The Use of Tesseract OCR Number Recognition for Food Tracking and Tracing

Conference Paper · February 2020

DOI: 10.1109/IT48810.2020.9070558

Stevan Cakic
University of Donja Gorica
20 PUBLICATIONS **70** CITATIONS
SEE PROFILE

Tomo Popovic
University of Donja Gorica
162 PUBLICATIONS **1,102** CITATIONS
SEE PROFILE

Stevan Šandi
University of Donja Gorica
18 PUBLICATIONS **136** CITATIONS
SEE PROFILE

Srdjan Krco
DunavNET
107 PUBLICATIONS **3,038** CITATIONS
SEE PROFILE

24th International Conference on Information Technology (IT)
Zabljak, 18 – 22 February 2020

The Use of Tesseract OCR Number Recognition for Food Tracking and Tracing

Stevan Čakić, Tomo Popović, Stevan Šandi, Srđan Krčo, and Anita Gazivoda

Abstract – **One of the most interesting enabling technologies for digital transformation is computer vision. Object and character recognition has already become very popular and it is used in everyday life. This research focuses on the use of computer vision to read serial numbers from wine labels in order to enable applications based on tracking and tracing of each individual wine bottle. After experimenting with several OCR tools, an open source software called Tesseract OCR engine was selected for the pilot solution. The paper discusses the implementation and image processioning that improved detection accuracy. The coding was done in the Python programming language. The solution code was tested using real-life like images of wine serial numbers. In addition, a custom built web-based evaluation tool was created and used for the interactive evaluation of the system.**

Keywords – **computer vision, digital transformation, food tracking and tracing, OCR, Tesseract, wine authentication**

I. Introduction

Nowadays, we are witnessing the dramatic process of digital transformation in which digital technologies are changing how businesses connect and create value for their customers [1]. Digital transformation is gaining an important role in the agri-food sector in production and supply chain applications. Food tracking and tracing is one such trend where digital technology allows end-to-end visibility and tracking and tracing food products through their lifecycle [2]. An example of an application from this category is one that can track and trace each individual wine bottle in order to provide counterfeit prevention and brand protection [3]. In this paper we focus on the use of computer vision and optical character recognition (OCR) on mobile devices in order to read serial numbers of individual wine bottles, thus to use that capability for a wine tracking and tracing solution.

Computer vision is used for unlocking a mobile phone with face, or in applications like FaceApp, or when someone scans QR code from different products. Computer vision and artificial intelligence are used in medicine to recognize different diseases, as well as in the car production industry for self-driving vehicles where the camera detects the area around the car [4]. As the name suggests, OCR is used to recognize and extract the characters from the image [5]. These characters can be anything, from letters in different languages to numbers. In this research, a software solution capable of recognizing serial numbers from wine bottles is presented. Wine bottle serial numbers can be found on wine labels placed around the bottle. A serial number typically contains five to seven digits, sometimes with horizontal lines across them, and these lines pose the biggest challenge for our OCR solution.

The main objective of this research is to uniquely identify each individual bottle of wine using only a smart mobile phone equipped with camera. While there are various options for tagging products such as QR code, RFID tags, and other advanced tags, this would require intervention or change of the existing wine bottling process. In this research we focused on the OCR solution for reading existing wine labels that already have unique serial numbers. Please note that standard GS1 bar-codes used in retail all over the world do not uniquely identify the instance of the product, but rather a product type or a family [6]. The target application the research is to be able to use the mobile app to scan the wine, get the information about the wine in that bottle (i.e. type, vintage, origin, ratings), but also to get the info on that particular wine bottle. Wine originality authentication and counterfeit prevention would be one such benefit of having this additional information. For all of this to work the mobile app needs some intelligence and heuristics, as well as the cloud system in the back-end of the solution, but in this paper we only focus on the sensors and the use of mobile device camera to perform OCR and read the product serial numbers.

The company "13. jul Plantaže" is one of the biggest wine produces in South Eastern Europe and their production includes over 17 million bottled products annually. Within the same type, wine bottles have unique serial numbers. It may occur that two different types of wine have the same serial number, however the chances of this happening are extremely small.

