Automatic text detection for mobile augmented reality translation

Marc Petter[1] Victor Fragoso[2] Matthew Turk[2] Charles Baur[1]

[1]École Polytechnique Fédérale de Lausanne
{marc.petter, charles.baur}@epfl.ch

[2]University of California, Santa Barbara
{vfragoso, mturk}@cs.ucsb.edu

Abstract

We present a fast automatic text detection algorithm devised for a mobile augmented reality (AR) translation system on a mobile phone. In this application, scene text must be detected, recognized, and translated into a desired language, and then the translation is displayed overlaid properly on the real-world scene. In order to offer a fast automatic text detector, we focused our initial search to find a single letter. Detecting one letter provides useful information that is processed with efficient rules to quickly find the reminder of a word. This approach allows for detecting all the contiguous text regions in an image quickly. We also present a method that exploits the redundancy of the information contained in the video stream to remove false alarms. Our experimental results quantify the accuracy and efficiency of the algorithm and show the strengths and weaknesses of the method as well as its speed (about 160 ms on a recent generation smartphone, not optimized). The algorithm is well suited for real-time, real-world applications.

1. Introduction

Text detection in natural scenes is a field of active research as it provides useful information for several applications, such as assistance for the visual impaired [3, 17], multimedia database indexing and retrieval tasks [2], text removal in video sequences [12], and others. Thus, many inventive solutions have been proposed for achieving text detection in natural scenes. The results and performance obtained with those methods vary depending on the approach and the intended application context. However, most of these methods are not suitable for use in real-time applications and on mobile devices. The aim of this research was to provide a novel and fast text detection algorithm for use in mobile augmented reality using cell phones for computation and display.

The capabilities of mobile devices (with respect to computing power, sensors, display resolution, etc.) have increased dramatically in recent years, making mobile phones an important platform for a wide variety of applications. One example of such an application is described in [8], where Fragoso *et al*. presented an augmented reality translation application for smartphones which translates a word of interest and presents the result in an augmented reality overlay on the live video stream. For initiating the translation process, TranslatAR required the user to tap on the screen near the center of the word of interest. In order to offer an improved, more ergonomic version of the application, we developed a fast text detection method that automatically finds the text for translation.

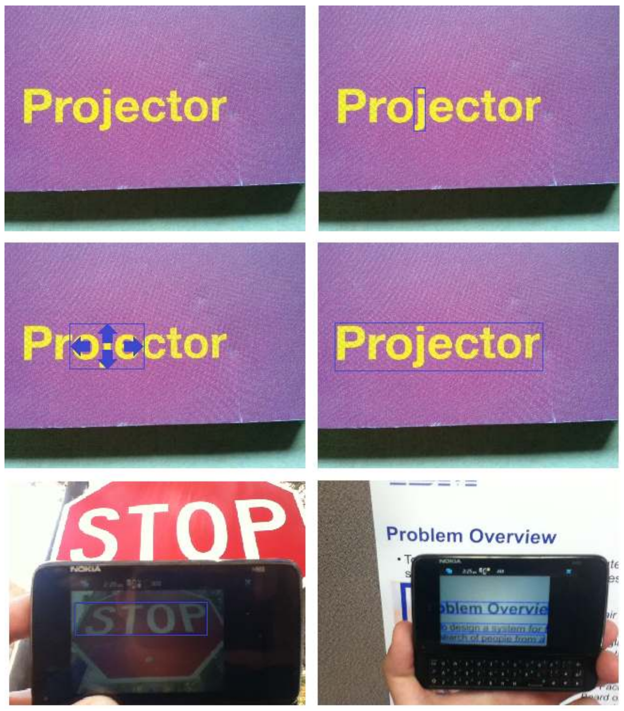

Figure 1. The algorithm scans the input image (upper left) until it finds a zone of interest that contains text (upper right). Subsequently, the algorithm expands the zone of interest with efficient rules (middle left), and finally, our method produces the final bounding box (middle right and bottom row).

Our algorithm introduces a novel method, intended for use on mobile devices, for finding words in natural scene images, typically of billboards, road signs, informative signs or other texts that may be useful to translate. Some reasonable assumptions were made to simplify the detection of such words: (1) The surface on which the text is printed is planar; (2) The text will be almost horizontal ($0 \pm \delta$ degrees, where δ is a small orientation distortion); and (3) The contrast between the background and the text is high. Although these assumptions are occasionally violated, they describe a high percentage of public signs and notices that are likely targets for translation.

