Detecting Text in Natural Scenes with Stroke Width Transform

Conference Paper *in* Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition · July 2010

DOI: 10.1109/CVPR.2010.5540041 · Source: IEEE Xplore

3 authors, including:

Boris Epshtein
Houghton Mifflin Harcourt
19 PUBLICATIONS **1,826** CITATIONS
SEE PROFILE

Eyal Ofek
Microsoft
160 PUBLICATIONS **7,513** CITATIONS
SEE PROFILE

Detecting Text in Natural Scenes with Stroke Width Transform

Boris Epshtein Eyal Ofek Yonatan Wexler

Microsoft Corporation

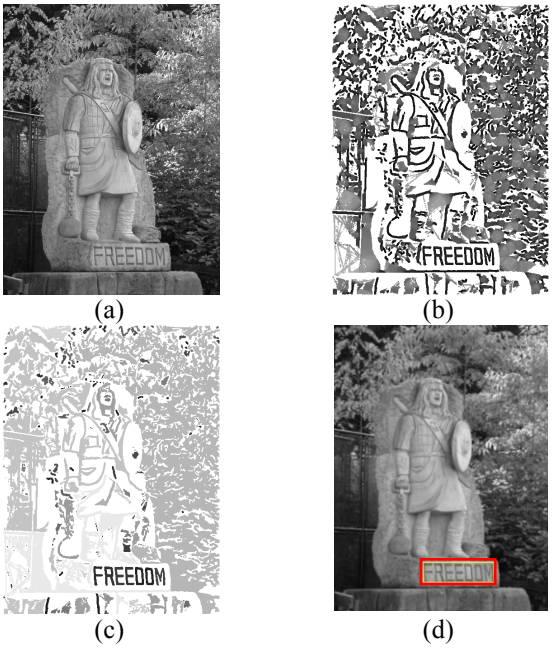
Abstract

We present a novel image operator that seeks to find the value of stroke width for each image pixel, and demonstrate its use on the task of text detection in natural images. The suggested operator is local and data dependent, which makes it fast and robust enough to eliminate the need for multi-scale computation or scanning windows. Extensive testing shows that the suggested scheme outperforms the latest published algorithms. Its simplicity allows the algorithm to detect texts in many fonts and languages.

1. Introduction

Detecting text in natural images, as opposed to scans of printed pages, faxes and business cards, is an important step for a number of Computer Vision applications, such as computerized aid for visually impaired, automatic geo-coding of businesses, and robotic navigation in urban environments. Retrieving texts in both indoor and outdoor environments provides contextual clues for a wide variety of vision tasks. Moreover, it has been shown that the performance of image retrieval algorithms depends critically on the performance of their text detection modules. For example, two book covers of similar design but with different text, prove to be virtually indistinguishable without detecting and OCRing the text. The problem of text detection was considered in a number of recent studies [1, 2, 3, 4, 5, 6, 7]. Two competitions (Text Location Competition at ICDAR 2003 [8] and ICDAR 2005 [9]) have been held in order to assess the state of the art. The qualitative results of the competitions demonstrate that there is still room for improvement (the winner of ICDAR 2005 text location competition shows recall=67% and precision=62%). This work deviates from the previous ones by defining a suitable image operator whose output enables fast and dependable detection of text. We call this operator the Stroke Width Transform (SWT), since it transforms the image data from containing color values per pixel to containing the most likely stroke width. The resulting system is able to detect text regardless of its scale, direction, font and language.

When applied to images of natural scenes, the success rates of OCR drop drastically, as shown in Figure 11. There are several reasons for this. First, the majority of OCR engines are designed for scanned text and so depend on segmentation which correctly separates text from background pixels. While this is usually simple for scanned text, it is much harder in natural images. Second. natural images exhibit a wide range of imaging conditions, such as color noise, blur, occlusions, etc. Finally, while the page layout for traditional OCR is simple and structured, in natural images it is much harder, because there is far less text, and there exists less overall structure with high variability both in geometry and appearance.



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

Figure 1: The SWT converts the image (a) from containing gray values to an array containing likely stroke widths for each pixel (b). This information suffices for extracting the text by measuring the width variance in each component as shown in (c) because text tends to maintain fixed stroke width. This puts it apart from other image elements such as foliage. The detected text is shown in (d).

