Automatic License Plate Recognition in Difficult Conditions – Technical Report

Petr Hurtik and Marek Vajgl
University of Ostrava, Centre of Excellence IT4Innovations
Institute for Research and Applications of Fuzzy Modeling
30. dubna 22, Ostrava, Czech Republic
petr.hurtik@osu.cz, marek.vajgl@osu.cz

***Abstract*—By a car license plate recognition, we mean a software system processing images and providing an alphanumeric transcription of car plates included in an image. We divide the task into four sub-tasks: license plate localization, license plate extraction, characters segmentation and characters recognition. All four sub-tasks are discussed in the context of standard approaches and own solution based on a chain of standard and soft computing image processing algorithms is presented. In this chain, the F-transform approximate pattern matching algorithm plays the crucial role. For the solution, we presented recognition ability for a dataset which includes 500 images with difficult conditions.**

1. Introduction

A car plate recognition application detects and reads license plates on cars in different conditions. It assumes a general image as the input, where one or more cars are captured. As the output of the car plate recognition application, we expect one or more alphanumerical strings which match with the car plate text captured in the input image. Such a system real usage is in handling road tools, searching for stolen cars etc.

A process of the car plate recognition can be split into four sub-tasks, namely *car plate localization*, *car plate extraction*, *characters extraction* and *characters recognition*. These sub-tasks can be explained as follows. As a considered camera captures not only a car plate but also its surrounding including a whole car, a road, vegetation, etc., a captured image has to be analyzed and potential image areas where a car plate can be presented have to be determined and marked. This step is called *plate localization*. The second step, *car plate extraction*, is focused on a processing of the marked areas. A marked area is typically bigger than the plate is, so it includes surrounding. Moreover, because the image area is affected by a perspective, the plate extracted from the marked area has to be also perspective adjusted. The obtained extracted area with a plate is an input for the next step, *characters extraction*. This step splits an image (the area with a car plate) into several disjoint parts, where each part contains exactly one character. These parts with the characters are recognized in last step – *characters recognition*.

In this paper, we will describe how our system for car plate recognition used in a commercial application was designed. The paper novelty is presented in two aspects. The first one is that instead of one method for *car plate localization*, an investigation of several methods is realized and their ensemble is established. We demonstrate that even trivial methods can achieve high accuracy of car plate localization if they are used together. The second novelty is in the usage of F-transform pattern matching algorithm for *characters recognition* where we show how a theoretical-based tool can be used in a real application where it achieves high success-rate and computation speed.

The structure of the paper is as follows. At first, Section 2 introduces the project to be solved, its restrictions and used data. The main merit is described in Section 3, where the four particular sub-tasks are explained and the achieved results for the particular sub-tasks are demonstrated. In Section 4 we discuss some minor difficulties and we compare obtained results with one state-of-the-art solution. And finally, Section 5 summarizes the paper and talk about a future work.

2. The car license plate recognition project

Our motivation for the research in this area has been a commercial project scanning and recognizing car plates of cars going beneath a scanning device. Scanning is done using common hardware and the offered solution has to beat currently used commercial approach. The project configuration is as follows. A camera is placed on a highway gate and shoots traffic from the top, where one image contains generally no one, one, or more cars. The system runs non-stop, so it has to process images in miscellaneous day / night / weather conditions. Especially, night winter time is a very difficult case for common systems due to snow or ice covering a car plate. Also, the real mentioned hardware system consists of several cameras with different resolution varying between 1392×1040px and 2048×1088px. All cameras provide only gray-scale images without color information. The system is situated in the Europe, so all Euro-Asian license car plate variations may occur. The time of processing one image should not exceed one second on a standard computer without GPU processing.

978-1-5090-4917-2/17$31.00 © 2017 IEEE

Such a commercial project has, unfortunately, two restrictions. The first one is the used hardware: it is not possible to replace current cameras by some specific, expensive cameras and therefore the quality of images cannot be significantly improved in this way. In the same time, the current computation server has to be used. Let us remark, this server does not perform GPU operations, therefore usage of deep learning etc. is omitted. The goal of the project is to process captured images, not to design a new hardware system.

The second restriction is in project data. All tests and deployments have been realized by the second subject in their offices and haven't been anyhow redistributed. Hence all images presented in this paper are only illustrative simulations of the official data used. No matter the combination of a car snapshot containing the car plate, without the possibility of identification of the driver, place and time of the photo acquisition, is not in the project context a subject of the personal data classification, we, as the researchers and solvers, were facing direct inaccessibility to the data.

