SCIENTIFIC JOURNAL OF POLISH NAVAL ACADEMY

2019 (LX) 2 (217)

sciendo

DOI: 10.2478/sjpna-2019-0014

TEXT SPOTTING IN THE WILD WITH EMBEDDED DEVICE

Artur Zacniewski[1], Tadeusz Bodnar[2]

Polish Naval Academy, Institute of Naval Weapon and Computer Science, Śmidowicza 69 Str., 81-127 Gdynia, Poland; e-mail: [1]a.zacniewski@amw.gdynia.pl, [2]t.bodnar@amw.gdynia.pl; ORCID ID: [1]0000-0002--3604-4051, [2]0000-0001-8398-6087

ABSTRACT

Detecting and recognizing text in natural scenes (e.g. streets, restaurants, shops, etc.) could be a part of an artificial intelligence system, especially with regard to the speech synthesis system. Properly detected text is passed to a recognition stage and then to the speech synthesis system, which translates text to speech. Research is carried out for the 'Toucan Eye' project — embedded device with artificial intelligence system able to help people with impaired sight. Due to constrained resources and abilities of embedded devices, criteria for text spotting must be met. First criterion is quality of detected and recognized regions with text and the second is time spent on both operations. Particular stages of the system and chosen methods of text spotting under aforementioned constraints are presented.

Key words:

text spotting, assisting persons with impaired sight, Toucan Eye.

Research article

INTRODUCTION

Scene text localization and recognition, known as the text-in-the-wild problem or text spotting, is a key component of potential applications such as automated translation, image/video database indexing or assistance to the visually impaired [1, 4, 8]. The paper presents particular stages of the system and reports chosen methods, whose goal is to perform the task of text spotting under the aforementioned constraints.

The process of text spotting is mainly decomposed into two stages — text detection followed by text recognition. *Text detection* refers to finding and localizing instances of text (for example words) from the image, correctly ignoring background clutter. *Text recognition* refers to the visual decoding of a localized, cropped instance of text (a word image) into the string of characters depicted. While these two stages are not necessarily wholly distinct (i.e. the result of the text recognition stage can be used to further improve the initial results of the text detection stage), it is proven that this separation makes sense computationally, since the inference associated with detection is often very different and less computationally expensive than that associated with recognition [4, 7].

Detecting text in natural scenes is a difficult task. Natural scenes contain lots of different types of objects, both text and non-text objects, which are composed of many small shapes and strokes. When looking at text and characters, the simple shapes that represent them are actually very common in natural scenes, and so it is very difficult to discriminate between text-like objects in images that are due to text and those that are products of background objects and imagery. This leads to the problem of false-positive text detection, and so a major challenge is to correctly disambiguate background noise from true text and create a system with *high precision*. Another aspect of the problem is to ensure that all instances of text are correctly found and localized rather than rejected as background noise, which is especially hard when characters are occluded or imaging noise, lighting, and texture distort their appearance. Seeking a system which finds all instances of text warrants a system with *high recall*. Achieving an ideal high precision, high recall text detection system is therefore a big challenge [4, 5].

To perform text recognition, a system must correctly identify the sequence of characters depicted in a word image. This is not a trivial task: text in scenes comes from a huge variety of different fonts or is even handwritten, so characters can look vastly different. In addition to the issue of different fonts and visual styles, the text is rendered on different surfaces, with different textures and colors, and can fall

under varying lighting conditions across a single word or character, creating varying visual properties and distortions of the characters. The camera taking the image is usually not front-parallel with the surface the text is rendered on, meaning there are perspective distortions, as well as rotation or curvature to the baseline of the text, and the limit of the camera itself means text may be blurred, noisy, and low resolution. These issues pose a significant challenge to text recognition [4, 5].

