# Character Extraction by Integrating Color into Edge-based Methods

Naoki Chiba
Rakuten, Inc.
4-13-9 Higashishinagawa, Shinagawa-ku,
Tokyo, 140-0002, JAPAN
naoki.a.chiba@mail.rakuten.com

Xinhao Liu
Tokyo Institute of Technology
2-12-1 O-okayama, Meguro-ku,
Tokyo, 152-8550, JAPAN
liuxinhao@ok.ctrl.titech.ac.jp

## Abstract

*Text recognition is difficult in e-commerce images, which contain a mixture of digital born text and natural scene text. To extract character regions in an image, a method using the consistency of the stroke widths of a character, called stroke width transform (SWT), is promising because of its simplicity and low computation. The method is to measure stroke widths after detecting edges. On the other hand, character extraction based on color clustering has been investigated separately. This paper presents algorithms for extending the SWT method by integrating color-clustering information. Our experimental results show the effectiveness of the proposed algorithms both with digital-born and natural-scene images.*

## 1 Introduction

Images are frequently used to embed textual information. As the number of mobile cameras and the usage of images on the Internet increase, the need for information retrieval from images has been growing rapidly. One application is to recognize text on product photos for e-commerce, especially for e-marketplaces. The images contain both text of the product, such as product model or maker names, and text added by sellers using an editing tool for promotion such as prices or shipping information. Text in these images is a mixture of digital born text and natural scene text. The performance of conventional optical character recognition (OCR) is insufficient for the current applications.

Text recognition research can be divided into two categories: text localization (also called text detection) and word recognition. Text localization means to detect rectangular regions of words in an image. Word recognition means to recognize a word from the cropped rectangle of a word region in an image. Both categories require character extraction as a module in a system.

For extracting character regions, a method called stroke width transform (SWT) [2] is gathering attention because of its simplicity and computational efficiency. Figure 1 shows an example. This method is used both for text localization and word recognition.

SWT, however, has the following two problems. First, it cannot determine foreground and background. Figure 1 (b) shows an example. It can correctly extract foreground text but falsely extracts background text candidates. Second, when a character has sharp corners or stroke joints, SWT fails to extract them. Figure 2 shows an example. These problems affect the performance significantly in the later stages.

To solve these problems, we propose algorithms for integrating color-clustering information into the SWT method. Unlike previous methods such as simply using the mean color value of an extracted region, the proposed algorithms integrate extracted regions by using color clustering into those by using edge detection. We explain how we integrate these two extracted results.

### 1.1 Related work

Character extraction is a module to extract character regions in a form of either a group of connected pixels or a bounding box. This module plays an important role in a system both for text localization and word recognition. For text localization, characters are extracted first. Then characters aligned on a line are grouped, followed by word separation. In word recognition, a cropped rectangular word-region contains multiple characters. We need to extract characters first to recognize characters followed by word recognition.

Many techniques have been proposed for character extraction. These techniques can be categorized into three types: binarization [10, 9], segmentation and classification [12]. The segmentation category can be further divided into two subcategories: edge-based [15, 14, 7] and color-based [11, 14] methods. Binarization and segmentation methods extract connected components (CC) followed by CC analysis. Binarization and edge-based methods are fast to compute, but their precisions are lower than that of color-based method. Classification has higher precision but is slow to compute because it requires sliding windows across an image. Most text localization and recognition systems have combinations of these techniques.

SWT is an edge-based method of segmentation that is simple, computationally efficient, and relatively accurate. It has been applied to many text localizing methods [2, 13, 5, 6, 14, 3], text recognition [8], and even a popular open source software from libccv [1]. The original SWT method [2] was combined with post processing such as character grouping and word separation for text localization that requires parameter tuning. Yi and Tian presented a similar method with SWT [14] . Yao et al. [13] extended the SWT method to avoid the tuning by combining classifiers.

On the other hand, color-based methods, which cluster pixels in color, have also been proposed [11, 14]. Yi and Tian [14] reported that the edge-based method is faster than the color-based method, but the color-based method is more accurate.