II. Materials and Methods

An example of a picture of a wine serial number is show in Figure 1. This is a digital image obtained by mobile phone camera and then cropped to a size to include only the serial number. This is an almost "perfectly taken" digital image

*Research supported by the Ministry of Science of Montenegro, scientific project DIPOL, Grant Agreement No. 01-788/2

S. Čakić is with with Faculty for Information Systems and Technologies, University of Donja Gorica, Oktoih 1, 81000 Podgorica, Montenegro (e-mail: stevan.cakic@ udg.edu.me).
T. Popović is with Faculty for Information Systems and Technologies, University of Donja Gorica, Oktoih 1, 81000 Podgorica, Montenegro (e-mail: tomo.popovic@ udg.edu.me).
S. Šandi is with Faculty for Information Systems and Technologies, University of Donja Gorica, Oktoih 1, 81000 Podgorica, Montenegro (e-mail: stevan.sandi@ udg.edu.me).
S. Krčo is with DunavNET, Bulevar oslobođenja 127/13, 21000 Novi Sad, Serbia (e-mail: srdjan.krco@dunavnet.eu).
A. Gazivoda is with "13. jul Plantaže", Put Radomira Ivanovića 2, 81000, Podgorica, Montenegro (e-mail: anitagaz@t-com.me).

and most of modern OCR tools would read the numbers without any problems or needs for preprocessing. However, when taking a picture of serial number from a bottle, it is not uncommon that the numbers appear curved due to the roundness of the bottle or the angle of the camera at the moment the picture is taken. Also, the horizontal lines that are present on some labels introduce an additional challenge. These challenges are illustrated in Figure 2 where you can see how various OCR tools read the serial number that is covered with horizontal lines and curved due to the way how the picture was taken. In this particular example, none of this tools correctly read the original picture. In order to recognize and read the serial numbers from wine bottles, we need three steps: 1) image thresholding, 2) removing of horizontal lines, 3) OCR recognizing.

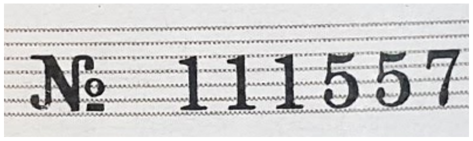


Figure 1. An example of the serial number captured by mobile phone

A. Selection of the OCR Tool, Tesseract

For the implementation of the solution we opted for Tesseract OCR engine that is free and released under the Apache V2.0 open source license. Tesseract is one of the most accurate open-source OCR engines in which development is sponsored by Google [7,8]. Before we selected Tesseract, we also tested:

- Azure Computer Vision by Microsoft [9],
- Vision AI by Google [10].

Please note that these tools are not free for commercial projects, but they can be very useful for prototyping and experimenting. These tools offer a limited number of API calls, but after that the usage must be paid for each API request. Additionally, we noticed that in some instances the Tesseract gave better results than all of the above tools.

B. Implementation Approach, Development Tools

The first step in reading the serial numbers obtained from a wine label is image thresholding that is used to remove background noise and unnecessary objects from the image. After that, a morphology operator is used to remove horizontal lines from the image. Finally, the OCR tool is used to recognize the numbers and read the whole serial number.

The OCR solution for wine bottle serial numbers was implemented using the Python programming language [11]. Python has a very strong community support with lots of helpful packages and libraries. It is also one of the most popular programming languages for data science, machine learning, artificial intelligence, and scientific computing in general.

The pyTesseract library was used as a Tesseract wrapper for implementing OCR function and serial number reading [12]. For image preprocessing steps, the cv2 library was used. The cv2 as a Python wrapper for the popular C++ library called OpenCV [13]. It is used for various image transformations, removing background noises, etc. In our case, the Otsu Thresholding algorithm combined with Gaussian Thresholding was used in order to convert images to a binary black and white format [4], which is much easier for pytesseract to process (Figure 2).