Because of the second restriction, we created 500 images simulating the real project images by our own. Two of them – for night and day – are shown in Figures 1 and 2. The test set reflects different image resolutions mentioned above, are gray-scale and software modified to simulate night / rain / snow etc.

Figure 1. Top: an example of an image which is difficult to recognize due to night conditions. Bottom: manually extracted and binarized car plate in order to clarity.

3. Automatic License Plate Recognition

In this section, we are going to explain how the sub-tasks in the automatic license plate recognition are designed.

3.1. License plate localization

Car license plate localization is a process which takes an image with a whole scene as an input and marks areas where license plates may appear. Let us explain how is the process solved in a literature. The most trivial solutions are based on edges. e.g., [1]: an image is converted into a grayscale image

Figure 2. Top: an example of an image taken during the day where the car includes several pieces with text. Bottom: manually extracted and binarized car plate in order to clarity.

and a gradient magnitude is computed. The neighborhood of pixels with the strong gradient magnitude in the vertical direction defines possible areas. However, this approach is not general enough if an image contains different vertical edges. The interesting idea based on local surrounding is used in [2] where so-called line segments are taken and the segments with a certain number of intensity jumps are taken as candidate plate areas. The gradient magnitude is also used in projection histograms [3] where an image with the gradient magnitude is projected into a horizontal or vertical histogram and a text pattern is searched.

The third way using the gradient magnitude approach is HOG [4]. From the gradient, descriptors are designed and the searching is realized by some machine learning algorithm, usually SVM [5]. A similar idea of building descriptors is also used in LBP [6] or SIFT [7] approaches. The different way of searching a car license plate is to search for characters directly. Recently, this approach was realized by Convolution Neural Network [8]. The disadvantage of using the Convolution Neural Network is a need of huge learn set which we do not have. In the case of searching characters directly without the deep-learning, it is necessary to face a standard computer graphic problem called "text in the wild". Such a problem can be possibly solved by *Stroke Width Transform* technique [9], if two related issues will be solved: at first, improving the SWT small characters detection ability; and the second, detecting of a lot of characters in the image (different/multiple signs placed onto cars) leading to a high number of potential candidates. The case, when an image can include a lot of text, is illustrated in Figure 2.

Our proposal is to take several trivial, fast methods in order to detect car plate position and evaluate their success rate. We chose such the methods which output can be represented as an image. From the methods, their output image aggregation is established and it serves as a heat-map where pixels intensities represent "weight" of a car license plate occurrence. The list of tested methods is following:

W1, 2 *Gradient magnitude*. The gradient magnitude is detected by Sobel operator separately for vertical and horizontal directions.

W3 *Projections histogram*. Vertical and horizontal histograms are computed and their values visualized back into the image.

W4 *Simplified stroke width detection*. Each pixel is inspected, horizontal and vertical lengths of the similar intensities of surrounding pixels are measured. Only pixels with defined horizontal and vertical lengths are marked.

W5 *Possible occurrence weighting*. Although a car plate can occur in an arbitrary image position, the camera shoots cars in a predefined way where the car plates occur mainly around "center" of the image. Therefore we give a bigger weight to the center of the image.

W6 *Fuzzy transform searching*. Fast, general method [10] for pattern matching has been adopted. The method tries to search a one general car license plate in the given image. the output of the method, locations with success rate, is projected back to the image.

W7 *Edge detection*. From the gradient magnitude image, only 1px edges are marked.

W8 *Rectangle searching*. Using the edge output, edges are traced and objects with a rectangular shape are marked.

W9 *Line segments*. Realization of Zhao's algorithm [2].

Given the list of algorithms, particular output images are taken and areas with the sum of intensities inside higher than a threshold are taken as candidate areas where car plate should be localized. The success rate of car plate localization is following: $W1 = 88.0\%$, $W2 = 89.0\%$, $W3 = 84.6\%$, $W4 = 89.0\%$, $W6 = 92.4\%$, $W7 = 87.4\%$, $W8 = 68.8\%$, $W9 = 90.8\%$. Because $W5$ is only a static image, it is not classified. From the list, we decided to omit $W3$ and $W8$ due to their poor score. Although the average success rate is **89.4 %** and the highest individual one is **92.4 %**, by aggregating the methods into one heat-map and searching for the car plate localization within it, the success rate rises to **97.2 %**. As we use only fast trivial methods, the final method remains fast too. The additional advantage is that methods can be easily added/removed. The heat-map is visualized in Figure 3 and as you can see in the image, the car license plate is detected even if it is almost invisible for a human.