CRITERIA FOR ALGORITHMS TO BE USED IN EMBEDDED DEVICE

Two types of criteria should be considered for the detection and recognition algorithms to be used on a mobile device: the first for detecting text, and the second one for the execution time of a given task — detection and recognition of text (this is particularly important in real-time systems and similar to them).

To understand the first criterion, let's consider a simple example where the program is supposed to detect circles in a set consisting of circles and other geometric figures. It was assumed that in the whole set there are a total of 12 circles and 10 other figures. The program detected 8 elements, 5 of which are circles, and 3 are other figures, as shown in fig. 1.

Objects detected by program are called *positives*. Others objects (not detected) are *negatives*. If the program detected the right object, then this object is called *true positive*, otherwise it is *false positive*. Similarly, relevant objects that were not detected are called *false negatives*, and wrong objects that were not detected are called *true negatives* [6].

For the first criterion, three parameters are analyzed:

a) precision p, i.e. the ratio of the sum of the detected relevant elements to the sum of all detected elements (i.e. the sum of true positives and false positives). In the analyzed case, 5 significant elements were detected from 8 detected at all, i.e. $p = 5/8$;
b) recall r (sometimes called sensitivity) is the ratio of the sum of the detected significant elements to the sum of all relevant elements that were available (i.e. the sum of true positives and false negatives). In this case, 5 out of 12 circles were detected, hence $r = 5/12$;
c) $F1$-score (or f, sometimes also referred as F-score) is a measure of effectiveness that takes into account two preceding parameters p and r and is defined as:

$$F1 = 2 * p * r/(p + r) \quad (1)$$

and can be interpreted as a weighted average of the two component parameters. In aforementioned case $F1 = 1/2$.

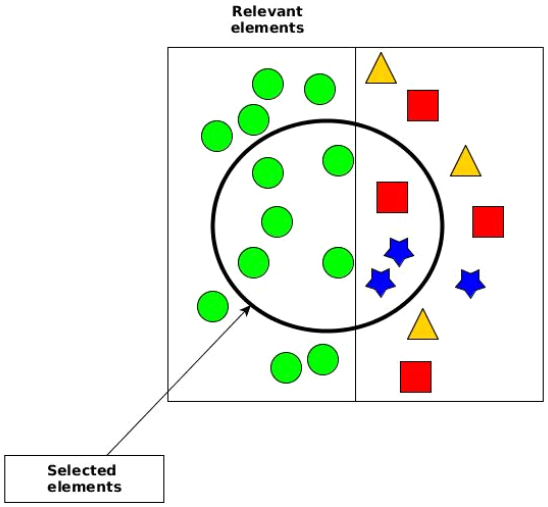

Fig. 1. Precision and recall

As for the second criterion, the time of operation of an algorithm depends, of course, on its computational complexity (the number of operations required), but also on the hardware platform and processing units available on it, such as standard CPU or GPU, ARM, DSP, and so on. In the articles the most commonly used parameter is a processing time of a single image by the given algorithm. In the case of analyzed algorithms, this is the time of detection t_d (or localization t_l), the time of recognition t_r and the total time t, which is the sum of the previous two times.

THE METHOD OVERVIEW

The system of text spotting is divided into two parts: text detection and text recognition, as shown in fig. 2. Input is represented by a 3-channel image. After finding interest points (so-called *keypoints*) of the image and creating character candidates, every candidate goes through a classifier, which makes the decision character or no-character. All properly classified characters are stored and processed by a text grouping algorithm, which merges similar characters into words. Finally, every group of text is tilted to a horizontal position, cropped and fed into the recognition system. The result of every recognition is saved for later use in the consecutive stage. In our case it will be a Text-To-Speech system (TTS). Every part of the system is described in the following sections.

Target platform for this system is Jetson TX2 provided by NVIDIA Corporation [9]. This module was chosen due to its computing capabilities, proper size

allowing building mobile device, good technical support from company and rich documentation. Algorithms will be utilizing TensorRT software, created especially for this platform. GPU (Graphical Processing Unit) unit of Jetson allows faster computing of some algorithms used in the system. It is worth noting, that not every algorithm is suitable for being executed faster on GPU than on CPU (Central Processing Unit). Algorithms used in this system were investigated on both units with real data and the tasks execution split (CPU or GPU) was done [13].