One feature that separates text from other elements of a scene is its nearly constant stroke width. This can be utilized to recover regions that are likely to contain text. In this work, we leverage this fact. We show that a local image operator combined with geometric reasoning can be

used to recover text reliably. The main idea presented in this work shows how to compute the stroke width for each pixel. Figure 1c shows that the operator output can be utilized to separate text from other high-frequency content of a scene. Using a logical and flexible geometric reasoning, places with similar stroke width can be grouped together into bigger components that are likely to be words. This reasoning also allows the algorithm to distinguish between text and arbitrary drawings as shown in Figure 2. Note that we do not require the stroke width to be constant throughout a letter, but allow slow bounded variations instead.

The method suggested here differs from previous approaches in that it does not look for a separating feature per pixel, like gradient or color. Instead, we collect enough information to enable smart grouping of pixels. In our approach, a pixel gradient is only important if it has a corresponding opposing gradient. This geometric verification greatly reduces the amount of detected pixels, as a stroke forces the co-occurrence of many similarly matched pairs in a small region. Another notable difference of our approach from previous work is the absence of scanning window over a multiscale pyramid, required by several other approaches [e.g. 3, 4, 25]. Instead, we perform a bottom-up integration of information, merging pixels of similar stroke width into connected components, which allows us to detect letters across a wide range of scales in the same image. Since we do not use a filter bank of a few discrete orientations, we detect strokes (and, consequently, text lines) of any direction.

Additionally, we do not use any language-specific filtering mechanisms, such as OCR filtering stage [3] or statistics of gradient directions in a candidate window pertaining to a certain alphabet. This allows us to come up with a truly multilingual text detection algorithm.

Figure 2: Detected text in natural images

Not every application of text detection requires a further step of character recognition. When such step is needed, a successful text segmentation step has great impact on the recognition performance. Several previous text detection algorithms [3, 18, 19] rely on classification of image regions and therefore are not providing a text segmentation mask required for subsequent OCR. Our method carries enough information for accurate text segmentation and so a good mask is readily available for detected text.

2. Previous work

A great number of works deals directly with detection of text from natural images and video frames. Related works from other domains study the extraction of linear features.

For comprehensive surveys of methods for text detection, see [1, 2]. In general, the methods for detecting text can be broadly categorized in two groups: texture-based methods and region-based methods. Texture-based methods [e.g. 3, 4, 18, 19, 22] scan the image at a number of scales, classifying neighborhoods of pixels based on a number of text properties, such as high density of edges, low gradients above and below text, high variance of intensity, distribution of wavelet or DCT coefficients, etc. The limitations of the methods in this category include big computational complexity due to the need of scanning the image at several scales, problems with integration of information from different scales and lack of precision due to the inherent fact that only small (or sufficiently scaled down) text exhibits the properties required by the algorithm. Additionally, these algorithms are typically unable to detect sufficiently slanted text.

Another group of text detection algorithms is based on regions [e.g. 5, 6, 23]. In these methods, pixels exhibiting certain properties, such as approximately constant color, are grouped together. The resulting connected components (CCs) are then filtered geometrically and using texture properties to exclude CCs that certainly cannot be letters. This approach is attractive because it can simultaneously detect texts at any scale and is not limited to horizontal texts. Our method falls into this category, but the main feature that we use is very different from the typically used color, edges or intensity similarity. We measure stroke width for each pixel and merge neighboring pixels with approximately similar stroke width into CCs, which form letter candidates.

The work that uses a somewhat similar idea of detecting character strokes is presented in [7]. The method, however, differs drastically from the algorithm developed in this paper. The algorithm proposed in [7] scans an image horizontally, looking for pairs of sudden changes of intensity (assuming dark text on bright background). Then the regions between changes of intensity are examined for color constancy and stroke width (a range of stroke widths is assumed to be known). Surviving regions are grouped within a vertical window of size W and if enough regions are found, a stroke is declared to be present. The limitations of this method include a number of parameters tuned to the scale of the text to be found (such as vertical window size W), inability to detect horizontal strokes, and the fact that detected strokes are not grouped into letter candidates, words and sentences. Consequently, the algorithm is only able to detect near-horizontal text. The performance results presented in the paper are done using a metric that is different from the ICDAR competition

metric. We implemented the metrics from [7] and show that our algorithm outperforms [7] - see Section 4 for comparison.

Another method [21] also uses the idea of stroke width similarity, but is restricted to finding horizontal lines of small text, due to the traversal along horizontal scan lines to detect vertical strokes, and the use of morphological dilation to connect candidate pixels into connected regions. While performance results on the ICDAR database are not provided, the algorithm would not be able to deal with arbitrary directions of strokes. Our method is invariant to the stroke direction (see Figures 8, 10, 12).

Finally, the work [25] uses the idea of stroke width consistency for detecting text overlays in video sequences. The limitations of the method include the need for integration over scales and orientations of the filter, and, again, the inherent attenuation to horizontal texts.