A key observation behind our algorithm is that words are normally arrangements of closely-spaced letters on a single line. Therefore, by only knowing the position of one letter in a word, useful information can be extracted. This information can subsequently be processed by efficient and effective rules, which can find the remaining letters of the word. Ergo, a substantial amount of time can be spared avoiding a conventional search for all the letters in a word.

This paper is organized as follows: Prior work about automatic text detection is discussed in Section 2; Section 3 describes our proposed method and presents in detail each step of the algorithm; Section 4 gives implementation details; experimental results are presented in Section 5; and finally, we conclude in Section 6.

2. Related work

Many different methods have been proposed for automatic text detection. We will briefly summarize the two main categories of these methods: connected component-based and texture-analysis methods.

The first group of methods, connected component-based methods, use the similarity of text pixels to group them together (*e.g.*, [10, 13, 15, 19]). They search for homogeneous properties such as color, edge directions, and strengths to build connected components. These methods check which connected components are words according to some criteria.

Epshtein *et al*. [6] proposed an algorithm based on the fact that the width of a stroke composing a letter remains almost the same anywhere in the character. The image is processed in order to compute the Stroke Width Transform (SWT) which is a map that indicates per pixel the width of the potential stroke at that location. Subsequently, pixels with similar stroke-width are grouped and the algorithm detects text with the stroke-width-variance of a connected-component, as text regions possess constant stroke-width. This method provides very good results, as well as scale-invariance, orientation-invariance and robustness. However, the computational cost for a mobile device seems fairly high as the stroke width needs to be computed for every pixel.

Sanketi *et al*. [17] proposed an algorithm based on blobs formed with some connectivity rules. These blobs are then grouped together to produce super-blobs. To find text regions among those super-blobs, a series of tests considering several image features are applied to categorize the super-blobs as text. The results obtained with this algorithm were especially good with low-resolution and blurry images. Nevertheless, the high number of image features considered in the approach makes the algorithm cumbersome to use.

Ma *et al*. [16] developed an algorithm for text detection on a mobile device based on connected components and a bank of feature-constraints. Given a connected component and its boundig box, the set of constraints is used to select letter candidates. Finally, the average height of the bounding boxes is used to discard outliers. However, the set of rules discard letters that are captured at a certain scale and view angle, as the set of rules applied are defined to constraint certain shape and size.

The second group of methods, texture analysis methods (*e.g.*, [11, 12, 14, 20]), leverage the textural properties of text regions, making differentiating letters from backgrounds possible. The image is thus segmented using texture.

Ferreira *et al*. [7] developed a text detector based on a texture segmentation. The algorithm is based on Gabor filters combined with edge information (Sobel response) as features that consequently are clustered with K-Means. However, the memory footprint can be high and clustering can take time, making the approach not suitable for real-time applications.

Our method is based on the main key ideas of both groups to provide an efficient and robust algorithm. However, most of the steps in our method are part of the first category. Thus, in order to offer a fast novel algorithm in this category, the key idea of our method is to build only one connected component per word (see Section 3). This allows for efficiency in comparison with the other methods of the first group, which create all the connected components of an image.

3. Proposed text detection algorithm

The general structure of the algorithm is shown in Fig. 2. The algorithm works on a grayscale image for faster processing time and can be separated into three main steps: (1) Localize a first potential letter (zone of interest); (2) Verify that a letter was found; (3) Find the rest of the word based on the found letter.

3.1. Step 1: Finding a zone of interest

The aim of this step is to find a zone of interest that may contain a letter. The proposed approach for locating the zone of interest is based on existing methods [9, 15, 19] because of their efficiency and good performance. These

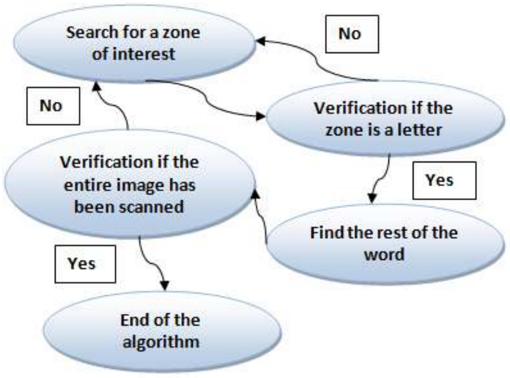
Search for a zone of interest
No
Verification if the zone is a letter
No
Verification if the entire image has been scanned
Yes
Find the rest of the word
Yes
End of the algorithm

Figure 2. Structure of our algorithm. The algorithm can be separated in 3 main steps: Finding a zone of interest; Verifying if this is really a letter; and finding the rest of the word.