To incorporate colors with the SWT method, the original work [2] simply computed the mean color value

[1] http://libccv.org/


(a) Original (b) Result with regular SWT (c) Result with our color integrated SWT

Figure 1. Comparison between results of regular SWT and our color integrated SWT


Figure 2. SWT's problem (left shows original image; right shows extracted result with character missing joints and corners)

of each character for grouping characters. Meng et al. [6] used color partially by adding the representative color to each edge segment of a character. Because these methods add color *after* extracting a character, they do not solve the above mentioned two problems. Huang et al. [3] measured color variation while detecting a character. Their method may deal with the second problem of missing corners or joints, but the statistical performance of this module was not measured. Furthermore it does not solve the first problem of brightness ambiguity. None of the methods uses color clustering information.

To solve the second problem of missing corners or joints, Yi and Tian [14] suggested applying morphological filters, but the effectiveness was not discussed adequately. The first problem of ambiguity in brightness patterns between foreground text and backgrounds has never been discussed before.

## 2 Color integrated SWT

### 2.1 Stroke Width Transform (SWT) - Overview

SWT comes from the observation that text has similar stroke widths [2]. It finds edge pairs on the basis of edge directions after using the Canny edge detector. Starting from an extracted edge pixel, it finds a pairing edge pixel that has the opposite direction, within a pre-determined allowable angle difference ($\frac{\pi}{6}$) with a similar edge strength, examining pixels along the direction like ray-tracing. By recording the width not only on the edge pixel but also on the pixels along the ray, it transforms an image into a stroke width map that has stroke width values in pixels. This enables us to use conventional image processing techniques such connected component analysis. This operation is conducted twice for two different brightness patterns. The next step is to group pixels for character candidates by using conventional connected-components labeling.

### 2.2 Color clustering for character extraction

Since many text strings both in digital-born images and natural-scene images have uniform color, clustering color pixel values effectively extracts text regions. Our objective is to retrieve dominant colors while reducing the number of colors. Our color clustering is based on the algorithm proposed by Senda et al. [11]. We describe the method briefly in this section.

It consists of three steps:

1. Color histogram
2. Color cluster merging
3. Color segmentation.

The first step is to make a histogram in a reduced color space that has 6 bits per RGB color channel. Each histogram bin corresponds to a cluster.

The second step is to merge similar color clusters into a representative cluster by measuring the distance of clusters.

### 2.3 Integration of color into edge-based methods

While SWT tends to miss joints or corners, color clustering can preserve them. Therefore, to enhance SWT components, we propose an algorithm that unifies one component extracted by SWT (SWT component) and another by color-clustering (color component). In preparation, we record the position and size of the bounding box for each connected component as well as its small image corresponding to the bounding box. An SWT component stores a stroke width map in the small image while a color component stores a color index number as a pixel value. We consider a pair of components that overlap in the images.

The following formula represents this image operation of a union between two extracted images:

$$S = E \cup C \quad s(x,y) = \begin{cases} e(x,y) & \text{if } e(x,y) > 0 \\ w_m & \text{else if } c(x,y) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $S$ represents the union component, $E$ represents an SWT component, $e$ represents its stroke width map, $C$ represents a color component, $c$ represents its color index image, $s$ is a stroke width map of union, $x, y$ are image coordinates, and $w_m$ is the mean value of the stroke widths of $E$. Since the image sizes of two bounding boxes can be different, we consider a rectangular image that includes both bounding boxes.

This operation, however, can be applied only to the ideal case when both color and edge components mostly overlap for a character. In real images, there are many exceptions. For example, when a few character obtained by SWT are merged into one caused by low image resolution, corresponding color components are not always merged. We thus need to handle the one-to-many matching problem.

### 2.4 One-to-many matching

To handle one-to-many matching, we obtain a union again across different color components that overlap the SWT component of interest. The integrated component $U$ is represented by the following equation

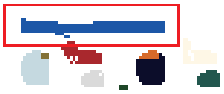
(a) Original

(b) SWT segments of bright text

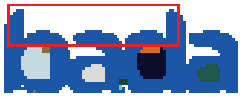
(c) SWT segments of dark text

(d) SWT segments combined

Figure 3. Brightness pattern problem



(a) SWT component (blue region)



(b) Color component (white region)

Figure 4. Edge component and color component

$$U = \coprod_{i=1}^{n} S_i \tag{2}$$

where $n$ represents the number of overlapping color components and $S_i$ is a union between an edge component $E$ and a color component $C_i$, represented by the formula

$$S_i = E \cup C_i. \tag{3}$$

We repeat this operation for all of the connected SWT components.

