ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

A Review: Text Detection in Natural Scenes with Stroke Width Transform

Mr.Hemil A. Patel*, Mrs.Kishori S. Shekokar
M.E. Student of Computer Department, Sigma Institute of Engineering, Vadodara
Head of Computer Department, Sigma Institute of Engineering, Vadodara

Abstract

Detecting text segments in an image of a natural scene, by using the Stroke Width Transform The approach of SWT is grouping pixels together in an intelligent way, instead of looking for separating features of pixels. 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 application receives an RGB image to search in, and returns a new image where the discovered text segments are marked. Due to the features of the SWT, the resulting system is able to detect text regardless of its scale, direction, font and language.

Keywords: Text Detection, edge detection, Stroke Width Transform (SWT), Connected Component algorithm.

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 [1, 2].

Text detection in natural scenes is a highly researched field, and there are numerous approaches for solving this problem.

However, most text detection schemes restrict the user to Specific languages, scale and direction of the text.

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) [2], 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.

SWT is an elegant approach to detect text. However, its performance heavily relies on the quality of edges which drive the transform computation [3].

The connected component (CC) based approaches are motivated by grouping pixels which exhibit similar text properties [3].

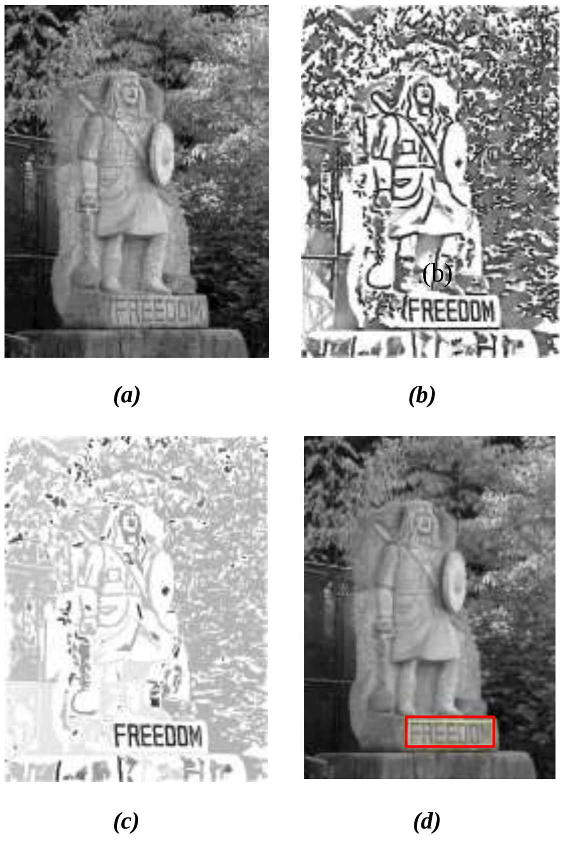


Figure 1: The SWT converts the image (a) from containing gray values to an array containing likely stroke widths for each pixel (b). This information suffices for extracting the text by measuring the width variance in each component as shown in (c) because text tends to

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

maintain fixed stroke width. This puts it apart from other image elements such as foliage. The detected text is shown in (d) [2].

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 compare to scanned text.

Text Detection approaches

Current text detection approaches can be roughly classified into three groups: region-based approaches, texture based approaches and hybrid approaches.

Region-based approach

Region-based approaches [1, 2] attempt to use similarity criterions of text, such as color, size, stroke width, edge and gradient information, to gather pixels together into connected components (CCs) and non-text CCs are filtered out with geometric hypothesis or conditional random fields (CRFs). These approaches usually have lower computation cost and the outputs can closely cover text regions. However, they may meet great challenges in background cluttered [5].

Texture-based approach

Texture-based approaches are similar to the approaches of general object detection with sliding windows. They utilize distinct textural properties of text regions to extract candidate sub-windows and the final outputs are formed by merging these sub-windows. These approaches are able to deal with background cluttered scenes. But there are always some 'imitation' regions in outputs and they are very time consuming [2].

Hybrid approach

Hybrid approaches [4, 5] seek to introduce textural property of text regions into region-based approach. These approaches take advantages of both region-based approaches which can closely cover text regions and texture-based approaches which can estimate of the coarse text location in cluttered scenes. Nonetheless, they are also time consuming [5].

Detection algorithm

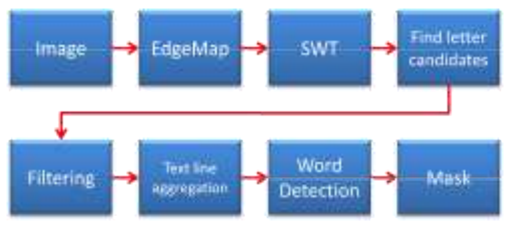


Figure 2: The flowchart of the algorithm [1, 2].