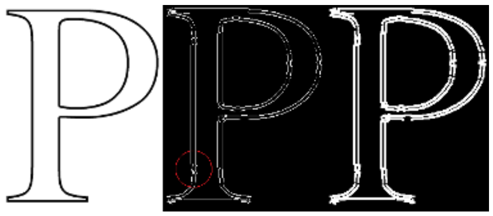

Figure 3. Contour reconstruction process: Original picture (Left), edge-map produced with Canny (Middle), reconstruction with dilatation operator and a cross-shaped structuring element (Right).

methods leverage the high rate of edges contained in text areas. Therefore, a potential letter can be found on an edge map by building objects composed of closed contours that later can be categorized as letter or non-letter. In the following paragraphs, we explain in more detail the building blocks of this step.

Edge detection. Prior to detecting edges, a Gaussian smoothing filter of size 5x5 pixels is applied to reduce noise that could cause errors in further computation. The Canny edge detector [1] is used for producing a binary map indicating the presence per-pixel of every edge. This edge detector is efficient and provides accurate results which makes it suitable for our purpose.

Contour dilation. The original image is sometimes too blurry for edges to be detected. Thus, some shapes, including letters, could be overlooked by the edge detection and not appear in the edge map (see Fig. 3). To ensure the continuity of the contour, a preprocessing step is necessary before starting the contour building. Avoiding this step can produce an incorrect contour by the algorithm. For reconnecting the edge pixels together, we use dilation, a binary morphological tool. For our implementation, a cross-

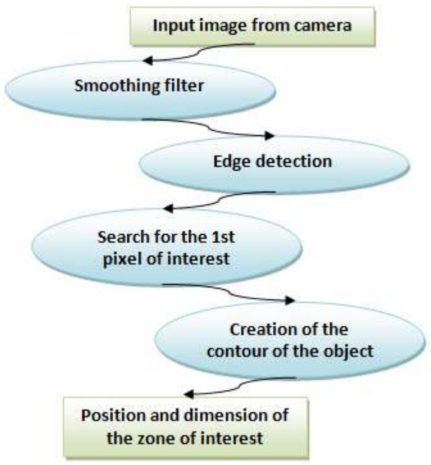
Input image from camera
Smoothing filter
Edge detection
Search for the 1st pixel of interest
Creation of the contour of the object
Position and dimension of the zone of interest

Figure 4. Structure of the first step of the algorithm.

shaped structuring element of pixel size 3 worked the best for filling the holes in the contours.

Contour building. The algorithm starts scanning from left to right and top to bottom to find an edge pixel in the binary map. When an edge pixel is found, the contour of the object containing this pixel is built with an 8-connectivity connected component algorithm. The 8-connectivity algorithm [4] is a region-based segmentation algorithm which checks the 8 pixel neighbors of a pixel and connects this pixel with its similar neighbors.

Information about the bounding box containing the computed contour, such as height, width, position of the centroid, etc., is available as an outcome of this step.

3.2. Step 2: Verification of the zone of interest

The main intention of this step is to verify whether or not the zone of interest actually contains a character. This task is not straightforward, as the words contained on billboards, road signs or books have different sizes or fonts which makes learning precise shapes of letters a challenging task. However, since signs are typically meant to be easily readable, discriminating text regions from non-text regions with geometric information should be possible.

A Support Vector Machine (SVM) and a set of image features are adopted to accomplish the discriminating task. SVMs are widely used in the literature (*e.g.*, [12, 20]) and are quite useful for binary categorization tasks. SVMs have a strong mathematical foundation and provide a simple geometric explanation of their classification. Moreover, the solutions found with SVMs are always the global minima solutions [5], *i.e.*, solutions with the best support that generalize well.