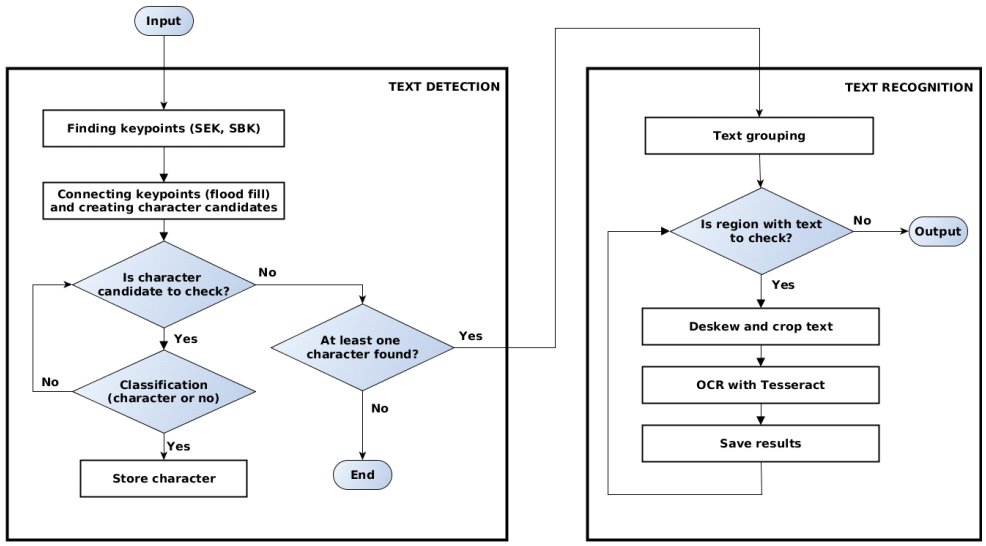

Fig. 2. System of text spotting

TEXT DETECTION

Scene text detection is a complex problem and several strategies have emerged in literature. Generally detection of an object consists of localization of this object followed by its classification. Potential text candidate should be localized first, and then the classifier marks it as a character or not.

The first approach in text detection is based on a sliding window which is shifted across the image and at each position the presence of a character or a word is checked by a classifier. The main drawback of these methods is that the number of windows that need to be evaluated grows rapidly if text with different parameters (scale, rotation, aspect) has to be found. Typical processing time ranges from tens of seconds to minutes per a single 1MPx image. This is the main reason why the region-based approach has become increasingly exploited as text of different

parameters can be detected in a single or only a few passes. Despite being faster than sliding-window methods, the fastest region-based methods have running times ranging from half a second to a second per a 1MPx image. The main cause is that the region detector is not text-specific and therefore the false detections require an additional classification step, which slows down the processing [2].

The proposed keypoint detector is inspired by FAST ext detector, presented in [2], which was the first text-specific detector with good results. Considering we are only interested in detecting character strokes, two keypoint classes are borrowed from [2]: the Stroke Ending Keypoint (SEK), which fires on a stroke ending, and the Stroke Bend Keypoint (SBK), which fires on a curved segment of a stroke. Fig. 3 presents our version of detector (on the left), and the version from [2] that we developed (on the right).

