kinds of under-

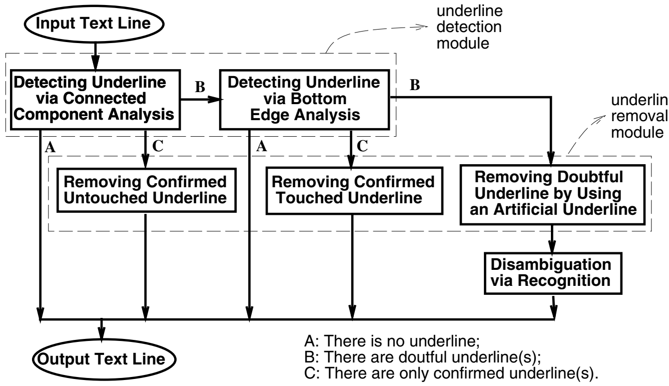


Figure 2. The overall architecture of our approach for underline detection and removal.

lines; Finally, a disambiguation module is used to reduce the risk of removing wrongly doubtful underlines. In the following, we elaborate on details of each block in Fig.2.

2.1. Underline Detection Module

Usually, an underline is with long length, small height, and at the bottom of a text line. Based on these geometric features, two strategies are developed for underline detection, where 1) connected component (CC) analysis is for detecting confirmed untouched underline(s) and doubtful underline(s), and 2) bottom edge analysis is for detecting confirmed touched underline(s). For each detection step, the input is a single text line, while the output has following three options: (A) If there is no underline, we output the text line directly; (B) If doubtful underlines are detected in the first detection module, the text line image is forwarded to the second detection module for further detection by using a different strategy. If doubtful underlines remain, the text line image is forwarded further to an underline removal module by using a strategy of an artificial underline (to be explained later); (C) If there are only confirmed underlines, we remove them accordingly.

2.1.1 Detection via Connected Component Analysis

Before listing detection rules, some definitions are given first. For a CC, C_i, parameters $w_c(i)$, $h_c(i)$, $t_c(i)$, $b_c(i)$ are defined to be its bounding box's width, height, vertical positions of top and bottom edges, respectively. For a text line, y_{mid} is defined to be the vertical middle position of the whole text line's bounding box. Text line height h_p is estimated by a moving window approach as follows: We segment horizontally the text line into n equal parts, with $n = max(4, L_w/L_h)$, where L_w and L_h are whole text line bounding box's width and height respectively; If we sort heights of those n blocks in descending order, then h_p is set to be the jth height, where $j = 30\%n$. Such an estimation for h_p is robust for curved underlines. After all CCs in the text line are obtained, following heuristic rules are used for making a decision:

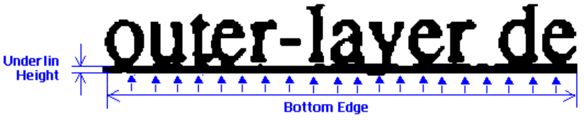


Figure 3. Underline detection via bottom edge analysis.

- If $h_c(i) \leq \alpha_1 \cdot h_p$ and $t_c(i) > y_T$ and $w_c(i) \geq \alpha_2 \cdot h_p$ and $w_c(i) \geq \alpha_3 \cdot h_c$, we label the C_i as
  - a confirmed untouched underline, if thresholds are set as $\alpha_2 = \frac{3}{2}$,$\alpha_3 = 5$; or
  - a doubtful untouched underline, if thresholds are set as $\alpha_2 = \frac{1}{2}$,$\alpha_3 = 2$.

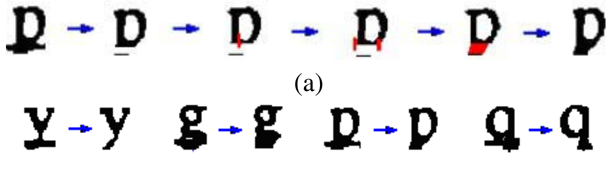
  In both cases, we set $\alpha_1 = \frac{1}{4}$, $y_T = \frac{1}{4}h_p + y_{mid}$;
- If $w_c(i) \geq \alpha_4 \cdot h_p$ and $b_c(i) > y_T$, we label the C_i as a doubtful touched underline. Here we set $\alpha_4 = 2$. y_T is the same as the above setting;
- Otherwise, there is no underline in the text line.

2.1.2 Detecting Underline(s) via Bottom Edge Analysis

In this step, another detection strategy is used and illustrated by example shown in Fig.3. Let's use $y_b(i)$ to denote a black pixel's vertical position along the bottom edge. If there is no black pixel at a horizontal position $x = i$, we set $y_b(i) = 0$. For a block of m contiguous black pixels on the bottom edge, if $m > \alpha_2 h_p$ and $y_b(i) > y_T$ for all i in this block, then a confirmed touched underline is detected. Thresholds are set as $\alpha_2 = \frac{3}{2}$, and y_T is the same as in previous setting. Decisions for this step are made as follows:

- If only doubtful touched underlines are detected in the first detection module via CC analysis, and
  - none of them is confirmed to be a touched underline, then no underline exists; or
  - all of them are confirmed to be touched underlines, then only confirmed touched underlines exist in the input text line;
- If some doubtful untouched underlines are detected in the first detection module via CC analysis, the decision remains the same in this step.