In order to select the best features to address this discrimination task, we conducted experiments evaluating several

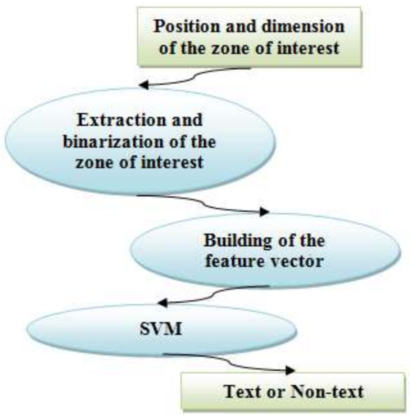
Position and dimension of the zone of interest
Extraction and binarization of the zone of interest
Building of the feature vector
SVM
Text or Non-text

Figure 5. Structure of the second step of the algorithm.

combinations of image features. The most effective features [1] we found are the First-Order Moments (FOM) defined as in equations 1 and 2; and Second-Order Moments (SOM) normalized with the number of pixels on the contour (NB) as defined in equations 3 and 4, where x, y are the coordinates of the pixel in the clipped zone of interest and $I(x,y)$ denotes the intensity of the pixel:

$$FOM\ 1 = \sum_x \sum_y x\, I(x,y) \tag{1}$$

$$FOM\ 2 = \sum_x \sum_y y\, I(x,y) \tag{2}$$

$$\frac{SOM\ 1}{NB} = \frac{\sum_x \sum_y x^2\, I(x,y)}{NB} \tag{3}$$

$$\frac{SOM\ 2}{NB} = \frac{\sum_x \sum_y y^2\, I(x,y)}{NB} \tag{4}$$

3.3. Step 3: Finding the rest of the word

In order to robustly find the rest of the word given the position of the first letter, we propose to combine two features that will provide information about the surrounding characters: image intensities and the edge map. These features will determine when to stop scanning in the surrounding areas, and therefore, to determine the spatial extent of the bounding box.

3.3.1 Background and Foreground intensities

Given the first letter of the word or phrase to be detected, the background and foreground intensities in the grayscale domain can be extracted. We can safely assume in most cases that each word is contained in a homogeneous colored background and the letters have approximately the same intensity; we can then infer the intensity and find the remaining letters. The K-Means algorithm with $k = 2$ is used to find the two intensities. In order to know which intensity corresponds to the letter, we create a second bounding box with the same center as the first bounding box of the zone of interest. The width and height of the new bounding box are computed as follows: $width_2 = width + \delta_w$ and $height_2 = height + \delta_h$, where $\delta_h = \delta_w = 2$ pixels (see Fig. 6). The pixels on the perimeter of that new box are likely to be background elements, and therefore, the closest intensity to the mean of those perimeter pixels is chosen to be the intensity of the background. Consequently, the remaining intensity is attributed to the letter.

[1]These features are easy and fast to compute and provide an acceptable discriminating rate according to our observations.

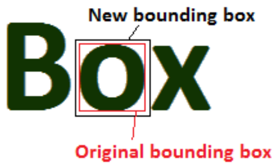

Figure 6. Method to find the intensity of the background. A second box is created and the mean of the intensity of the pixels on the perimeter of the new box is associated to the background.

3.3.2 Edge map

Edge pixels around the found letter are likely to be part of the rest of the word because text regions present a high edge density. Useful information to estimate the position of the remaining letters is extracted from the adjacent edge pixels of the zone of interest.

3.3.3 Scanning

In order to speed up the full word bounding box computation, we scan the image horizontally with 3 line segments. A single line segment is positioned on top, middle and bottom of the found character's bounding box. Each segment is then scanned on the left and right side of the zone of interest considering a gap of size s on every side. The algorithm looks for pixels with intensities similar to the letter's intensity along the segment. Edge pixels that are present in the analyzed gap are considered simultaneously. In this manner we guarantee that in fact we are likely to see pixels representing letters on the image. The size of the gap used in our algorithm is calculated as follows: $s = 1.1 * H$, where H is the height of the found letter. The size as a function of the height allows us to consider the breach that exists between two adjacent characters in a word. However, when such breach is less than $1.1 * H$, the algorithm considers both

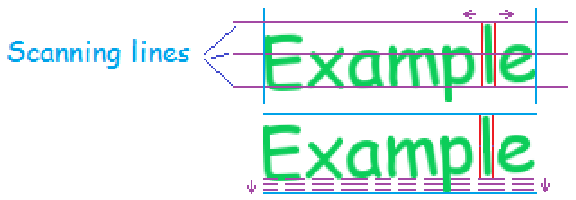

Figure 7. Horizontal and vertical scanning to find letter pixels (intensity or edge pixels). Gaps of size s are analyzed between the letters during the procedure.