Our definition of stroke is related to linear features which are commonly dealt with in two domains: remote sensing (extraction of road networks) and medical imaging (blood vessel segmentation). In road detection, the range of road widths in an aerial or satellite photo is known and limited, whereas texts appearing in natural images can vary in scale drastically. Additionally, roads are typically elongated linear structures with low curvature, which is again not true for text. Most techniques of road detection rely on the assumptions listed above, and therefore are not directly applicable for text detection. For a survey of techniques, see [10]. The closest work is [11], which uses the fact that road edges are antiparallel for detecting points lying on center lines of the roads, then groups these candidate center points together. No attempt is made to use constant road width to facilitate grouping. Our method uses dense voting on **each pixel of the stroke**, thus resulting in a much more stable identification of strokes without requiring a difficult and brittle process of grouping center point candidates. Another method [12] uses lines extracted from low-res images and border edges extracted from hi-res images to find road candidates. In the case of text detection, a whole multiscale pyramid of images would be required for a similar strategy; moreover, small or thin text still is unlikely to be detected using this method.

For a survey on blood vessel segmentation, see [13]. Works in this domain use model fitting (snakes, generalized cylinders), ridge finding (ridge operators, binarization followed by thinning, wavelets) and other methods. Studies that use vessel width as an additional feature for tracking vessels starting from a user-designated seed include [14, 15]. None of the existing works try to detect vessels directly, in a bottom-up fashion, using low variance of widths, the way described in this work.

3. The text detection algorithm

In this section, we describe the text detection algorithm. We first define the notion of a stroke and then explain the *Stroke Width Transform* (3.1), and how it is used for grouping pixels into letter candidates (3.2). Finally, we describe the mechanism for grouping letters into bigger constructs of words and lines which enables further filtering (3.3). The flowchart of the algorithm is shown on Fig. 5.

3.1. The Stroke Width Transform

The *Stroke Width Transform* (SWT for short) is a local image operator which computes per pixel the width of the most likely stroke containing the pixel. The output of the SWT is an image of size equal to the size of the input image where each element contains the width of the stroke associated with the pixel. We define a stroke to be a contiguous part of an image that forms a band of a nearly constant width, as depicted in Figure 3(a). We do not assume to know the actual width of the stroke but rather recover it.

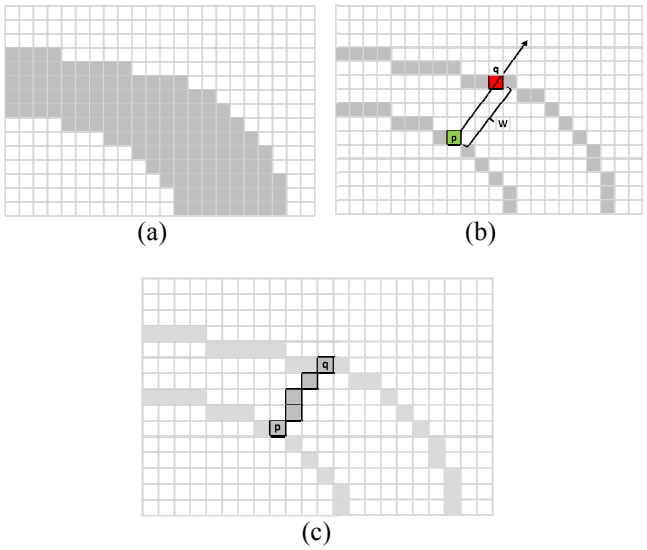


Figure 3: Implementation of the SWT. (a) A typical stroke. The pixels of the stroke in this example are darker than the background pixels. (b) p is a pixel on the boundary of the stroke. Searching in the direction of the gradient at p, leads to finding q, the corresponding pixel on the other side of the stroke. (c) Each pixel along the ray is assigned by the minimum of its current value and the found width of the stroke.