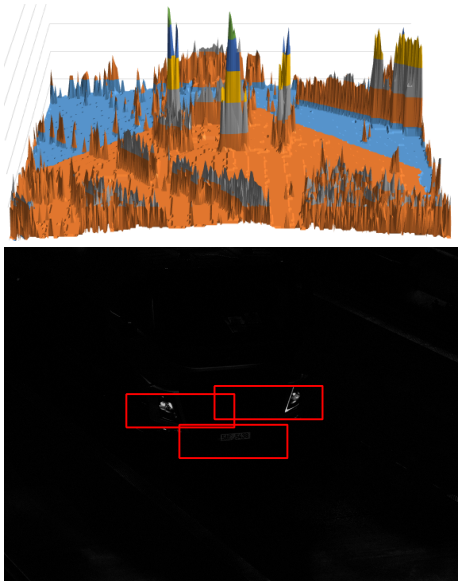

Figure 3. Car plate localization. Top: the heat-map. Bottom: visualization of areas which are marked as candidate ones.

3.2. License plate extraction

As the input into this step, we consider a set of n candidate areas in the form of images taken from the *car plate localization* step. The output of this step is a set of m images containing only a license plate without background. For the input/output holds $m \leq n$, i.e., the candidate areas without a license plate are excluded. In the existing solutions, the car plate is usually localized and extracted at once. If a car license plate has to be extracted separately, the Hough transform [11] can be used well. The Hough transform approach works as follows: an image is prepossessed by a gradient operator, transformed into a dual Hough space and four perpendicular points with the highest weight are searched. An application of the Hough transform as a license searcher can be seen in [12]. However, in our case, we have found that this approach fails when an input image contains some parallel lines near a car license plate such as a structure of a car mask.

In our application, we propose the sequence of steps described below. For the simplicity, the partial step output is also visualized in Figure 4.

1) *Image preprocessing.* The image is normalized according to its histogram.
2) *Adaptive thresholding.* We choose a threshold equal to the preprocessed image intensity of 0.9 percentile and set all the pixels with lower values to 0, i.e., black.
3) *Image repairing.* This step has as the input result of Step 1. We establish a 3×3 neighborhood around each pixel with intensity lower than 0.9 percentile. If there are at least four pixels with intensity higher than 0.6 percentile, the pixel intensity is projected into the thresholded image (output of Step 2). This is similar to a dilatation process, but in this case, the object border does not expand so much.
4) *Cluster analysis*. Using connected component analysis [13], non-black clusters are detected. Clusters whose are too close to each other are merged. After that, the biggest cluster is taken, the rest are removed. Then, the result is processed via "Image repairing" as is described in Step 2, but with 0.5 percentile threshold.
5) *Corners detection*. The most top-left, top-right, bottom-left and bottom-right cluster points are detected. The corners positions are inspected: if they are too close, or they do not form "rectangle" (we allow small perspective deformation of a rectangle), the area is excluded. Otherwise, the corners are modified in order to obtain perfect parallel lines.

6) *Perspective correction*. Using homography (for details see chapter homography in [14]) and the known corner points, the perspective distortion is removed.
7) *Cropping*. The image is cropped in order to contains only the detected car plate.

The whole sequence of the car plate extraction is visualized in Figure 4. We tested the sequence using the 500 images dataset and we measured **97.8 %** success rate of car plate extraction.