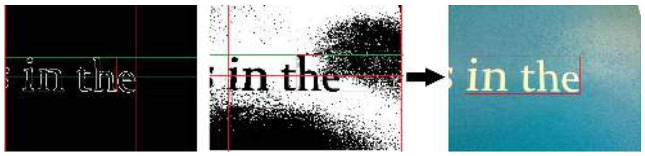

Figure 8. Considering only information from edges (left) or intensity (middle) can determine an incorrect bounding box. Combining both features produces a better bounding box (right).

adjacent words as a single word. The procedure is applied until no edge pixels are detected or no similar intensity is found in the analyzed gap of every line segment. As an outcome of this procedure we obtain the width of the bounding box.

To find the height boundaries, we scan pixels along horizontal line segments with lengths equals to the computed bounding box widths described earlier (see Fig. 7). We scan these lines following the same pixel criteria of intensities and edges used earlier. The algorithm moves the lines up and down until this criteria is fulfilled.

The combination of these two procedures computes a rectangular bounding box that encloses the letters of a certain text in the analyzed image (see Fig. 8).

Scanning with three horizontal parallel line segments tolerates a certain perspective distortion of the letters that compose the word. However, the produced bounding box computed with these procedures may be slightly larger or smaller than the minimum bounding box due to noise present in the image.

3.4. Post-detection processing

Once a word is found, the search for additional words in the image continues until every pixel of the image not part of a word bounding box has been scanned.

3.5. Additional step for video stream: Temporal redundancy

An additional step is applied when our algorithm is used on a video stream, which is the case in TranslatAR. In order to keep track of stable text-regions and remove false alarms as much as possible, we leverage the temporal information that we can obtain from the video stream. We are interested in tracking these stable text regions. Since the scene does not change much from frame to frame, assuming that the frames on the video stream are generated at a high frame rate, the stable regions repeat and the position and area of the true-positives detected bounding boxes does not vary much; therefore, false alarms will behave more unstably in this sense. The stability of these correct bounding boxes allows the algorithm to remove a fair amount of false positives.

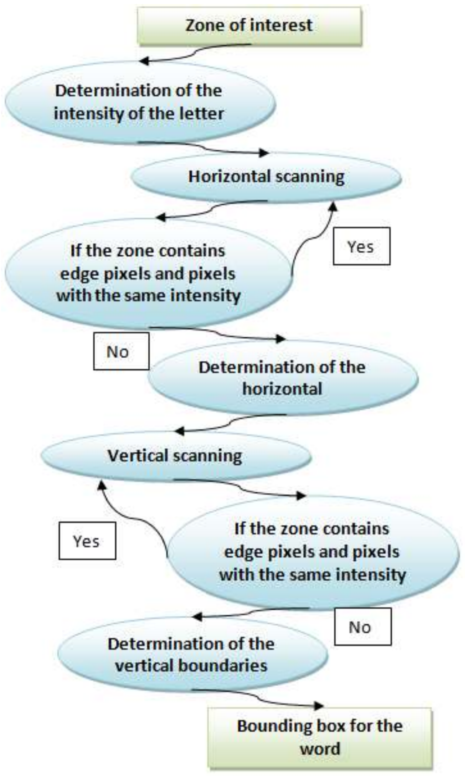
Zone of interest
Determination of the intensity of the letter
Horizontal scanning
If the zone contains edge pixels and pixels with the same intensity
Yes
No
Determination of the horizontal
Vertical scanning
Yes
If the zone contains edge pixels and pixels with the same intensity
No
Determination of the vertical boundaries
Bounding box for the word

Figure 9. Structure of the third step of the algorithm.

The algorithm retains the center position and the area of the detected bounding boxes on the first frame. On subsequent frames, we re-detect the bounding boxes and we match them with the previously seen boxes based on areas and centroids. For every retained bounding box we increment a counter c if the bounding box matches a previously seen region, and decremented if it is not seen. A bounding box is considered to be stable if $c > 1$.

There are three different cases for matching that occur when comparing two bounding boxes (see Fig. 10):

1. One of the bounding boxes contains the other one.
2. The two bounding boxes intersect.
3. The two bounding boxes do not intersect and neither of them contains the other one.

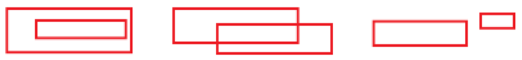

Figure 10. Considered cases when comparing bounding boxes: Inclusion (left), Intersection (middle), and Disjoint boxes (right).