The initial value of each element of the SWT is set to ∞. In order to recover strokes, we first compute edges in the image using Canny edge detector [16]. After that, a gradient direction $\boldsymbol{d_p}$ of each edge pixel $\boldsymbol{p}$ is considered (Fig. 3b). If $\boldsymbol{p}$ lies on a stroke boundary, then $\boldsymbol{d_p}$ must be roughly perpendicular to the orientation of the stroke. We follow the ray $\boldsymbol{r}=\boldsymbol{p}+\boldsymbol{n}\cdot\boldsymbol{d_p}$, $\boldsymbol{n}>0$ until another edge pixel $\boldsymbol{q}$ is found. We consider then the gradient direction $\boldsymbol{d_q}$ at pixel

$\boldsymbol{q}$. If $\boldsymbol{d_q}$ is roughly opposite to $\boldsymbol{d_p}$ ($\boldsymbol{d_q}$ = $-\boldsymbol{d_p}\pm\pi/6$), each element $\boldsymbol{s}$ of the SWT output image corresponding to the pixels along the segment $[\boldsymbol{p},\boldsymbol{q}]$ is assigned the width $\|\overrightarrow{p-q}\|$ unless it already has a lower value (Fig. 4a). Otherwise, if the matching pixel $\boldsymbol{q}$ is not found, or if $\boldsymbol{d_q}$ is not opposite to $\boldsymbol{d_p}$, the ray is discarded. Figure 3 shows the process of SWT computation.

As shown in Fig. 4b, the SWT values in more complex situations, like corners, will not be true stroke widths after the first pass described above. Therefore, we pass along each non-discarded ray again, compute median SWT value $\boldsymbol{m}$ of all its pixels, and then set all the pixels of the ray with SWT values above $\boldsymbol{m}$ to be equal to $\boldsymbol{m}$.

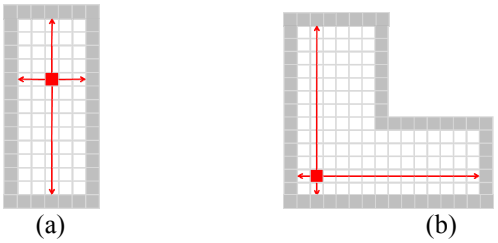


Figure 4: Filling pixels with SWT values. (a) An example red pixel is filled with minimum between the lengths of vertical and horizontal rays passing through it. Proper stroke width value is stored. (b) An example red pixel stores the minimum between the two rays lengths; this is not the true stroke width - this shows the necessity of the second pass (see text).

The SWT operator described here is linear in the number of edge pixels in the image and also linear in the maximal stroke width, determined at the training stage.

3.2. Finding letter candidates

The output of the SWT is an image where each pixel contains the width of the most likely stroke it belongs to. The next step of the algorithm is to group these pixels into letter candidates. In this section we describe a set of fairly general rules employed towards this end.

Two neighboring pixels may be grouped together if they have similar stroke width. For this we modify the classical Connected Component algorithm [17] by changing the association rule from a binary mask to a predicate that compares the SWT values of the pixels. We found that a very conservative comparison suffices, and group two neighboring pixels if their SWT ratio does not exceed 3.0. This local rule guarantees that strokes with smoothly varying widths will also be grouped together, hence allowing more elaborate fonts and perspective distortions (Fig. 8). In order to accommodate both bright text on dark background and vice-versa, we apply the algorithm twice, once along $\boldsymbol{d_p}$ and once along $-\boldsymbol{d_p}$.

We now need to identify components that may contain text. For this we employ a small set of fairly flexible rules. The parameters of each rule were learned on the training set of [8]. The first test we perform is to compute the variance of the stroke width within each connected component and reject the ones whose variance is too big. This rejects areas such as foliage, that is prevalent in many natural images including both city and rural scenes and is known to be hard to distinguish from text. As shown in Figure 1(c), this test suffices to distinguish the text region which is much more consistent than the foliage. The learned threshold is half the average stroke width of a particular connected component.

Many natural processes may generate long and narrow components that may be mistaken for possible letters. Additional rule prunes out these components, by limiting their aspect ratio to be a value between 0.1 and 10. Similarly, we limit the ratio between the diameter of the connected component and its median stroke width to be a value less than 10.

Another common problem is connected components that may surround text, such as sign frames. We eliminate those by ensuring that the bounding box of a component will includes not more than two other components (this often happens in italicized text).

Lastly, components whose size is too small or too large may be ignored. Learned from our training set, we limit the acceptable font height to be between 10 and 300 pixels. The use of height measure enables us to detect connected scripts, such as handwriting and Arabic fonts, and accounts for the tendency of small letters in a word to get connected due to aliasing and imperfection of the edge detection stage.

Remaining components are considered letter candidates and in the next section we describe how these are agglomerated into words and lines of text.

All thresholds for the geometric tests were learned on the fully annotated training set [8] by optimizing performance. Specifically, on the training set we computed the connected components representing letters within each bounding box (provided by annotation) by doing adaptive binarization using Otsu algorithm [20], followed by extraction of connected components. We tuned the parameters of each filtering rule so that 99% of the connected components were detected.