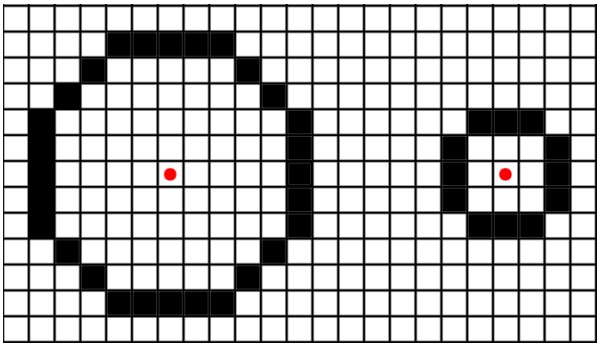

Fig. 3. Keypoint detectors

Red dot inside is a currently checked pixel, and the black pixels on the circle are being checked in the first instance. Detailed procedures for both detecting SBK and SEK are described in [2]. Example of using detector is shown on fig. 4.

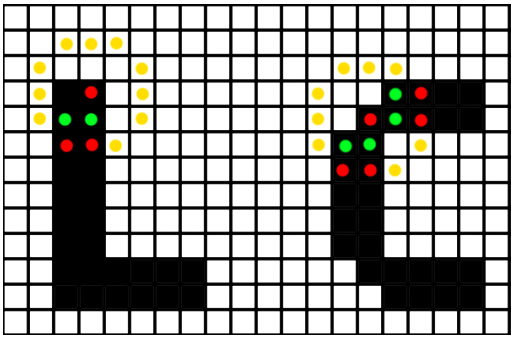

Fig. 4. Stroke Ending Keypoint and Stroke Bend Keypoint detectors [2]

Intensities of the circle pixels are compared with intensity of center pixel, and every pixel form the circle gets a label: similar, darker or brighter. If the combination of pixels amount and intensities is proper to assign center pixel flag SBK or SEK, then a connectivity test must be done, to be sure that center pixel connects with similar pixels from the circle. Red dots on fig. 4 are similar pixels, which is a necessary condition to assign SBK (or SEK) to central pixel. Yellow dots indicate pixels that are brighter (or darker in other cases). Green dots show pixels that must be checked in connectivity test between central pixel and pixels on the circle. Example of finding keypoints is presented in fig. 5.

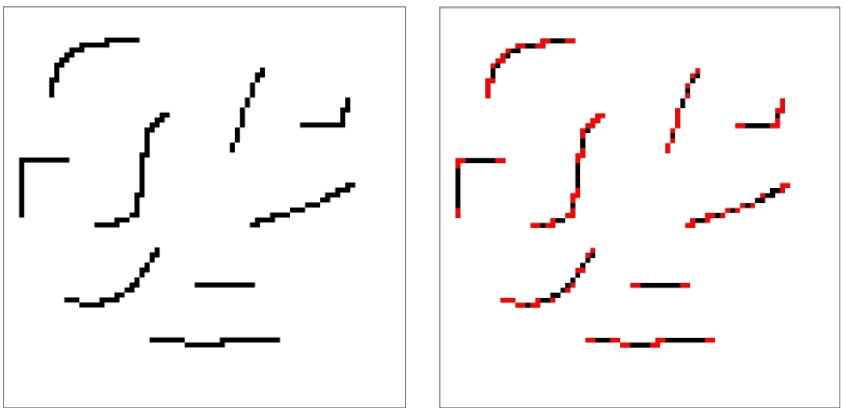

Fig. 5. Original image (left) and all detected keypoints (right)

Circle of pixels from [2] around central pixel consists of 28 pixels in our case and more false candidates are rejected with increased precision and recall. It allows us to detect more sophisticated combinations of pixels, that lead to find potentially more reliable candidates for character. Image is scaled up to 8 scales, and detector works at every scale to find candidates. Red points on fig. 5 are detected keypoints (both SEK and SBK) according to aforementioned rules.

Only one keypoint on the bend or on the end of the character is enough, and in order to remove redundant keypoints that lie close to each other, non-maximum suppression (NMS) technique is applied. Keypoints with similar properties can be connected, if the properties of pixels between them are similar. It leads to creating a character candidate by using a *flood-fill* algorithm. Other keypoints met during work on these algorithms could be merged to current process of character candidate creation.