Cases 1 and 2 are situations where the bounding boxes in question can represent the same word. In order to know which case correspond to two bounding boxes, the positions of their upper left and lower right corners are compared. Once two bounding boxes are considered to be potentially the same word, further aspects are analyzed in order to determine a match.

To determine a match for the first case 1 we evaluate the ratio r between the smallest area and the biggest area. A match is determined if $r > 0.7$. For the second case, we evaluate the absolute value of the displacement of the centers c_1 and c_2, *i.e.*, $\delta_x = |c_{x_1} - c_{x_2}|$ and $\delta_y = |c_{y_1} - c_{y_2}|$, as well as the ratio of the areas used in the first case. The method declares a match considering the following criteria: $\delta_x < \epsilon_x$, $\delta_y < \epsilon_y$, and $r > 0.7$, where $\epsilon_x = 0.35 * width$, $\epsilon_y = 0.35 * height$ (the height and width correspond to the smallest bounding box).

Subsequently, we average the centroids and areas of the matching bounding boxes in order to keep track of the box on the remaining frames.

4. Implementation details

The system was tested on a Nokia N900 smartphone (first available in fall 2009) which possesses a TI OMAP 3430 SoC with a 600MHz ARM Cortex A8 CPU and runs Maemo (Linux-based OS).

The automatic text detection algorithm was implemented in C++ using the OpenCV 2.2 library. The different test cases that evaluated the algorithm were implemented with Qt libraries to measure the runtime and its accuracy.

We integrated our text detection algorithm into TranslatAR: We coupled the new text detection algorithm in the pipeline used by TranslatAR, reusing therefore its components (see [8] for more details).

5. Evaluation

In order to thoroughly evaluate the algorithm, different experiments were carried out. We evaluated the text detection algorithm and the integration of this method with TranslatAR which uses Tesseract as OCR [18].

5.1. Text detection accuracy

We created our own dataset to test the algorithm in a more realistic context, *i.e.*, low-resolution, mobile camera, and others. This dataset comprises 400 images, each containing a single word from natural scene which follow the assumptions made for this project (see section 1), and 400 non-text images.

In order to evaluate the performance of the proposed method, we evaluated every outcome manually, and the outcome was labeled as successful if all the letters of the word were contained in the bounding box. The results of this experiment are reported in Table 1.

Table 1. Text detection accuracy.

Words Found	False Alarms	Precision	Recall	f-score
87%	41%	68%	87%	76%

The algorithm found the majority of the words; however, it is also susceptible to a substantial rate of false alarms. By analyzing the failures, both missing words and false alarms, we concluded that the main problem occurs in the second step (*c.f.* Section 3.2, verifying the zone of interest). The SVM was the most common source of failure for the case of false negatives, failing to detect words (see Table 2).

Table 2. Distribution of failure for the missed words.

1st step [%]	2nd step [%]	3rd step [%]
4.57	81.04	14.39

We observed that false alarms arise in images with high edge densities, as the first step declares those regions as zones of interest and therefore the second step declares them as text regions. We noticed that the SVM with SOM/FOM as features, tends to declare any symmetrical non-text region in an image as text (see Fig. 11).

5.2. Temporal Redundancy

We collected 50 videos of different texts from indoor and outdoor natural scenes and conducted a quantitative visual evaluation with and without the temporal redundancy. The temporal redundancy enabled the application to reduce up to 70% of the false alarms. However, the approach also penalized the detection rate when the user imposed a significant movement of the phone because the position of the bounding box around the word changed significantly. Approximately 15% of the words were not found because of the temporal redundancy.

5.3. Runtime

As TranslatAR requires a real-time text detection algorithm, the faster the detector the more compelling the application appears. The average runtime for an image of size 320x240 (size used in TranslatAR) is reported in Table 3.

The first step takes 58% of the total time. Nevertheless, the algorithm runs fast on the mobile phone which makes the approach a good asset for automatic text detection.