### 2.5 Background component elimination

Integrating color into SWT can solve not only the missing joint-corner problem but also the opposite brightness pattern problem. Figure 3 shows an example. As we can see in (c), SWT detects regions falsely between characters because we apply SWT twice for two different character brightness patterns as described by Epshtein et al. [2]. We explain how we solve this problem by integrating color.

The characteristic of falsely detected SWT components on the background is that they are mostly overlapped, while the areas of color components are larger than those of intersecting pixels because color components correspond to the background. If we expand the region of interest, the area of a color component increases. Figure 4 shows an example. By comparing areas between the intersecting pixels and the corresponding color component, we can identify if the SWT component is on the background or foreground text. The intersecting area in pixels can be obtained by the following INTERSECTION image operation.

$$T = E \cap C \quad t(x,y) = \begin{cases} e(x,y) & \text{if } e(x,y) > 0 \text{ and } c(x,y) > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

where $T$ represents the intersection component, $E$ represents an SWT component, $e$ represents its stroke width map of an SWT, $C$ represents a color component, $c$ represents its color index map, $t$ is a map of intersection, and $x, y$ are image coordinates.

We obtain the number of pixels (area) in the intersection component as $A_t$ by counting non-zero pixels in the intersection component $T$. When we count the number of pixels of the color component as $A_c$, we can define the ratio $R$ between these two areas as:

$$R = \frac{A_t}{A_c}. \tag{5}$$

If the ratio $R$ is smaller than a pre-defined threshold $R_t$, we determine that the SWT component is in the background.

Again, this is an ideal case for one-to-one matching. To handle multiple components, we sort overlapping color components in descending order in terms of the number of intersecting pixels. Comparing with the largest color component, if $R$ is smaller than $R_t$, we eliminate the SWT component as the background.

This operation can be combined with the previous operation of enhancing SWT components in 2.3 and 2.4. The procedure can be summarized as follows

1. Choose one SWT component $e$
2. Find all color components that overlap the SWT component $e$
3. Sort them in terms of the number of intersecting pixels $A_t$
4. Find the color component that has the most intersecting pixels $A_c$
5. Compute the ratio $R$ for the SWT component
6. If $R$ is smaller than the pre-defined threshold $R_t$ then discard it as the background and finish for $e$
7. Otherwise, enhance it and compare another color component to enhance $e$
8. Repeat until no overlapping color components are left.

We repeat these steps for all of the SWT components.

## 3 Experimental results

The ICDAR dataset [4] was used for evaluation. Figure 1 shows an example of the results with digital-born images. The last letter "y" is enhanced while components between characters were correctly eliminated. Figure 5 shows an example of the results with natural scene images. False character candidates between characters were correctly eliminated in 5(c).

We measured the performance statistically on digital-born and natural-scene test images, consisting of 141 and 233 images respectively. We used character segmentation data of the test set with ground-truth. Our intention is to measure the ability of extracting character candidates as a module in a text localization or word recognition system. For this reason, the results may not be as good as those of others that are combined with other techniques such as classifiers in the ICDAR competition. In addition, our focus is on comparing the difference between the original SWT and color integrated SWT.

Table 1 shows the result. The f-score for digital-born images significantly improved from 16.2% to 53.2%, and that for the natural scene increased from 14.2 % to


(a) Original (b) SWT

(c) Proposed

Figure 5. Result comparison of a scene image

46.4 %. The performance was measured by the atom-based framework (Recall, Precision and F-score) proposed by Clavelli et al. [1], which considers character structure rather than simply counting overlapping pixels.

In the experiment, the threshold of $R_t$ was set to 0.8. Color clustering was conducted only in a portion of the image around the detected SWT components, which was expanded 50% both for the height and width of the bounding box of a component to form a rectangular sub-image. The threshold of color distance in clustering was 200 in RGB. We used three rays in computing SWT, by adding two rays at ± 30 degrees. We set the maximum stroke width to 70 pixels.

| Data | Methods | Recall | Precision | F-score |
|---|---|---|---|---|
| Digital born | SWT | 11.8% | 26.0% | 16.2% |
| | Proposed | 45.0% | 65.2% | 53.2% |
| | Morphology | 8.5% | 13.6% | 10.4% |
| Natural scene | SWT | 13.0% | 15.2% | 14.2% |
| | Proposed | 44.8% | 48.2% | 46.4% |
| | Morphology | 14.4% | 11.5% | 12.8% |

Table 1. Comparison with ICDAR 2013 images

We also compared the difference against the improvement by using the morphological operation [14]. The performance decreased due to enhancement of false candidates in the background.