No decision is made on the existence of doubtful touched underlines because of the great ambiguity brought by some difficult cases, especially for Chinese characters.



(a)

(b)

Figure 4. Removing touched underline for special letters.

2.2. Underline Removal Module

The previously discussed underline detection strategies also suggest some ways for underline removal. For different kinds of underlines in Fig.1: 1) An untouched underline can be detected and confirmed via CC analysis, thus it can be removed simply by deleting the CC; 2) A touched underline can be detected and confirmed by an elaborate analysis of bottom edge, that also facilitates greatly the removal of touched underline(s); 3) A broken underline is much more difficult for both detection and removal. Fortunately, it usually includes some doubtful untouched underlines, although sometimes it also includes some confirmed touched underlines. Therefore, a broken underline can be detected via detecting some doubtful untouched underlines. For broken underline removal, using an artificial underline can transform the problem into removing an untouched or a touched underline, as shown in Fig.5; 4) A curved underline may accompany one of the above kinds of underlines. No additional strategy is needed for untouched underline. As for touched underline, using a robust estimation of text line height h_p and local bottom edge values $y_b(i)$ is effective for solving the problem. In the following, we fully discuss above strategies that need more explanations.

2.2.1 Removing Confirmed Touched Underline(s)

Underline's height, as shown in Fig.3, which is important for touched underline removal, is estimated as follows: First, an array of $y_b(i) - y_t(i)$ is obtained, where $y_t(i)$ is the vertical position of the first white pixel by tracing upward from the bottom edge at the horizontal position $x = i$. Then, we sort the array in descending order according to its elements' values. If the array length is n, then the value of kth element in the sorted array is used as an estimation of the underline's height h_u, where $k = 80\%n$.

After h_u is estimated, we use a new estimation $\tilde{y_b}(i) = median(y_b(j))$ in the local area: $max(0, i - h_p) \leq j \leq min(i+h_p, l_u)$, to replace the original value of $y_b(i)$, where l_u is the underline's length. It not only helps smooth some noises, but also offers a better estimation of underline's true bottom edge for some difficult cases such as letters with black pixels below the underline as shown in Fig.4.

http://www.abbyy.com

Figure 5. Using an artificial underline for the broken underline in Fig.1c.

In removing a touched underline, black pixels of characters in character-underline crossing areas should be kept. Our procedure to achieve this is as follows (cf., Fig.4a): 1) Mark a confirmed touched underline within the horizontal band between $\tilde{y_b}(i) + 1$ and $\tilde{y_b}(i) - h_u - 1$; 2) For each black-pixel component (if any) below the marked underline, if its width w_l satisfies the condition $T_1 \leq w_l \leq T_2$, where $T_1 = 2$, $T_2 = \frac{1}{3}h_p$, it is regarded as the low part of a character, thus is kept; otherwise, it is deleted; 3) Mark the crossing area from each remaining black-pixel component below the marked underline to its nearest unmarked upper neighbor CC. Form an echelon to circumscribe each of such marked crossing area(s); 4) If marked pixels in the echelon are originally black, they are kept. All other black pixels in above marked areas are removed accordingly.

Results of some touched underline removal examples are shown in Fig.4b and Fig.6b.

2.2.2 Removing Doubtful Underline(s) by Using an Artificial Underline

In this step, all confirmed and doubtful underlines are first marked. If there are more than N_D (here we set $N_D = 2$) doubtful underlines, we regard a global broken underline exists. An artificial underline is then constructed to facilitate the removal of such a broken underline (cf. Fig.1c and Fig.5). The height of the artificial underline is decided by the longest marked underline. In an unmarked position, its local $\tilde{y_b}$ is decided by the value of the nearest marked position. After the construction of the above artificial underline, it can be removed either as an untouched underline or as a touched underline. In this global artificial underline, if some confirmed underlines exist, they are removed by previous strategies for confirmed underlines, while other parts are removed as a part of the artificial underline.

2.3. Disambiguation via Recognition

To reduce the risk of a wrong underline removal, before removing an artificial underline, we use our OCR engine to recognize the input text line L_1 (If there are some confirmed underlines in the artificial underline concerned, L_1 is the text line after confirmed underlines are removed), and obtain a text line confidence measure CM_1 [1]. After removing the artificial underline, we can get a new text line L_2, and a new text line confidence measure CM_2. If $CM_2 > \gamma \cdot CM_1$ (here we set $\gamma = 1$), which means the text

Table 1. Benchmark test results for pure-text images without any underlines.

