UNIVERSITY POLITEHNICA OF BUCHAREST
FACULTY OF AUTOMATIC CONTROL AND COMPUTERS
COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

# DIPLOMA PROJECT

Automatic tire-markings detection & recognition
Thesis advisor:

## Robert-Mihai Lică

**BUCHAREST**

Prof. dr. ing. Iuliu Vasilescu

Prof. dr. ing. Daniel Rosner

2022

# CONTENTS

## ABSTRACT

This work shall provide an image processing pipeline for automatic detection and recognition of tire-markings from images. Because tires can be one of the most expensive perishable components of a truck, they are susceptible to illegal swapping with lower quality ones without the owner noticing. My solution shall provide an easy way to extract from an image the serial code and/or certification number in order to aid an individual to realize a swap happened. At the moment the system can identify a wheel in an image and "unwrap" it in order to obtain a rectangular image.

## SINOPSIS

Această lucreare o să ofere o bandă de procesare a imaginilor pentru detectarea automată și recunoașterea marcajelor de pe marginea unui cauciuc. Deoarece cauciucurile pot fi unul dintre cele mai scumpe consumabile ale unui tir, există riscul să fie schimbate ilegal cu unele de o calitate mai scăzută fără ca proprietarul să observe. Soluția mea va oferi o metodă ușoară de a extrage din imagini codul de serie și/sau numărul certificării pentru a ajuta un individ să realizeze că un schimb a avut loc. La acest moment sistemul poate identifica o roată într-o imagine și să o "desfășoare" pentru a obține o imagine dreptunghiulară.

# 1 INTRODUCTION

## 1.1 Motivation

In an ideal world, tires would not be that expensive, especially in the truck market, and their theft or illegal swapping wouldn't be that profitable. Because we do not pay much attention when it comes to our tires, aside from: in the winter to swap them with the winter ones and in summer put the summer ones, if somebody was to change our tires for other similar looking ones (just the same color is enough) the majority of the people would not even notice. You would have to keep track of what tires you are using, probably mark down their serial number and certification number (if present) and from time to time check that the ones on your car or trucks have still the same information.

The problem of tire theft and illegal swapping is more predominant in the truck market as there tires are more expensive and wear out faster. For a truck fleet owner this is a big problem because swapped lower quality tires can be a road hazard and in case of failure they can have catastrophic results. The solution at the moment is a labor intensive, slow and prone to human error or ill will: a person ,that can be bribed, has to manually write down the information after buying a new tire and at some intervals of time it has to manually check again that the tire on the truck is still the same one, with the same specifications, and that nobody illegally swapped it for an older or lower quality one. In the case of tire theft, there isn't much to do as there is no way to determine where the tires end up utilized. If we were to apply the same solution to tire theft – as we apply to tire illegal swapping – we would need people that would inspect at the tires passing on a road, mark down the information and enter it in a database. If the respective tire was declared stolen, there would be at least a starting point in the theft investigation.

But this human inspection approach doesn't scale when it comes to hundreds of millions of vehicles. So, this is a great task for automation. It could be replicated indefinitely and the limiting factor would only be the hardware required. An automatic system for collecting information from the side of tires and reporting to a database that the serial number was seen in a particular location at a certain moment would help in identifying stolen tires that are put again in use on the roads. Or can check that the tires on a truck have the serial numbers that they should have and were not swapped.

## 1.2 Problem Statement

The center of such a system – that automatically collects and reports the information from the side of tires – is the automatic recognition of tire-markings. This process would extract from the side photo of a tire the tire-markings that are consisting usually of: a serial code (Figure 0a), a certification number (Figure 0b), a manufacturer name(Figure 0c), maximum load (Figure 0d), construction materials etc.

These markings are found as embossed letters on the side of tires and are put by manufacturers to represent the characteristics of the tire, show its certification and to distinguish between different production batches. A common serial code found on a tire is the DOT code (Figure 0a), the acronym meaning "The Department of Transportation". It is a marking that is mandated by the United States of America to be present on all the tires that are commercialized in the country and is as close as possible to a serial number. Because having different production lines is expensive, manufacturers print this DOT code also on tires that are sold in other regions of the world. This code is usually composed of: DOT marking (Figure 0a – i), tire manufacturer or manufacturing plant code (Figure 0a – ii), the size code (Figure 0a – iii), tire manufacturer (Figure 0a – iv) and finally the week and the year the tire was produced in (Figure 0a – v). Unfortunately, some of these groups of letters are sometimes missing. Anyway, this code is not enough to uniquely identify a tire as the most unique part is the date which has weekly increments only.

Another code present on the tire is the E-mark used in Europe (Figure 0b) to mark the certification which the respective tire follows and complies with. It consists of a circle with letter 'e' or 'E' followed by a number (representing the country who issued the approval that the tire meets the certification) inside a circle and next to the circle is a code or 2 lines of code representing the certification itself that the tire complies with.