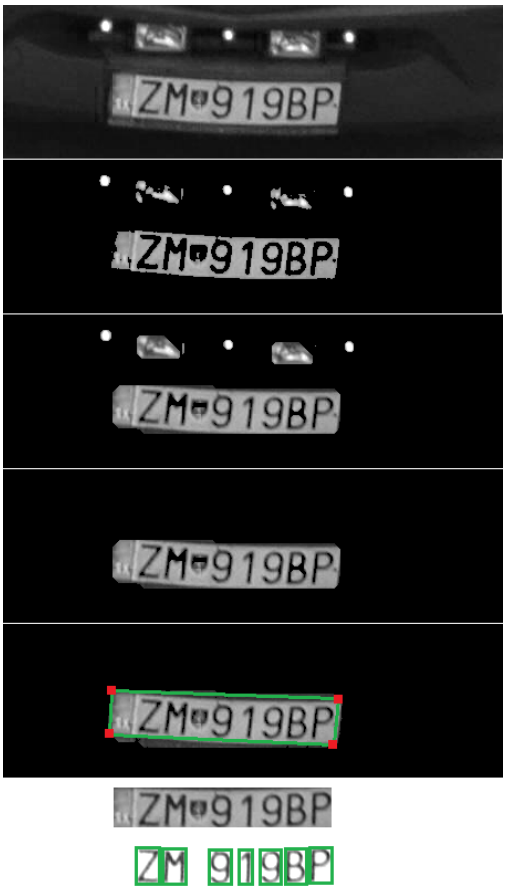

Figure 4. Seven images illustrating the car plate extraction process. From top: normalized image; thresholded image; repaired image; small clusters removing; corners detection; perspective correction and cropping. The seventh, most bottom image visualizes the output of the character segmentation process.

3.3. Characters segmentation

Extracted car plates can be recognized in two different ways. The first one splits an extracted plate into a set of characters and each of the characters is recognized separately. The second one looks for and recognizes the characters at once in the whole image. Because of an extracted plate can still contain a lot of scratches etc., we decided to realize the first type of character recognition. Therefore, we are going to describe the process of characters segmentation at first. The goal of the chosen reading type – segmentation – is to take an extracted car plate and extract characters from it one by one.

We can find several connections between the problem of character segmentation and a general Optical Character Recognition (OCR) approach. In both cases, we need to separate characters from a background. The separation is usually established by an adaptive (local) thresholding [15]. In our case, we have an expert knowledge that the characters cover approximately 5 % of the whole plate and the characters should be dark on a light background. Therefore, we use a thresholding with a value of 0.05 percentile of the image intensities. If a pixel intensity is higher than the percentile, it is changed to white, otherwise, we preserve the intensity.

Further, in the case of license plates covered by snow or debris, it may happen that the characters will not be separated correctly. Therefore, we propose a sequence of the following steps:

1) By using connected component analysis we detect blobs of non-white pixels and establish a rectangle over each of them.
2) Rectangles which are too close to each other are merged.
3) Rectangles which are too high or too small in comparison with the rectangles median height are removed.
4) Rectangles which are too thin in comparison with the rectangles median width are removed.
5) Rectangles which are too wide in comparison with the rectangles median width are split into several rectangles.
6) Rectangles which have too low gradient magnitude in their area in comparison with the rectangles median gradient magnitude are removed.
7) We search for large areas not occupied by other rectangles. If found, we try to establish a new rectangle there. The check of the width, height and gradient magnitude is realized for the newly established rectangle.
8) Finally, each pixel in each rectangle is replaced by a pixel from the original non-thresholded image and then the percentile thresholding for each of the rectangles is applied separately.

The character segmentation process as is described has several benefits. The first one is that we work with percentile values instead of fixed thresholds. Therefore, the algorithm is invariant to light conditions. The second one is that the algorithm offers rectangles positions which can be used later in order to determine a license plate country. For the illustration, the last but one image in Figure 4 shows input into the character segmentation step and the last image in the same figure shows the output of the segmentation process.

In our 500 images experiment, we consider as an successfully segmented characters only those which have perfectly removed background, the rectangle over character perfectly fits and none of character part is missing. As a bad characters we consider characters which do not satisfy above mentioned conditions, non-detected characters and false detected characters. The success rate of character segmentation is **95.5 %** for particular characters and

86.4 % for a whole license plate (that is all characters in a certain plate are correct). Let us remark, the success rates were computed even for non-successfully extracted plates. The most common reason why the segmentation fails is a detection of non-character parts. On the other hand, these parts are excluded in the *Character recognition* step.

3.4. Character recognition

By a recognition, we mean a process which assigns a label to each segmented character represented by an image. Nowadays, the recognition can be solved by a lot of approaches. One of the most commonly used is Tesseract [16] which creates a polygonal representation of each character and try to match this representation with a known database representation. The problem is that the Tesseract cannot be used freely for a commercial application. To be complete, a comparison with a solution where the Tesseract is used is realized in Section 4.