Image Types	No Underline Detected	Underline Detected	Wrongly Removed
C-Pen (827)	85.0% (703)	15.0% (124)	0.4% (3)
UW-I (1016)	84.3% (856)	15.7% (160)	0.5% (5)
Total (1843)	84.6% (1559)	15.4% (284)	0.4% (8)

Table 2. Benchmark test results for images with some underlines.

Underlines Types	Image Types	Underline Detected	Correctly Removed
Untouched	UW-I (62)	100% (62)	98.4% (61)
Touched	UW-I (90)	100% (90)	94.4% (85)
Broken	C-Pen (87)	100% (87)	92.0% (80)
Total	(239)	100% (239)	94.6% (226)

line after underline removal can be recognized more confidently by our OCR engine, we regard L_2 is what we want; otherwise, we keep the original text line L_1.

3. Experiments and Results

To verify the effectiveness of our approach, a benchmark test is performed. All testing images are single text line images, either extracted from UW-I database or obtained by C-Pen. The setting of control parameters described in previous sections is used in all experiments. Our benchmark test results are summarized in Tables 1 and 2, where numbers in $(\cdot)$ indicate the absolute number of relevant images. The judgement of the correctness of underline removal is made by authors' visual inspection of the processed images. From Table 1, it is observed that although our approach has a false alarm rate of 15.4%, only 0.4% of pure-text images are wrongly output with some pixels being falsely removed as underline(s). This demonstrates that our approach has little negative effect on pure-text images. From Table 2, it is observed that for images with underlines, all underlines are correctly detected, and 94.6% of them have their underlines correctly removed. This confirms the effectiveness of our approach for underline detection and removal. Note that the benchmark results for C-Pen images (including both English and Chinese scripts) are obtained by using a more complicated system described in a companion paper [2], where the underline detection and removal modules presented in this paper are only parts of that system.

Fig.6 shows results of underline removal for images in Fig.1 by using our approach. All underlines are successfully removed. However, FineReader 7.0 [4] gets wrong results for images in Figs.1(b)(d). For the image in Fig.1b, its recognition result is "streamlined outer-laver devices called OLDS.", which implies that the pixels of 'y' under the underline might be wrongly removed. For the image in Fig.1d, its recognition result is "Jjken,cio,jp/supplement/disease genes/", which implies that the left part of the curved broken underline might be wrongly segmented.

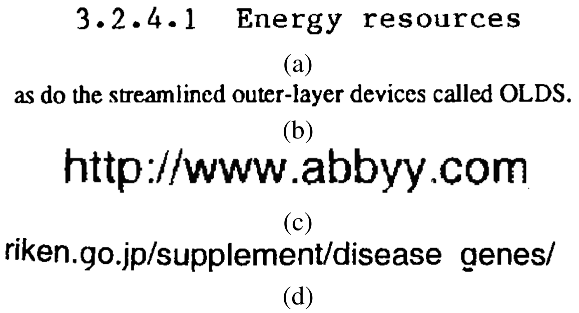

3.2.4.1 Energy resources
(a)
as do the streamlined outer-layer devices called OLDS.
(b)
http://www.abbyy.com
(c)
riken.go.jp/supplement/disease genes/
(d)

Figure 6. Results of underline removal for images in Fig.1.

In spite of the above, the overall performance of our approach is slightly worse than that achieved by FineReader 7.0 for pure English text lines with underlines. Because our OCR engine has to deal with mixed Chinese/English scripts, the setting of relevant thresholds is not optimized for English only scripts.

4. Conclusion

In this paper, we present a novel approach for underline detection and removal, that has been confirmed to work for dealing with untouched, touched, broken, and slightly curved underlines. As future works, we will study some more intelligent ways of setting relevant thresholds, develop better strategies for dealing with broken and doubtful underlines, improve the disambiguation module, and investigate how to do an automatic benchmark testing.

References

[1] Z. L. Bai and Q. Huo, "An approach to extracting the target text line from a document image captured by a pen scanner," in *Proc. ICDAR-2003*, Edinburgh, August 2003, pp.I-76-80.

[2] Z. L. Bai and Q. Huo, "A goal-oriented verification-based approach for target text line extraction from a document image captured by a pen scanner," submitted to *ICPR-2004*.

[3] S. Chen, M. Y. Jaisimha, J. Ha, R. M. Haralick, and I. T. Phillips, "Reference Manual for UW English Document Image Database I," University of Washington, August 1993.

[4] http://www.abbyy.com

[5] http://www.cpen.com/

[6] B. Yu and A. K. Jain, "A generic system for form dropout," *IEEE Trans. on PAMI*, Vol. 18, No. 11, pp.1127-1134, 1996.