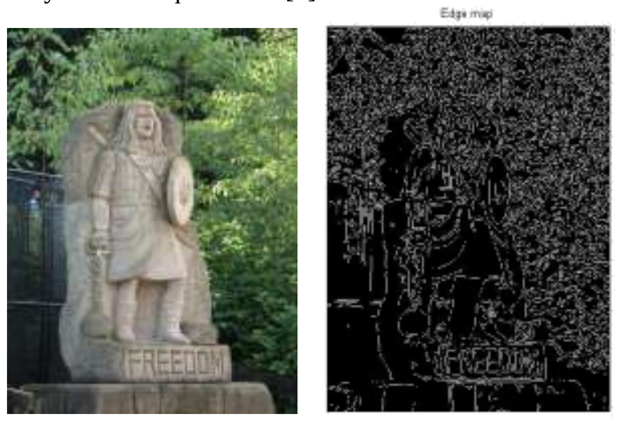
Edge Map:

In order to recover strokes, we first compute edges in the image using canny edge detector [6].

The points at which image brightness changes sharply are typically organized into a set of curved line segments termed edges [6].

The purpose of detecting sharp changes in image brightness is to capture important events and changes in properties of the world [6].

The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986. Canny also produced a computational theory of edge detection explaining why the technique works [7].

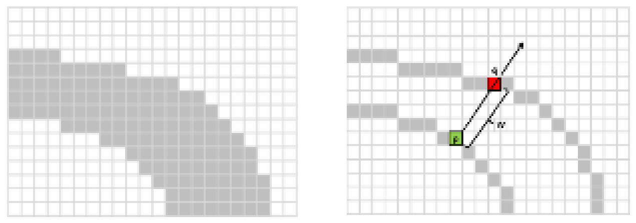


(a) *(b)*

Figure 3: (a) input image (b) edge detected image [1].

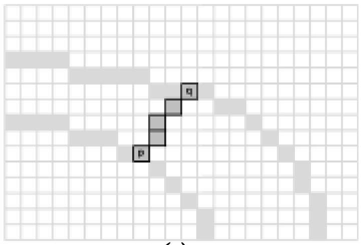
SWT (The Stroke Width Transform)

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 [2].



(a) *(b)*

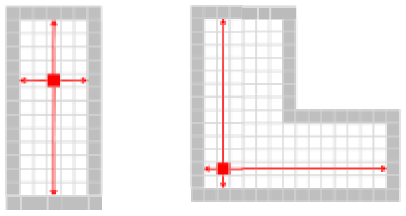
ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114



(c)

Figure 4: 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, and 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 [2].

First, all pixels are initialized with ∞ as their stroke width. Then, We consider the edges as possible stroke boundaries, and we wish to find the width of such stroke. If p is an edge pixel, the direction of the gradient is roughly perpendicular to the orientation of the stroke boundary. Therefore, the next step is to calculate the gradient direction gp of the edge pixels, and follow the ray r=p+n*gp (n>0) until we find another edge pixel q. If the gradient direction gq at q is roughly opposite to gp, then each pixel in the ray is assigned the distance Between p and q as their stroke width, unless it already has a lower value. If, however, an edge pixel q is not found, or gq is not opposite to gp, the ray is discarded. In order to accommodate both bright text on a dark background and dark text on a bright background, we need to apply the algorithm twice: once with the ray direction gp and once with –gp [1].



(a) *(b)*

Figure 5: 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 e two rays lengths; this is not the true stroke width - this shows the necessity of the second pass (see text) [2].

As shown in Fig. 5b, 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 m of all its pixels, and then set all the pixels of the ray with SWT values above m to be equal to m [2].

Finding letter candidates

We now have a map of the most likely stroke-widths for each pixel in the original image. The next step is to group these pixels into letter candidate. This will be done by first grouping pixels with similar stroke width, and then applying several rules to distinguish the letter candidates.

The grouping of the image will be done by using a Connected Component algorithm. In order to allow smoothly varying stroke widths in a letter, we will let two pixels to be grouped together if their SWT ratio is less than 3.0 [1].

Now we must detect the connected components which can pass as letter candidates, by applying a set of fairly flexibly rules. These rules are as follows:

I. The variance of the stroke-width within a component must not be too big. This helps with Rejecting foliage in natural images, which are commonly mistaken for text.

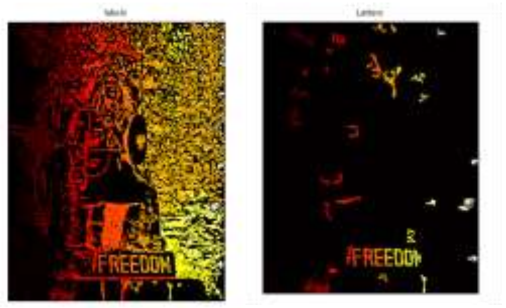
II.The aspect ratio of a component must be within a small range of values, in order to reject long and narrow components.