### 3.1 Discussions

Integrating color clustering with an edge-based method of SWT can improved the performance of character extraction. Although simple union and intersection pixel operations can be used, multiple segment handling is necessary because an edge-based component does not correspond exactly to a color-based component. We described algorithms that handle this situation to enhance SWT results. This provides more accurate results than before for computing image features. In addition, these algorithms can eliminate (or assign low confidence to) falsely detected SWT components in the background.

The computational cost is lower than that of traditional color clustering methods because color clustering is conducted only in surrounding areas of components extracted by SWT rather than in the entire image. The processing speed is in the middle between edge-based and color-based methods.

In the future, we plan to embed this character extraction to a text localizing or word recognition system.

## 4 Conclusions

We proposed algorithms for integrating color information into edge-based character extraction. Doing so can fill the holes or missing corners of a character caused by the imperfection of SWT. In addition, it can eliminate falsely extracted components in the background. Experiments on ICDAR 2013 image dataset showed that the f-scores increased both on digital-born (from 16.2% to 53.2%) and natural-scene images (from 14.2% to 46.4%).

## References

[1] Clavelli, A., Karatzas, D. and Llad'os, J.: A framework for the assessment of text extraction algorithms on complex colour images, *ICDAR*, pp. 19–26 (2010).

[2] Epshtein, B., Ofek, E. and Wexler, Y.: Detecting Text in Natural Scenes with Stroke Width Transform, *IEEE Conf. on CVPR*, pp. 1–8 (2010).

[3] Huang, W., Lin, Z., Yang, J. and Wang, J.: Text localization in natural images using stroke feature transform and text covariance descriptors, *ICCV*, IEEE, pp. 1241–1248 (2013).

[4] Karatzas, D. and et al.: ICDAR 2013 Robust Reading Competition, *ICDAR* (2013).

[5] Karthikeyan, S., Jagadeesh, V. and Manjunath, B.: Learing bottom-up text attention maps for text detection using stroke width transform, *ICIP* (2013).

[6] Meng, Q., Song, Y., Zhang, Y. and Liu, Y.: TEXT DETECTION IN NATURAL SCENE WITH EDGE ANALYSIS, *ICIP* (2013).

[7] Neumann, L. and Matas, J.: Text localization in real-world images using efficiently pruned exhaustive search, *ICDAR*, pp. 687–691 (2011).

[8] Neumann, L. and Matas, J.: A method for text localization and recognition in real-world images, *Computer Vision–ACCV 2010*, Springer, pp. 770–783 (2011).

[9] Niblack, W.: *An introduction to digital image processing*, Strandberg Publishing Company (1985).

[10] Otsu, N.: A threshold selection method from gray-level histograms, *Automatica*, Vol. 11, No. 285-296, pp. 23–27 (1975).

[11] Senda, S., Minoh, M. and Ikeda, K.: A method of extraction of character pattern from a color image based on the uniformity of the character color of a string, *Technical report of IEICE (The Institute of Electronics, Information and Communication Engineers), PRU94-29*, Vol. 09 (1994).

[12] Wang, T., Wu, D. J., Coates, A. and Ng, A. Y.: End-to-end text recognition with convolutional neural networks, *ICPR*, IEEE, pp. 3304–3308 (2012).

[13] Yao, C., Bai, X., Liu, W., Ma, Y. and Tu, Z.: Detecting Texts of Arbitrary Orientations in Natural Images, *CVPR*, pp. 1083–1090 (2012).

[14] Yi, C. and Tian, Y.: Text String Detection from Natural Scenes by Structure-based Partition and Grouping, *ICIP*, Vol. 20, No. 9, pp. 2594–2605 (2011).

[15] Zhong, Y., Karu, K. and Jain, A. K.: Locating text in complex color images, *Pattern Recognition*, Vol. 28, No. 10, pp. 1523–1535 (1995).