CHARACTER CLASSIFICATION

We decided to generate binary masks of the character candidates during work on flood-fill algorithms. It allows for faster computations in further stages. Fig. 6 presents masks obtained after this process, both character masks (third column) and noncharacter masks (second column).

Fig. 6. Original image (left) and chosen generated masks (right)

For this particular image the algorithm generated total 354 masks: 179 for non-characters and for 175 characters (we had few scales where detector works).

In order to choose a classifier we compared different types of them with *F*1-score and working time as a criteria. Tab. 1 shows the results of comparison. Training and test sets consisted of masks transformed by Local Binary Pattern (LBP) operator [10]. If the first number next to neural network is 59, then it is LBP uniform pattern. If 255, then it is standard LBP. In tests 1–4 the LBP uniform operator was used, in tests 5–8 the LBP standard operator was used with 30 k training samples and 8 k test samples in both cases. In tests 9–13 both LBP versions were used with 36 k training samples and 3.6 k test samples. Test 14 was performed on GPU (previous tests on CPU) with image as a input.

Tab. 1. Classification time and *F*1-score

#	Type of classifier	Time [ms]	*F*1-score
1	NN 59:50:2	25	0.90
2	SVM	1603	0.75
3	Boost	10	0.86
4	kNN	6881	0.87
5	NN 255:50:2	58	0.91
6	SVM	3809	0.83
7	Boost	12	0.88
8	kNN	24328	0.89
9	NN 59:100:2	39	0.92
10	NN 59:100:100:2	127	0.90
11	NN 59:200:2	83	0.92
12	NN 255:500:2	292	0.90
13	NN 255:1000:2	562	0.90
14	CNN image_size:32:64:128:6272:2	–	0.98

NN — Neural Network, where first number is an input size (length of LBP vector in this case), last number is an output size, and between there are sizes of hidden layers,

CNN — Convolutional Neural Network, where *image_size* is an image dimension, last number is an output size, and between there are numbers of filters in convolution layers; example of CNN architecture, that was used during tests is presented on fig. 7,

SVM — Support Vector Machine,

kNN — k Nearest Neighbours,

Boost — Adaptive Boosting (AdaBoost).

For LBP samples the Boost algorithm was fastest, but NN produced the best *F*1-score. Changing number of layers and their size didn't change the *F*1-score significantly. For CNN the *F*1-score was the highest among all classifiers, and its speed was increased via GPU. Time is not given, because this test was performed under different conditions than previous.

Architecture from fig. 7 was created with TensorFlow framework and visualized with TensorBoard [11]. Highlighted area is a zoom of the first convolutional layer.