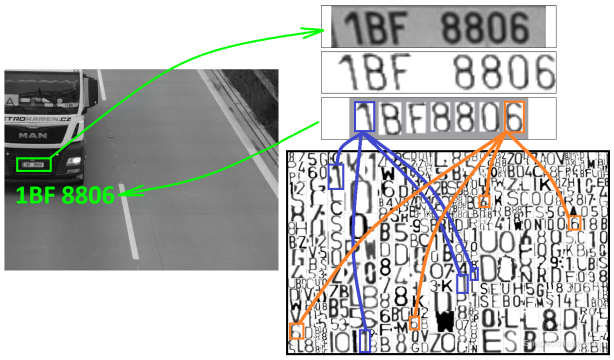

Figure 5. Car plate recognition process. The localized plate is extracted, characters are segmented and matched within the database letters. Each character has assigned label due to matching output.

In our approach, we have adopted the algorithm for a pattern matching described in [17]. Let us remark, this is the first usage of the algorithm as a classification tool and its first usage in a commercial application. In this approach, we consider a so-called database consisting of characters represented by images. Furthermore, each character in the database has a known label. At first, the algorithm preprocesses the database in the way that input images are transformed into so-called F-transform components and the components are stored. Then, a pattern – a character represented by an image – is transformed into F-transform components too and distances between the pattern and the database F-transform components are computed. In our case, we experimented with a various number of chosen characters and the best result was achieved by choosing the five most similar database characters with their labels. If all those database characters have the same label, we assign such label to the pattern character. If not, we try to shift the character rectangle by several pixels in each direction (for the case if the character segmentation was not perfect) and recognize each shift combination separately. Finally, the combination of the most equivalent labels is taken and the most frequent label is assigned to the pattern character.

As the database, we artificially created 28415 labeled characters. Then, for each character, we created four copies where each copy was brighter or darker than the original. From this, we have obtained the database of 142075 characters. The main advantage of the F-transform matching algorithm is the computation speed and allowance of parallelism. We consider several (four, in average) car plate readings per image, seven characters per plate (in average) and database with 142075 characters. So, we make approximately $6 \cdot 10^7$ character comparisons per one image. With the algorithm processing speed, the computation of all $6 \cdot 10^7$ character comparisons takes approximately only 300 ms on a standard desktop. The computation performance allows us to have a huge characters database including several various font types, sizes, character distortions, illuminations etc. A number of various types make the algorithm robust and general. For the 500 images dataset, the success rate of whole car plate recognition (reading) is **85.8 %** and includes plates which were not successfully extracted and characters which were not successfully segmented. One example of the designed application output including recognition on standard, clear visible image is presented in Figure 6.

The most challenging remaining problem is the similarity of characters like 0 and O, 2 and Z, 8 and B etc. These incorrect classifications can be partially suppressed using additional knowledge about national car plate format and by choosing the correct character from the tuple. The additional improvement remains for a future work.

Figure 6. The example of successful recognition of standard, visible highroad image.

4. Discussion

In the previous section, we have introduced a realization of our solution for the problem of car plate detection and recognition split into four partial steps. Because of lack of space, the realization was described in the form of explanation without any deeper mathematical background. On the other hand, we are convinced that it is the paper advantage in the same time: all the steps are available in one place without confusing details.

During our work, we met with opinions that the achieved success rate is not high enough because there exist systems achieving almost 100 % success rate. Let us recall that we focused on car plates in difficult conditions where a car plate reading can be complicated task even for a human. Moreover, during our investigation, several existing algorithms were tested. The best results for this task were obtained by OpenALPR software[1]. The software is divided into the same sub-tasks as the proposed one. A car plate is located by LBP algorithm, characters are segmented by connected components analysis and recognized by the Tesseract. We have to emphasize that the OpenALPR software works well for general images but in our difficult test set the OpenALPR successfully localized license plate only in **25.6 %** of all cases and, from them, only **43.7 %** plates were read correctly. Let us recall, our system localized license plate in **97.2 %** cases and, from that, **85.8 %** were read correctly.

5. Conclusion

The general car plate localization and reading in ideal conditions are well-solved applications, so we have mentioned several approaches which can be used to solve the task. On the other side, we have dealt with a car plate reading in difficult conditions, where – according to our experience – the existing systems might fail. By difficult conditions, we mean night images, images with bad weather conditions – ice, snow, rain, images with debris, images containing multiple texts, car plates in non-defined format, or over-exposured images.