C. Test Images

For this research, more than 150 images of serial numbers were collected using mobile phones. All of the images were of bottle labels from wines coming from the company "13. jul Plantaže". Each image was cropped manually to contain only serial numbers. Some images were better quality than others and some were intentionally taken using wrong angle or light conditions in order to provide a variety for experimenting. The variations in images came from different fonts and colors used in labels coming from different wine types, camera angles at the time the pictures were taken, lighting and shadows, etc. Some images were much better than the others. Images that were almost illegible to humans were removed from the dataset.

III. Implementation

A. Tesseract Installation and Configuration

In this project, Tesseract OCR engine was used to get the serial numbers from the image. The first step to make it work was to install and configure Tesseract library [7]. When the installation and configuration process is completed, getting a text from an image can be done using the simple command in the terminal:

```
tesseract myimage.png out.txt
```

This command returns a text from the image named *myimage.png* and store recognized text to *out.txt* file.

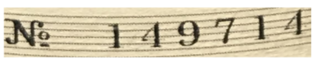
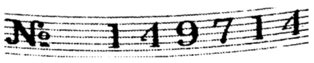
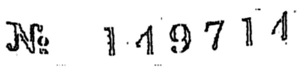
Image type	***Image***	***Azure CV***	***Google CV***	***Tesseract***
Original		-	14971	749714
After thresholding and Otsu filtering		-	-	974
After removing horizontal lines		7	149714	149714

Figure 2. Comparison of Different OCR Tools

Tesseract uses already trained data to fetch characters from the image. In addition, Tesseract allows the user to define some additional configuration, such as language (*-l*), page segment mode (*-psm*), and others. If set proprerly, these options can drastically improve the recognition accuracy. Some of these options are explained in more details below.

The language setting determines how to use already trained language or how to create and train a new one. In the installation folder of Tesseract, there is a folder called *tessdata*. All the languages that can be used must be located in this folder. The prefix of the file with an extension .traineddata is the language name that can be used by issuing the following command in terminal:

```
tesseract myimage.png out.txt -l somelan
```

where *somelan* is the name of the language. A list of the most used languages can be found on Tesseract Github [7]. If the user wants to train his or her own language, it can be done in two ways: manually and almost automatically with some human interaction. If it is done manually, the user needs a lot of time and patience because it is a very long process to make it work well. The recommended approach is the use of automatic process of training, but some human intervention is needed. The automated training is done with a tool called jTessBoxEditor, a box editor and trainer tool for Tesseract OCR [14]. At this stage of the project we did not perform additional training of the OCR engine.

B. Image Preprocessing

Before performing the number recognition, each image needs to be preprocessed. The first step is to remove background noise and convert it to a black-white or gray-scaled image. Preprocessing in order to "clean" the imaged is critical since the OCR tool was trained with such images. Preprocessing any kind of data is critical for later operations and in most cases [15]. We experimented with techniques for image preprocessing such as grayscale, erosion, Gaussian blur, and Otsu thresholding [4]. Different combinations and order of application of these techniques were experimented with before we settled on the final approach. In the experiment, we used 150 images containing manually cropped serial numbers from wine bottle labels. These are some observations: 1) An image with different shadows loses a lot of visible parts after preprocessing with some (Otsu thresholding, Adaptive Gaussian, Global thresholding) of the methods that are used. The bright part of the picture is visible and numbers from that part can be later recognized, but the dark shadow parts becomes black and nothing can be recognized from there; 2) When using Tesseract OCR engine, it is harder to recognize serial numbers from the curved images; 3) Also, it is harder to recognize serial numbers from images where horizontal lines are notable.