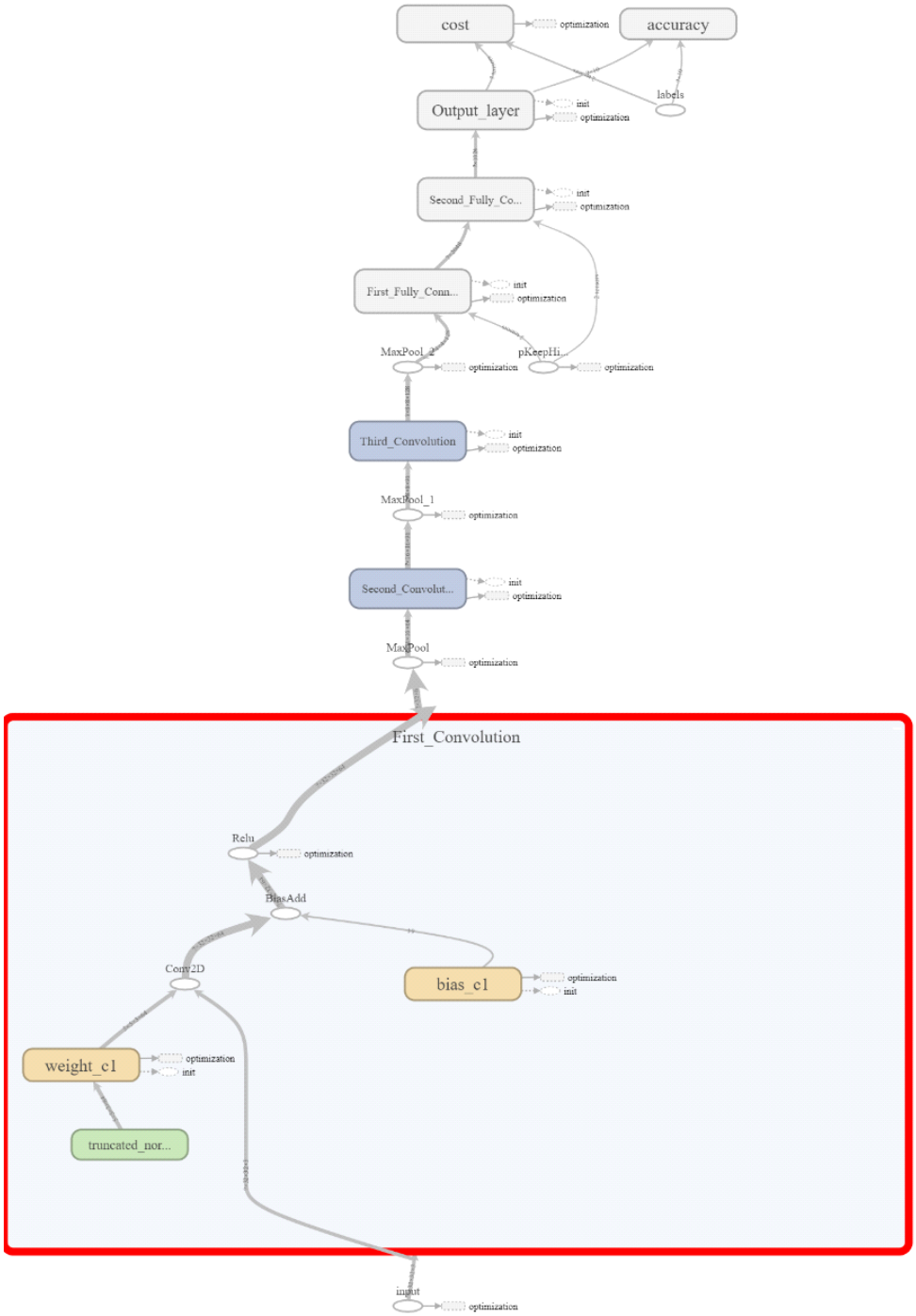

Fig. 7. Architecture of classifier based on CNN

TEXT GROUPING

After classification of character candidates, we get a set of characters that must be grouped in words. It is observed, that cohesive characters compose a word or sentence sharing similar properties such as spatial location, size, color, and stroke width regardless of language [3]. Fig. 8 presents blurred image with text oriented in two directions.

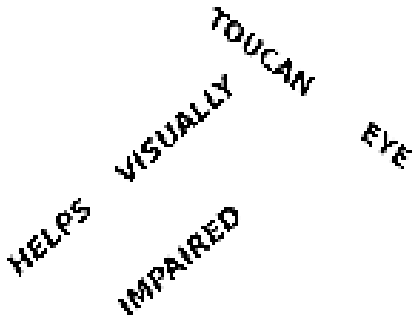

Fig. 8. Original image

Having characters, their contours are computed. Basing on contours, their minimal area rectangles are computed. It is a rectangle that encloses given contour with minimal area, that is shown on fig. 9.