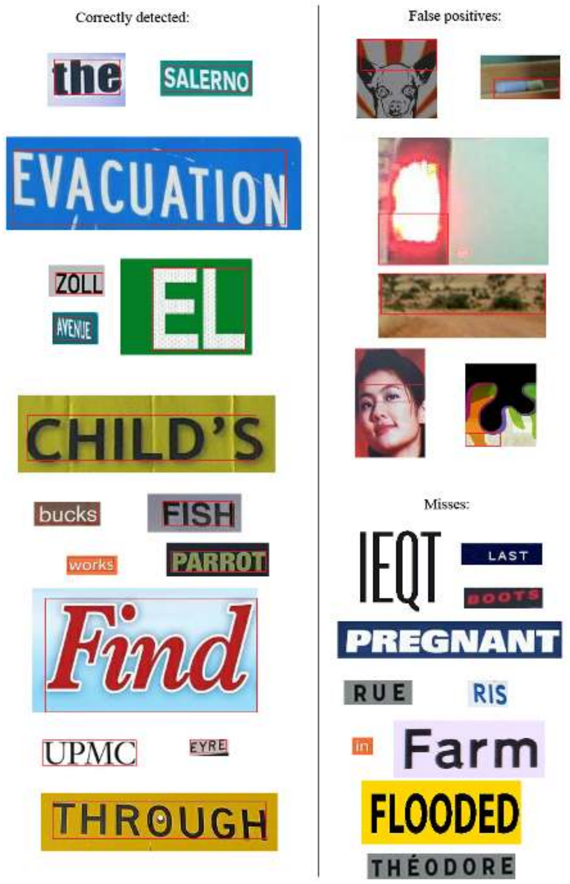

Figure 11. Words correctly detected (left) and failures (right).

Table 3. Mean runtime for an image of size 320 x 240 on the Nokia N900. The experiment was conducted 20 times on 50 text images and 50 non-text images.

Step	Mean runtime [ms]
1st step	**94.66**
Smoothing, edge detection and morphological operator	32.52
Zone of Interest	42.06
Building Contours	20.08
2nd step	**38.22**
Zone extraction and binarization	27.66
Feature vector computation	7.06
SVM call	3.5
3rd step	**26.28**
Color blob detection	14.24
Edge detection	12.04
Total	**159.16**

Table 4. Testing the integration with TranslatAR. 100 images from indoors and outdoors natural scenes were used for this experiment.

Words found	Words read correctly	Correct translation
58	26	20

5.3.1 Implementation in TranslatAR

Some modifications to improve the ergonomics of TranslatAR were implemented during this project. To launch the translation system, the user is required to point the phone at the text and to momentarily be still. The system detects a "stability" sensed via the accelerometers and triggers the detection-translation procedure. However, the system can become too sensitive to the user's movements and can affect the performance of the detection.

In order to evaluate the efficiency of the integration, 100 natural scene images (50 from indoors and 50 from outdoors) were tested for translation.

The text detection algorithm found 58% of the words in images with homogeneous background and with a high contrast between letters and the background. However, the sharpness of the input image was not very good in general and the edge detection algorithm did not perform well.

The OCR system (Tesseract) recognized correctly 26% of the entire dataset used which corresponds to 45% of the correctly detected words. The translation component was able to translate 20% of the words correctly detected. The text detection, translation and texture rendering took about 2.9 sec; meanwhile the tracking system took about 238ms per frame.

We determined some external causes that made the OCR fail (see Fig. 12). First, noisy pixels contained in the bounding box. Second, imprecision on the bounding box edges: big bounding box containing spurious pixels or short bounding box truncating letters.

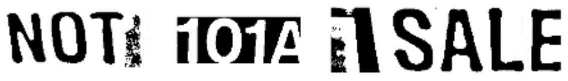

Figure 12. Imprecise bounding boxes and OCR readings: There are some noise pixels in the detected region - "NOTI" (left); the bounding box is too small - "101/1" (middle); and the bounding box is too big - "El SALE" (right).

6. Conclusion

This paper presented an algorithm for automatic text detection which makes it suitable for an augmented reality translation system on a mobile phone. The non-optimized algorithm takes on average around 160ms to run which makes the algorithm suitable for interactive applications;

running the detector on the device avoids external factors such as network latency, remote server-faults, and others. The key idea of the proposed method to speed-up the text detection is based on finding first a single character and leveraging the information extracted from the detected character to find the remaining characters. We also presented an approach that exploits the congruent temporal redundancy of information on the live video stream to remove spurious text regions. This redundancy enabled removal of 70% of the false alarms generated in the detection stage.

According to our experimental results that quantify the efficiency, performance, and integration of the translation system, the algorithm was able to detect words with a precision of 68% and a recall of 87% in our dataset devised for the TranslatAR context.

Our experiments showed that the features used in conjunction with the SVM to categorize a candidate region as text are not so efficient and generated false alarms when an image presented a high density of edges. Therefore, a more extensive experiment should be devised to determine better features. A potential feature that should be considered is the stroke-width of the characters, as it allowed a good text detection rate in a recent work [6], if it can be made efficient enough for use in an interactive application.