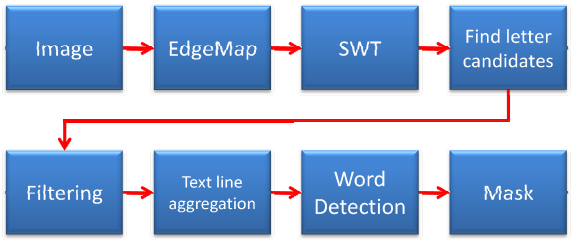


Figure 5: The flowchart of the algorithm

3.3. Grouping letters into text lines

To further increase the reliability of the algorithm, we continue a step forward to consider groups of letters.

Finding such groups is a significant filtering mechanism as single letters do not usually appear in images and this reasoning allows us to remove randomly scattered noise.

An important cue for text is that it appears in a linear form. Text on a line is expected to have similarities, including similar stroke width, letter width, height and spaces between the letters and words. Including this reasoning proves to be both straightforward and valuable. For example, a lamp post next to a car wheel would not be mistaken for the combination of letters "O" and "I" as the post is much higher than the wheel. We consider each pair of letter candidates for the possibility of belonging to the same text line. Two letter candidates should have similar stroke width (ratio between the median stroke widths has to be less than 2.0). The height ratio of the letters must not exceed 2.0 (due to the difference between capital and lower case letters). The distance between letters must not exceed three times the width of the wider one. Additionally, average colors of candidates for pairing are compared, as letters in the same word are typically expected to be written in the same color. All parameters were learned by optimizing performance on the training set, as described in Section 3.2.

At the next step of the algorithm, the candidate pairs determined above are clustered together into chains. Initially, each chain consists of a single pair of letter candidates. Two chains can be merged together if they share one end and have similar direction. The process ends when no chains can be merged. Each produced chain of sufficient length (at least 3 letters in our experiments) is considered to be a text line.

Finally, text lines are broken into separate words, using a heuristic that computes a histogram of horizontal distances between consecutive letters and estimates the distance threshold that separates intra-word letter distances from inter-word letter distances. While the problem in general does not require this step, we do it in order to compare our results with the ones in ICDAR 2003 database [8]. In the results shown for our database [26] we do not employ this step, as we have marked whole text lines.

4. Experiments

In order to provide a baseline comparison, we ran our algorithm on the publicly available dataset in [24]. It was used in two most recent text detection competitions: ICDAR 2003 [8] and ICDAR 2005 [9]. Although several text detection works have been published after the competitions, no one claimed to achieve better results on this database; moreover, the ICDAR dataset remains the most widely used benchmark for text detection in natural scenes.

Many other works remain impossible to compare to due to unavailability of their custom datasets. The ICDAR dataset contains 258 images in the training set and 251 images in the test set. The images are full-color and vary in size from 307×93 to 1280×960 pixels. Algorithms are compared with respect to *f*-measure which is in itself a combination of two measures: *precision* and *recall*. We follow [8] and describe these here for completeness sake.

Figure 6: Text detection results on several images from the ICDAR test set. Notice the low number of false positives.

The output of each algorithm is a set of rectangles designating bounding boxes for detected words. This set is called the *estimate* (see Fig. 6). A set of ground truth boxes, called the *targets* is provided in the dataset. The match m_p between two rectangles is defined as the area of intersection divided by the area of the minimum bounding box containing both rectangles. This number has the value one for identical rectangles and zero for rectangles that have no intersection. For each estimated rectangle, the closest match was found in the set of targets, and vice versa. Hence, the best match $m(r;R)$ for a rectangle r in a set of rectangles R is defined by

$$m(r;R) = max\{m_p(r;r0) \mid r0 \in R\} \quad (1)$$

Then, the definitions for precision and recall is

$$Precision = \frac{\sum_{r_e \in E} m(r_e,T)}{|E|} \quad (2)$$

$$Recall = \frac{\sum_{r_t \in T} m(r_t,E)}{|T|} \quad (3)$$

where T and E are the sets of ground-truth and estimated rectangles respectively.

The standard *f* measure was used to combine the precision and recall figures into a single measure of quality. The relative weights of these are controlled by a parameter α, which we set to 0.5 to give equal weight to precision and recall:

$$f = \frac{1}{\frac{\alpha}{Precision} + \frac{1-\alpha}{Recall}} \quad (4)$$

The comparison between precision, recall and *f*-measure of different algorithms tested on the ICDAR database is shown in Table 1.