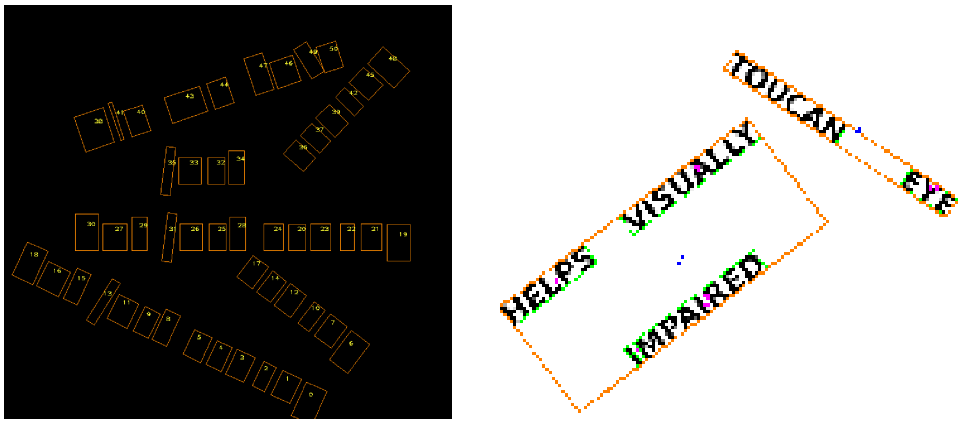

Fig. 9. Contours of characters (left) and result of final grouping (right)

Few heuristics are used to compare two candidates to be joint. Specifically, we compare two candidates on spatial location (distance between characters, lying

on one line etc.), size, angle and aspect ratio. If they satisfy the given properties, then we group them into the same word. To provide compact bounding boxes as output, we compute the minimum-area encasing rectangle [3]. We do not estimate the bottom or center line of characters. Instead, we estimate the regression line that passes through the centers (or their neighborhood) of characters. This line is updated after adding another character to the group. Result of grouping characters is presented on fig. 9. Orange line indicates groups that were connected during so-called *second pass* algorithm, that connects previously separated groups.

TEXT RECOGNITION

As a text recognition tool the Tesseract engine was chosen, due to its accuracy, and possibility of training, support for many languages and rich documentation. Newer version of this library is based on Long Short Term Memories (LSTM) neural networks, which increases accuracy, comparing to older engine versions [12].

Every group of text is first tilted to horizontal position, which is necessary step for Tesseract. Tab. 2 shows results for image from fig. 8. Confidence is a number reported by Tesseract, which shows how 'sure' engine is about given text. The great advantage of Tesseract, is that it can be trained with new data to increase the level of accuracy.

Tab. 2. Text recognition by Tesseract

Tilted and cropped image with text	Confidence	Recognized text
	97%	Toucan Eye
	96%	Helps visually impaired

It is possible to use synthetically created data for Tesseract, with different fonts, characters, sizes, lightning conditions etc. The results of the recognition stage is stored and can be passed to the consecutive stage. In our case, it will be a TTS system, who's role is to 'read' the text and play it in the headphones of the user.

CONCLUSIONS

The paper constitutes the first stage of the research and experiments carried out towards building a real embedded device, that should recognize text, translate it to speech and work in real-time conditions, within the project entitled 'Artificial intelligence system assisting persons with impaired sight — Toucan Eye' financed by National Center of Research and Development. The research described in this paper was focused mostly on software functionalities and building proper Computer Vision and Image Processing algorithms.

As a result of these experiments our algorithms can detect character candidates in natural images with high accuracy. These candidates are classified, grouped and then recognized, and finally can be passed to other systems in another form. There are also other algorithms, that we investigated less, but they will be used in the final device. For example, if we are sure that a given picture includes clear area with text (receipt, menu or book page), blob detection or Hough transform can be used to deal with text.

Despite the good results which were achieved so far, they do not satisfy the project team and generally they require improvement and further work. Many efforts will be made into optimizing the performance of developed algorithms onto a target mobile platform — JetsonTX2. Also the method of choosing a given text detection algorithm should be devised, because there is no algorithm that will be optimal in all situations.