We have presented the approach which split the task into four sub-tasks: car plate localization, car plate extraction, character segmentation and character recognition. For all the sub-tasks we proposed a solution based on a connection of fast standard and soft computing image processing methods. The standard methods, such as connect component analysis and percentile thresholding, allow us to realize simple and fast processing, while the soft-computing ones bring robustness.

The main goal of the paper, a presentation of the chain of existing algorithms combined in order to obtain general car plate recognition software, has been presented. We hope it can help other researchers to adapt their own car plate recognition system.

The future work will be focused on searching for arbitrary-sized car plates. Actually, it is possible to search for $\pm 30\%$ size plate that is defined by a user. After that, a comparison of existing car plate benchmark databases will be possible. In the same time, an exhaustive comparison with existing approaches and more detailed explanation of our algorithm will be the subject of a full journal version.

Proclamation

All images in the paper were created on public roads by ourselves without archiving any location, personal or time information.

1. http://www.openalpr.com/

Acknowledgment

This research was supported by the project "LQ1602 IT4Innovations excellence in science".

References

[1] S. Saha, S. Basu, M. Nasipuri, and D. K. Basu, "License plate localization from vehicle images: An edge based multi-stage approach," *International Journal of Recent Trends in Engineering*, vol. 1, no. 1, pp. 284–289, 2009.

[2] J.-L. Zhao, H.-G. Min, X.-C. Li, and Y. Pan, "A license plate recognition algorithm under low illumination environment," in *Smart Cities Conference (ISC2), 2015 IEEE First International*. IEEE, 2015, pp. 1–6.

[3] I. Bulugu, "Algorithm for license plate localization and recognition for tanzania car plate numbers," *International Journal of Science and Research (IJSR)*, vol. 2, no. 5, pp. 12–16, 2013.

[4] K. Zheng, Y. Zhao, J. Gu, and Q. Hu, "License plate detection using haar-like features and histogram of oriented gradients," in *Industrial Electronics (ISIE), 2012 IEEE International Symposium on*. IEEE, 2012, pp. 1502–1505.

[5] X. Li, "Vehicle license plate detection and recognition," Ph.D. dissertation, University of Missouri–Columbia, 2010.

[6] L. Liu, H. Zhang, A. Feng, X. Wan, and J. Guo, "Simplified local binary pattern descriptor for character recognition of vehicle license plate," in *Computer Graphics, Imaging and Visualization (CGIV), 2010 Seventh International Conference on*. IEEE, 2010, pp. 157–161.

[7] W. T. Ho, H. W. Lim, and Y. H. Tay, "Two-stage license plate detection using gentle adaboost and sift-svm," in *Intelligent Information and Database Systems, 2009. ACIIDS 2009. First Asian Conference on*. IEEE, 2009, pp. 109–114.

[8] H. Li and C. Shen, "Reading car license plates using deep convolutional neural networks and lstms," *arXiv preprint arXiv:1601.05610*, 2016.

[9] B. Epshtein, E. Ofek, and Y. Wexler, "Detecting text in natural scenes with stroke width transform," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2963–2970.

[10] P. Hurtik, P. Hodáková, and I. Perfilieva, "Approximate pattern matching algorithm," in *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Springer, 2016, pp. 577–587.

[11] D. H. Ballard, "Generalizing the hough transform to detect arbitrary shapes," *Pattern recognition*, vol. 13, no. 2, pp. 111–122, 1981.

[12] V. Ganapathy and W. L. D. Lui, "A malaysian vehicle license plate localization and recognition system," *Journal of Systemics, Cybernetics and Informatics*, vol. 6, no. 1, 2008.

[13] A. Rosenfeld and J. L. Pfaltz, "Sequential operations in digital picture processing," *Journal of the ACM (JACM)*, vol. 13, no. 4, pp. 471–494, 1966.

[14] M. Sonka, V. Hlavac, and R. Boyle, *Image processing, analysis, and machine vision*. Cengage Learning, 2014.

[15] M. R. Gupta, N. P. Jacobson, and E. K. Garcia, "Ocr binarization and image pre-processing for searching historical documents," *Pattern Recognition*, vol. 40, no. 2, pp. 389–397, 2007.

[16] R. Smith, "An overview of the tesseract ocr engine," 2007.

[17] P. Hurtik and P. Hodáková, "Ftip: A tool for an image plagiarism detection," in *Soft Computing and Pattern Recognition (SoCPaR), 2015 7th International Conference of*. IEEE, 2015, pp. 42–47.