In order to determine the importance of stroke width information (Section 3.1) and geometric filtering (Section 3.2), we additionally run the algorithm on the test set in two more configurations: configuration #1 had all the stroke width values less than ∞ set to 5 (changing this constant did not affect the results significantly). Configuration #2 had the geometric filtering turned off. In both cases, the precision and recall dropped (p=0.66, r=0.55 in configuration #1, p=0.65, r=0.5 in configuration #2). This shows the importance of information provided by the SWT.

In Figure 7 we show typical cases where text was not detected. These are due to strong highlights, transparency of the text, size that is out of bounds, excessive blur, and curved baseline.

Algorithm	Precision	Recall	*f*	Time (sec.)
Our system	**0.73**	**0.60**	**0.66**	**0.94**
Hinnerk Becker*	0.62	0.67	0.62	14.4
Alex Chen	0.60	0.60	0.58	0.35
Qiang Zhu	0.33	0.40	0.33	1.6
Jisoo Kim	0.22	0.28	0.22	2.2
Nobuo Ezaki	0.18	0.36	0.22	2.8
Ashida	0.55	0.46	0.50	8.7
HWDavid	0.44	0.46	0.45	0.3
Wolf	0.30	0.44	0.35	17.0
Todoran	0.19	0.18	0.18	0.3
Full	0.1	0.06	0.08	0.2

Table 1: Performance comparison of text detection algorithms. For more details on ICDAR 2003 and ICDAR 2005 text detection competitions, as well as the participating algorithms, see [9] and [10].
*The algorithm is not published.

In order to compare our results with [7], we have implemented the comparison measures proposed there. Our algorithm performance is as follows: the Word Recall rate is 79.04%, and the Stroke Precision is 79.59% (since our definition of a stroke is different from [7], we counted connected components inside and outside the ground truth rectangles. Additionally, we counted Pixel Precision, the number of pixels inside ground truth rectangles divided by the total number of detected pixels. This ratio is 90.39%. This outperforms the results shown in [7]

In addition to providing result on ICDAR database, we propose a new benchmark database for text detection in natural images [26]. The database, which will be made freely downloadable from our website, consists of 307 color images of sizes ranging from 1024x1360 to 1024x768. The database is much harder than ICDAR, due to the presence of vegetations, repeating patterns, such as windows, virtually undistinguishable from text without OCR, etc. Our algorithm's performance on the database is as follows: precision: 0.54, recall: 0.42, *f*-measure: 0.47. Again, in measuring these values we followed the methodology described in [8].

Since one of the byproducts of our algorithm is a letter mask, this mask can be used as a text segmentation mask. In order to evaluate the usability of the text segmentation produced by our algorithm, we presented an off-the-shelf OCR package with several natural images, containing text and, additionally, with the binarized images representing text-background segmentation. The results of the OCR in both cases are shown in Figure 11.

5. Conclusion

In this work we show how to leverage on the idea of the recovery of stroke width for text detection. We define the notion of a stroke and derive an efficient algorithm to compute it, producing a new image feature. Once recovered, it provides a feature that has proven to be reliable and flexible for text detection. Unlike previous features used for text detection, the proposed SWT combines dense estimation (computed at every pixel) with non-local scope (stroke width depends on information contained sometimes in very far apart pixels). Compared to the most recent available tests, our algorithm reached first place and was about 15 times faster than the speed reported there. The feature was dominant enough to be used by itself, without the need for actual character recognition step as used in some previous works [3]. This allows us to apply the method to many languages and fonts.

There are several possible extensions for this work. The grouping of letters can be improved by considering the directions of the recovered strokes. This may allow the detection of curved text lines as well. We intend to explore these directions in the future

References

[1] J. Liang, D. Doermann, H. Li, "Camera-based analysis of text and documents: a survey", International Journal on Document Analysis and Recognition", 2005, vol. 7, n° 2-3, pp. 83-200

[2] K. Jung, K. Kim, A. K. Jain, "*Text information extraction in images and video: a survey*", Pattern Recognition, p. 977 – 997, Vol 5. 2004.

[3] X. Chen, A. Yuille, "Detecting and Reading Text in Natural Scenes", Computer Vision and Pattern Recognition (CVPR), pp. 366-373, 2004

[4] R. Lienhart, A. Wernicke, "Localizing and Segmenting Text in Images and Videos" IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 12, NO. 4, APRIL 2002, pp. 256-268

[5] A. Jain, B. Yu, "Automatic Text Location in Images and Video Frames", Pattern Recognition 31(12): 2055-2076 (1998)

[6] H-K Kim, "Efficient automatic text location method and content-based indexing and structuring of video database". J Vis Commun Image Represent 7(4):336–344 (1996)