Acknowledgments

The research presented in the paper were funded by Polish National Center of Research and Development within the project No. POIR.01.01.01-00-0910/16, entitled 'Artificial intelligence system assisting persons with impaired sight — Toucan Eye'. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

REFERENCES

[1] Busta M., Matas J., Neumann L., *Deep TextSpotter: An End-to-End Trainable Scene Text Localization and Recognition Framework*, IEEE, International Conference on Computer Vision, 22–29 October, Venice, Italy, 2017, pp. 2223–2231.

[2] Busta M., Matas J., Neumann L., *FASText: Efficient Unconstrained Scene Text Detector*, IEEE, International Conference on Computer Vision, 13–16 December, Santiago, Chile, 2015, pp. 1206–1214.

[3] Cho H., Sung M., Jun B., *Canny Text Detector: Fast and Robust Scene Text Localization Algorithm*, IEEE, Conference on Computer Vision and Pattern Recognition, 26 June – 1 July, Las Vegas, USA, 2016, pp. 3566–3573.

[4] Jaderberg M., *Deep Learning for Text Spotting*, PhD thesis, Visual Geometry Group, University of Oxford, UK, 2015.

[5] Neumann L., Matas, J., *Efficient Scene text localization and recognition with local character refinement*, International Conference on Document Analysis and Recognition, 23–26 August, Nancy, France, 2015, pp. 746–750.

[6] Powers D., *Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation*, 'Journal of Machine Learning Technologies', 2011, Vol. 2, Issue 1, pp. 37–63.

[7] Ye Q., Doermann D., *Text Detection and Recognition in Imagery: A Survey*, IEEE, 'Transactions on Pattern Analysis and Machine Intelligence', 2015, Vol. 37, Issue 7, pp. 1480–1500, DOI: 10.1109/TPAMI.2014.2366765.

[8] Yin X., Zuo Z., Tian S., *Text Detection, Tracking and Recognition in Video: A Comprehensive Survey*, IEEE, 'Transaction on Image Processinges', 2016, Vol. 25, Issue 6, pp. 2752–2773, DOI: 10.1109/TIP.2016.2554321.

[9] *Jetson TX products*, [online], NVIDIA, https://www.NVIDIA.pl/autonomous-machines/embedded-systems-dev-kits-modules/ [access 27.07.2018].

[10] *Machine Vision Group at University in Oulu*, University in Oulu, [online], http://www.cse.oulu.fi/wsgi/CMV/Research/LBP [access 27.07.2018].

[11] *TensorFlow framework*, TensorFlow, [online], https://www.tensorflow.org/ [access 27.07.2018].

[12] *Tesseract engine*, Github, [online], https://github.com/tesseract-ocr/tesseract [access 27.07.2018].

[13] *Toucan Eye system*, Tucan systems, [online], https://toucan-systems.pl/toucaneye/ [access 27.07.2018].

DETEKCJA I ROZPOZNANIE TEKSTU W WARUNKACH NATURALNYCH ZA POMOCĄ URZĄDZENIA PRZENOŚNEGO

STRESZCZENIE

Autorzy artykułu w ramach projektu naukowo-badawczego przeprowadzili badania z użyciem przenośnego urządzenia z systemem sztucznej inteligencji Toucan Eye, które może pomóc osobom z wadami wzroku. Prawidłowo wykryty tekst przekazywany jest do etapu jego rozpoznawania,

a następnie do systemu syntezy mowy. W artykule zostały pokazane poszczególne etapy pracy systemu Toucan Eye oraz opisane wybrane metody, których celem jest wykonanie zadania detekcji i rozpoznania tekstu w warunkach naturalnych.

Słowa kluczowe:
wykrywanie tekstu w warunkach naturalnych, wspomaganie osób z wadami wzroku, Toucan Eye.

Article history
Received: 03.08.2018
Reviewed: 20.05.2019
Revised: 07.06.2019
Accepted: 27.06.2019