The preprocessing script goes through the folder containing all the images with serial numbers. The script reads and resize images one by one (function *image_resize*) to the same height size of 130 pixels, so that the resized images can later easily be processed using the same parameters. After that, the horizontal lines height is estimated by dividing the image height in pixels with a constant value. This is done with a constant value because images are resized to the same size. After experimenting with different values, it turned out that constant with value 35 worked the best. Then, this constant was used to create a kernel for the image dilation process [4]. The dilating method is used to gets the lines thinner. After dilating, Gaussian blur is used to make numbers bolder [4]. Next, the Otsu's thresholding is used to remove background and horizontal lines from the image[16]. The eroding operation is used to make numbers a little bit thicker [15]. Finally, the preprocessed image is stored for recognition by Tesseract OCR engine. Figure 3 illustrates preprocessing from the first to the last step. The first three preprocessing operations are in the left column, and the last three operations are in the right column.

C. Image Recognition, Serial Numbers Reading

After the preprocessing, the images are presumably filtered and cleared out of the horizontal lines. The pyTesseract module is used to invoke Tesseract OCR engine functions from Python script. For the evaluation purposes, the preprocessed images were renamed so that their file names match the serial numbers they represent. The evaluation script then opens image by image from a folder where these test images were stored. Different configuration settings were tested in the process of OCR evaluation.

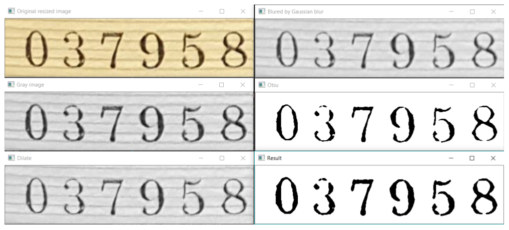


Figure 3. Numbers change through the preprocessing stage

The code snippet for OCR processing using pyTesseract is given in Figure 4. The parameter *psm* is used to set how tesseract treats the image [16], like a single line of text, a single word, block of text, etc. The value of 13 for *psm* is chosen because Tesseract should see the image as a single line of text, serial numbers are arranged in a single line. For example, if a *psm* parameter value is 8, the Tesseract module would treat the image as a single word and different recognition techniques may be used. The parameter *lang* enables developers to choose which language (traineddata) is used as a reference for Tesseract to recognize characters. As mentioned before, if needed, one can create his or her own language and train the solution using their own data. Parameter *tessdit_char_whitelist* is used to define which characters can be recognized from the image.

```
serial_number = pytesseract.image_to_string
(Image.open(image_path),
lang='eng', config='--psm 13
tessedit_char_whitelist=0123456789')
```

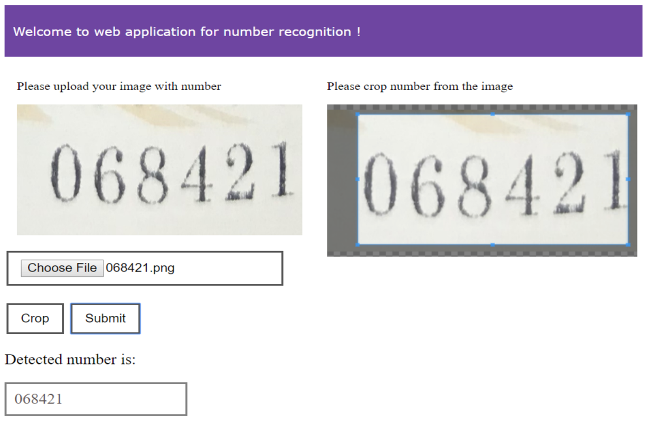
Figure 4. OCR Processing: Code Snippet for Number Recognition

IV. Results and Discussion

In the initial testing of the script, we used around 150 images of serial numbers obtained from wine labels. As mentioned, some of the images were intentionally bad quality and we did not expect the reading of serial numbers to be at 100%.

In the first run, we used the recognition script on the original test dataset without preprocessing and obtained around 62% images with full serial numbers successfully recognized (all 5-7 digits read correctly). The results would vary for different setting values of *psm*. After preprocessing images using the steps discussed in previous section, the success rate increased to 87.5%. This shows that the better the preprocessing and the better the images in the test dataset, the better the results, as was expected.