[7] K. Subramanian, P. Natarajan, M. Decerbo, D. Castañòn, "Character-Stroke Detection for Text-Localization and Extraction", International Conference on Document Analysis and Recognition (ICDAR), 2005
[8] "ICDAR 2003 robust reading competitions", Proceedings of Seventh International Conference on Document Analysis and Recognition, 2003, pp. 682-68
[9] "*ICDAR 2005 text locating competition results*", Eighth International Conference on Document Analysis and Recognition, 2005. Proceedings. pp 80-84(1)
[10] L.i J. Quackenbush, "A Review of Techniques for Extracting Linear Features from Imagery", Photogrammetric Engineering & Remote Sensing, Vol. 70, No. 12, December 2004, pp. 1383–1392
[11] P. Doucette, P. Agouris,, A. Stefanidis, "Automated Road Extraction from High Resolution Multispectral Imagery", Photogrammetric Engineering & Remote Sensing, Vol. 70, No. 12, December 2004, pp. 1405–1416
[12] A. Baumgartner, C. Steger, H. Mayer, W. Eckstein, H. Ebner, "Automatic road extraction based on multi-scale, grouping, and context", Photogrammetric Engineering & Remote Sensing, 65(7): 777–785 (1999)
[13] C. Kirbas, F. Quek, "A review of vessel extraction techniques and algorithms", ACM Computing Surveys (CSUR), Vol. 36(2), pp. 81-121 (2004)
[14] S. Park, J. Lee, J. Koo, O. Kwon, S. Hong, S, "Adaptive tracking algorithm based on direction field using ML estimation in angiogram", In IEEE Conference on Speech and Image Technologies for Computing and Telecommunications. Vol. 2. 671-675 (1999).
[15] Y. Sun, "Automated identification of vessel contours in coronary arteriogramsby an adaptive tracking algorithm", IEEE Trans. on Med. Img. 8, 78-88 (1989).
[16] J. Canny, "A Computational Approach To Edge Detection", IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.
[17] B. K. P. Horn, "Robot Vision", McGraw-Hill Book Company, New York, 1986.
[18] J. Gllavata, R. Ewerth, B. Freisleben, "Text Detection in Images Based on Unsupervised Classification of High-Frequency Wavelet Coefficients", 17th International Conference on Pattern Recognition (ICPR'04) - Volume 1, pp. 425-428
[19] H. Li, D. Doermann, O. Kia, "Automatic Text Detection and Tracking in Digital Video", IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 9, NO. 1, JANUARY 2000
[20] N. Otsu, "A threshold selection method from gray-level histograms". *IEEE Trans. Sys., Man., Cyber.* **9**: 62–66 (1979)
[21] V. Dinh, S. Chun, S. Cha, H. Ryu, S. Sull "An Efficient Method for Text Detection in Video Based on Stroke Width Similarity", ACCV 2007
[22] Q. Ye, Q. Huang, W. Gao, D. Zhao, "Fast and robust text detection in images and video frames", Image and Vision Computing 23 (2005) 565–576
[23] Y. Liu, S. Goto, T. Ikenaga, "A Contour-Based Robust Algorithm for Text Detection in Color Images", IEICE TRANS. INF. & SYST., VOL.E89–D, NO.3 MARCH 2006
[24] http://algoval.essex.ac.uk/icdar/Datasets.html.
[25] C. Jung, Q. Liu, J. Kim, "A stroke filter and its application for text localization", PRL vol 30(2), 2009
[26] http://research.microsoft.com/en-us/um/people/eyalofek/text_detection_database.zip

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

Figure 7: Examples of failure cases. These include: strong highlights (a), transparent text (b), text that is too small (c), blurred text (d) and text with curvature beyond our range (e)

Figure 8: The algorithm was able to detect text in very challenging scenarios such as blurry images, non planar surfaces, non uniform backgrounds, fancy fonts and even three dimensional fonts. All examples here are from the ICDAR dataset.

Figure 9: Detected text in various languages. The photos were taken from the web. These include printed and hand written, connected and disjoint scripts.

Figure 10: A mix of text detection in images taken on a city street using a video camera. Note the large variability of detected texts, including hard cases such as obscured texts and three-dimensional texts.