Acknowledgments: The Nokia Research Center Palo Alto provided partial support for this research. We would like to thank Daniel Vaquero for his help. M. Petter would like to specially thank Francesco Mondada for his official supervising during this research. V. Fragoso would like to thank UC MEXUS-CONACYT (Fellowship 212913) for the funding. This work was done while M. Petter was at UCSB Four Eyes Lab.

References

[1] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8:679–698, Nov. 1986. 3

[2] D. Chen, J.-M. Odobez, and H. Bourlard. Text detection and recognition in images and video frames. *Pattern Recognition*, 37(3):595 – 608, 2004. 1

[3] X. Chen and A. L. Yuille. Adaboost learning for detecting and reading text in city scenes. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2004. 1

[4] C.-C. Cheng, G.-J. Peng, and W.-L. Hwang. Subband weighting with pixel connectivity for 3-d wavelet coding. *IEEE Trans. on Image Process.*, 18(1):52–62, Jan. 2009. 3

[5] N. Cristianini and J. Shawe-taylor. An introduction to support vector machines and other kernel-based learning methods. *Cambridge Univ. Press*, 2000. 3

[6] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2963–2970, June 2010. 2, 8

[7] S. Ferreira, V. Garin, and B. Gosselin. A text detection technique applied in the framework of a mobile camera-based application. In *Proc. of Camera-based Document Analysis and Recognition (CBDAR)*, Seoul, Korea, 2005. 2

[8] V. Fragoso, S. Gauglitz, S. Zamora, J. Kleban, and M. Turk. Translatar: A mobile augmented reality translator. In *IEEE Wksp. on Applications of Computer Vision (WACV)*, pages 497 –502, Jan. 2011. 1, 6

[9] J. Gao and J. Yang. An adaptive algorithm for text detection from natural scenes. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 84–89, 2001. 2

[10] A. Jain and B. Yu. Automatic text location in images and video frames. In *Proc. Fourteenth Intl. Conf. on Pattern Recognition*, volume 2, pages 1497–1499, Aug 1998. 2

[11] K. C. Kim, H. R. Byun, Y. J. Song, Y. woo Choi, S. Y. Chi, K. K. Kim, and Y. K. Chung. Scene text extraction in natural scene images using hierarchical feature combining and verification. In *Intl. Conf. on Pattern Recognition*, pages 679–682, 2004. 2

[12] C. W. Lee, K. Jung, and H. J. Kim. Automatic text detection and removal in video sequences. *Pattern Recogn. Lett.*, 24:2607–2623, November 2003. 1, 2, 3

[13] C. Li, X. Ding, and Y. Wu. Automatic text location in natural scene images. In *Proc. Sixth Intl. Conf. on Document Analysis and Recognition*, pages 1069 –1073, 2001. 2

[14] R. Lienhart and A. Wernicke. Localizing and segmenting text in images and videos. *IEEE Trans. on Circuits and Systems for Video Technology*, 12(4):256 –268, Apr 2002. 2

[15] Y. Liu, S. Goto, and T. Ikenaga. A contour-based robust algorithm for text detection in color images. *IEICE - Trans. Inf. Syst.*, E89-D:1221–1230, March 2006. 2

[16] D. Ma, Q. Lin, and T. Zhang. Mobile camera based text detection and translation, 2011. 2

[17] P. Sanketi, H. Shen, and J. Coughlan. Localizing blurry and low-resolution text in natural images. In *IEEE Wksp. on Applications of Computer Vision (WACV)*, pages 503 –510, Jan. 2011. 1, 2

[18] R. Smith. An overview of the tesseract ocr engine. In *Proceedings of the Ninth Intl. Conf. on Document Analysis and Recognition - Volume 02*, pages 629–633, Washington, DC, USA, 2007. IEEE Computer Society. 6

[19] Q. Ye, W. Gao, W. Wang, and W. Zeng. A robust text detection algorithm in images and video frames. In *Proc. Fourth Intl. Conf. on Information, Communications and Signal Processing and Fourth Pacific-Rim Conf. on Multimedia*, volume 2, pages 802 – 806 vol.2, Dec. 2003. 2

[20] Q. Ye, Q. Huang, W. Gao, and D. Zhao. Fast and robust text detection in images and video frames. *Image and Vision Computing*, 23:565–576, 2005. 2, 3