Finally, for testing and evaluation purposes, we created a custom web application (Figure 5). The user interface (frontend) is implemented using HTML, CSS, and JavaScript with CropperJS library which is used to allow users to crop numbers from a picture. The backend of this testing application is created using Python with Flask [17], a library for API creation, combined with the packages and scripts needed to implement the OCR. Using the web application helped us get the feel about what type of images can be successfully processed.


Welcome to web application for number recognition !

Please upload your image with number

Please crop number from the image

Choose File 068421.png

Detected number is:

068421

Figure 5. Web-based Test Platform

The next step will be to create a simple mobile app for reading the serial numbers from wine labels, which will also be used for the purpose of collecting a solid dataset for creating a trained dataset, which will allow us to create and train the Tesseract for a new language and to improve the success rate of the readings. We expect that the quality of images can also be improved if the mobile app provides better directions on how to position the camera and lighting while capturing the image. In addition to images, we plan on collecting additional information such as phone vendor/model, and possibly feedback from the user about the serial number.

V. Conclusion

This research aims the use of computer vision for food tracking and tracing applications. More specifically, this paper describes an effort to implement a solution for OCR reading of serial numbers from wine bottle labels in order to uniquely identify each individual bottle. The problem of detecting numbers from images is not solved for all cases. We evaluated several existing OCR tools and they all showed some weaknesses when presented with images that may contain shadows, roundness, horizontal lines, etc.

We opted to use Tesseract OCR engine, and an implementation of one possible solution is presented in this paper. The evaluation of the solution is done with a small set of 150 real-life images of wine serial numbers. The results showed that the image preprocessing plays a major role in improving the success rate.

The results are promising and the further research will utilize expanded datasets that will allow better tuning and additional training of the OCR engine.

References

[1] D. L. Rogers, "The digital transformation playbook: Rethink your business for the digital age," Columbia University Press, 2016.
[2] H. Sundmaeker at al., "Internet of Food and Farm 2020", Chapter in "Digitising theIndustry," River Publishers, ISBN 9788793379817, pp 129 - 150., 2016
[3] S. Sandi at al., "Smart tags for brand protection and anti-counterfeiting in wine industry," 23rd Information Technology Conference IT 2018, Žabljak, Montenegro, pp 1-5, 2018.
[4] Information Resources Management Association, "Computer Vision: Concepts, Methodologies, Tools, and Applications", 2018.
[5] A. Chaudhuri, K. Mandaviya, P. Badelia, S. K. Ghosh, "Optical Character Recognition Systems for Different Languages with Soft Computing", 2017, vol. 352, page 9
[6] GS1 barcodes. Accessed January 2020. [Online]. Available: https://www.gs1.org/standards/barcodes
[7] Tesseract Github repository with documentation. Accessed January 2020. [Online]. Available: https://github.com/tesseract-ocr/tesseract
[8] R. Smith, "An Overview of the Tesseract OCR Engine", in Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), 2007, vol. 2, pp. 629–633.
[9] Microsoft Azure Computer Vision. Accessed January 2020. [Online]. Available: https://azure.microsoft.com/en-in/services/
[10] Google Vision AI. Accessed January 2020. [Online]. Available: https://cloud.google.com/vision/
[11] Python programming language. Accessed January 2020. [Online]. Available: https://www.python.org/
[12] pyTesseract GitHub repository. Accessed January 2020. [Online]. Available: https://github.com/madmaze/pytesseract
[13] OpenCV project. Accessed January 2020. [Online]. Available:https://opencv.org/
[14] K. El Gajoui, F. Ataa Allah, M. Oumsis, "Training Tesseract Tool for Amazigh OCR", in 15th International Conference on Applied Computer Science (ACS '15), 2015.
[15] R. R. Palekar, Sushant U. Parab and Dhrumil P. Parikh, Prof. Vijaya N. Kamble, "Real Time License Plate Detection Using OpenCV and Tesseract", in International Conference on Communication and Signal Processing, April 6-8, 2017, India
[16] A. Khasgiwala, "Word recognition in nutrition labels with convolutional neural network", Utah State University, 2018
[17] M. Grinberg, "Flask Web Development: Developing Web Applications with Python", 2014, page 173