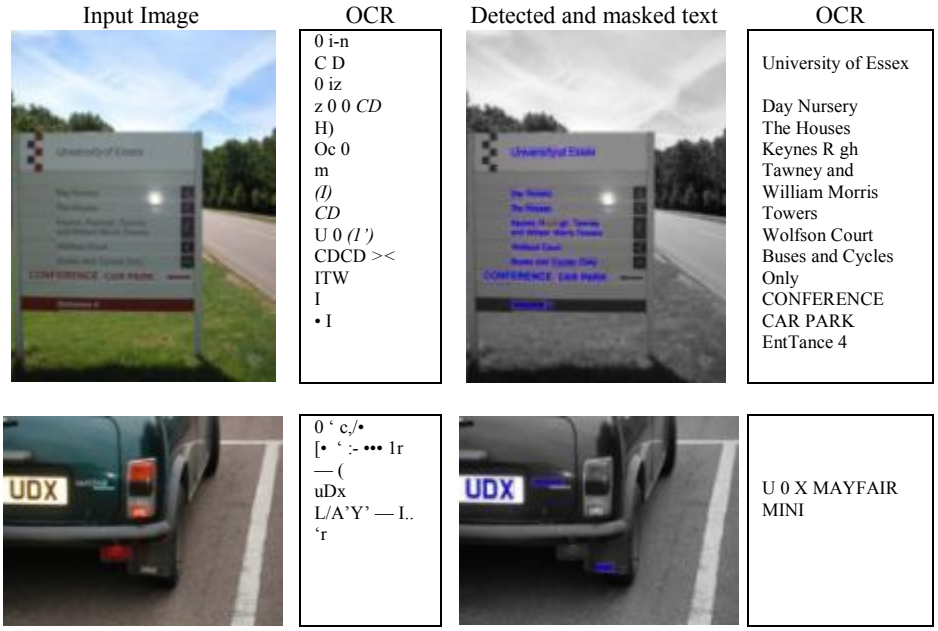


Figure 11: OCR results on the original image and on the recovered text segmentation masks. Columns, from left to right: original image, OCR output on the original image, text segmentation mask (superimposed on graylevel versions of original images), OCR output on the masks.

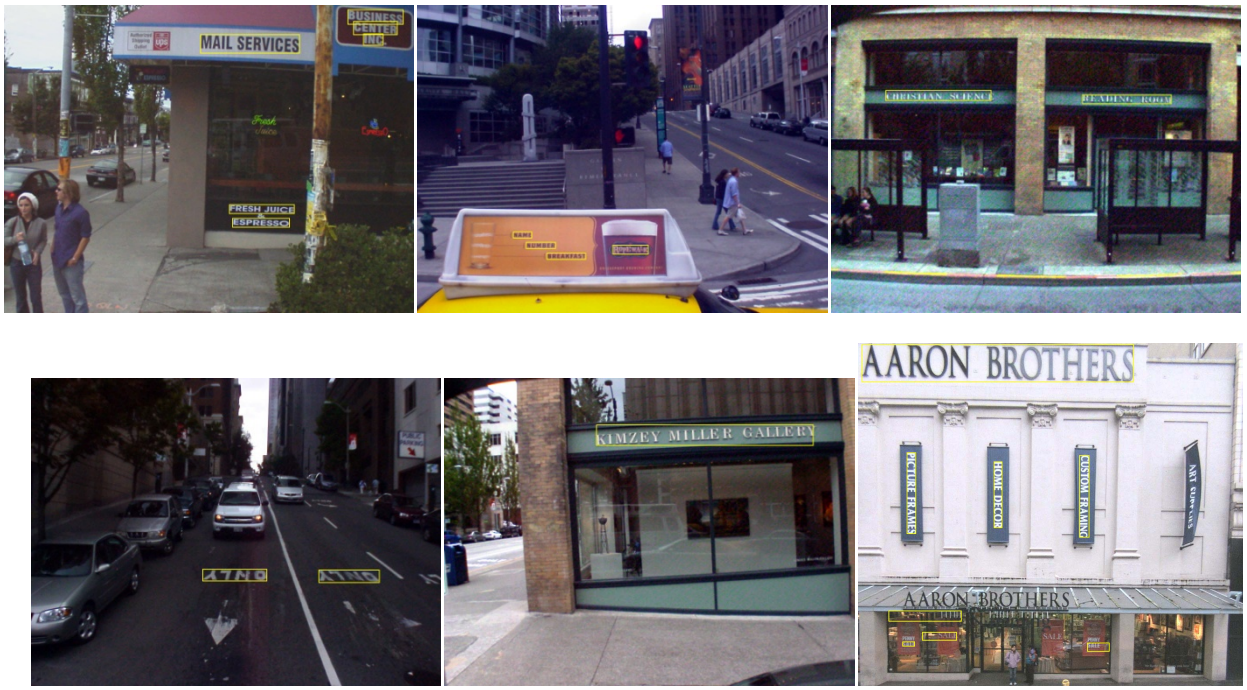


Figure 12: Additional examples of detecting text in streetside images.

.