III. The ratio between the diameter of the component and its median stroke width to be less than a learned threshold. This also helps reject long and narrow components.

IV. Components whose size is too large or too small will also be ignored. This is done by limiting the length, width, and pixel count of the component.

V. In addition to these rules so far, I added another rule which helped me eliminate much noise.

This rule state that the ratio between the pixel count of the component and the amount of pixels in the bounding box of the component should be within a bounded range. This rejects components that spread over a large space, yet have a small pixel count, and components which cover most of their bounding box [1].



(a) *(b)*

Figure 6: (a) SWT output (b) after finding letter candidate output [1].

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

Grouping letters into text lines

Since single letters are not expected to appear in images, we will now attempt to group closely positioned letter candidates into regions of text.
This filters out many falsely-identified letter candidates, and improves the reliability of the algorithm results [1].
Again, we will use a small set of rules to group letters together into regions of text. These rules will consider pairs of letters, and are as follows:
I. Two letter candidates should have similar stroke width. For this reason we limit the ratio between the median stroke-widths to be less than some threshold.
II. The ratio between the heights of the letters and between the widths of the letters must not Exceed 2.5. This is due to capital letters next to lower case letters.
III. The distance between letters must not exceed three times the width of the wider one.
IV. Characters of the same word are expected to have a similar color; therefore we compare the Average color of the candidates for pairing [1].
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 [2].

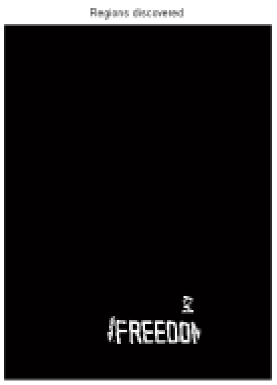


Figure 7: Text detected image [1].

Performance measures parameters

Algorithms are compared with respect to f-measure which is in itself a combination of two measures: precision and recall.
The output of each algorithm is a set of rectangles designating bounding boxes for detected words. This set is called the estimate.
The match mp 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

$$m(r, R) = \max\{mp(r; r0 \mid r0 \in R)\} \qquad (1)$$

Then, the definitions for precision and recall is

$$\text{precision} = \frac{\sum r_e \in E^{m(r_e,T)}}{|E|} \qquad (2)$$

$$\text{Recall} = \frac{\sum r_t \in T^{m(r_t,E)}}{|T|} \qquad (3)$$

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:

$$f = \frac{1}{\frac{\alpha}{\text{precision}} + \frac{1-\alpha}{\text{Recall}}} \qquad (4)$$

Conclusion

This review results that Text Detection in Natural Scenes with Stroke Width Transform allows us to apply the method to many languages and fonts. Connected component algorithm having some pre-defined rules. This algorithm use two approaches like texture based and Region-based approaches so detection of single character is not possible. The grouping of letters can be improved by considering the directions of the recovered strokes. This may also allow us to detect curved text lines.

References

[1] Gili Werner." Text Detection in Natural Scene with Stroke Width Transform".ICBV, February, 2013.
[2] B. Epshtein, E. Ofek, and Y. Wexler, "Detecting text in natural scenes with stroke width transform," in Computer Vision and Pattern Recognition(CVPR),Conference on. IEEE, 2010.
[3] S. Karthikeyan, Vignesh Jagadeesh and B. S. Manjunath" learning bottom-up text attention maps for text detection using stroke width transform",20th IEEE International Conference on ICIP,2013.
[4] Anhar Risnumawan, Palaiahankote, Shivakumara, Chee Seng Chan, Chew Lim Tan. "A robust arbitrary text detection system for natural scene images". Elsevier, 2014.
[5] Yuning Du, Genquan Duan, Haizhou Ai." context-based text detection in natural scenes" In Proceedings of the ICIP,19th IEEE International Conference ,2012.

[6] Masoud Nosrati, Ronak Karimi, Mehdi Hariri, Kamran Malekian. “Edge Detection Techniques in Processing Digital Images: Investigation of Canny Algorithm and Gabor Method”. World Applied Programming, Vol (3), Issue (3),2013.

[7] J. Canny.”A Computational Approach To Edge Detection”,IEEE Trans. Pattern Analysis and Machine Intelligence, 1986.

[8] Chucai Yi,YingLi Tian” Text String Detection from Natural Scenes by Structure-based artition and Grouping ”,Image Processing, IEEE Transactions,2011.

[9] Huizhong Chen, Sam S. Tsai, Georg chroth, David M. Chen, Radek Grzeszczuk and Bernd Girod”robust text detection in natural images with edge-enhanced maximally stable extremal regions”,18th IEEE International Conference ,ICIP,2011.

[10] Yao Li,Huchuan Lu” Scene Text detection via Stroke Width”,21st International Conference on Pattern Recognition (ICPR), IEEE 2012.