A detection system that wishes to identify a tire or extract this crucial information from it should be able to recognize the DOT code and the E-mark at least. Other supplementary markings like the "ISO metric tire codes" (Figure 0e) only contain information to the physical characteristics of the tire itself and would be nice to obtain. At the moment, such detection systems require extra equipment to create special environment conditions [2] or are computationally intensive and require processing on an external server [1].

TODO: Figure 0 a – DOT code with i, ii, . . . , v under each segment of the code — 0 b – E-mark — 0 c – manufacturer's name — 0d – maximum load — 0e – ISO metric tire codes

## 1.3   Solution

My solution is to create a more robust image processing pipeline, that would not require supplementary incident lights for better letter contrast with the background, that could be run in situ, without the need of an external server for processing, in the detriment of some precision (TODO: give exact numbers preferably) in text recognition. In my image processing pipeline I will be feeding photos of car tires captured with a Cannon EOS 1300D with a resolution of 5184 by 3456 pixels. The images were taken from approximately (TODO: masoara de la ce distanta fac poze in medie) cm and at leat the hole tire was always in the shot (Figure 1). With this resolution and distance from the captured object, the characters composing the serial number have around 60 by 20 pixels in size, so not very much. I divided my pipeline in 3 big steps that tackle the problem sequentially: tire unwrapping, text detection, and text recognition.

### Tire Unwrapping

Consists of the process of determining where in the image the tire is situated, detecting its outer edge, inner edge and the center of the tire, and converting the circular shape of the tire in a rectangular one (unwrapping). In Figure 1 we can see the image of a tire that in Figure 2a has its circles detected and in Figure 2b is the unwrapped result of this step. In the unwrapped version it can be seen that the tire is not perfectly straight because of the perspective of the captured image. If the camera was not perpendicular on the tire's plane and in line with the wheel's axle, the tire has a slight oval shape that is accentuated by the unwrapping as an oval has 2 centers and not only one.

TODO: Figure 1

TODO: Figure 2a and 2b

### Text Detection

The scope of this step, to determine the regions where text might be present. Because the next step, text recognition, can be one of the most computationally intensive parts and the image's pixel count is still high after the unwrapping (TODO: get the average size of the unwrapped images), we want to reduce the space where we try to recognize characters.

I opted this step to not use machine learning and pre-trained models and instead to have a deterministic approach by using a combination of processing techniques and greedy components detection. My approach was to pass the image in the frequency domain in order to remove the high frequencies from the image – as those tend to represent noise – and the low frequencies – who usually represent the uniform background color of the tire, leaving the markings behind. This was difficult to accomplish as the tire-markings – being embossed letters – are not very prominent compared to the background of the tire (Figure 4a) without controlling lighting conditions. I needed through multiple tests to come up with a series of

3

heuristics to propose the areas that contain text. At this step I accepted to allow a higher number of false positives in the regions than a higher number of false negatives (who would have meant to miss on some text regions). It will be the task of the next step to deal with the falsely voted regions of text. In the end, I output a binary image (Figure 3) with the same size as the unwrapped one. The white pixels represent supposed text area that the next step should attempt to recognize.

TODO: Figure 3

*Text Recognition*
Text Recognition TODO

## 1.4   The Document's Structure

In this section I presented the problem as a hole and then focused on the problem of automatically recognizing tire-markings. I showed what a solution to this problem would require and my approach to solving it.

The next section – 2. State of the Art – (TODO: introduce reference) will present other work in the field, what their approaches were, their setups and results and how I am bringing new contribution to the field.

In section 3. The Proposed Solution (TODO: introduce reference) I will go in depth in the image processing pipeline. I will present each step, their sub-steps and each action performed to. I will provide information on each action in a consistent pattern: the output, how it's done – the algorithm in pseudo-code and its explanation – and optionally other approaches that didn't work and motivated me to chose this action in the end.

In section 4. Implementation Details (TODO: introduce reference) I will present and explain the code, the libraries used and how the data is managed. When is the case, I will motivate the choice of an implementation over another one.

TODO: talk also about evaluation

# 2  STATE OF THE ART

In designing my solution I've taken into account other work performed in the field of automatically recognizing tire-markings. The difficulty of the task is twofold. On one hand, the markings on tires' sides are rarely printed because they would fade in time or under the effects of the elements. Instead, the approach employed by manufacturers is to have embossed letters on the side of the tire that would fade slower than their painted counter parts. They are not foolproof either as they can also fade in time as the letters dull and become indistinguishable from the normal side of the tire, but are usually enough to outlive the tire grooves that dull the first because of the tire's usage. One characteristic of these embossed letters is that they are practically rubber on rubber, so black on black (Figure 4a). Their visibility is not great, even with the human eye can be hard to distinguish the letters in some lighting conditions. For the human eye, it can be beneficial to have an incident source of light that would make the letters cast a shadow on the side of the tire like can be seen in Figure 4b and a person could accomplish this with a flashlight that he positions at an optimal angle.

TODO: Figure 4 (a and b)

This was also the approach of Wajahat Kazmi et alia in their work [2]. They had a 2 camera setup that each would capture half of the wheel and a supplementary source of "Strobe light incident at steep angles with respect to the plane of the sidewall" to quote them. This arrangement would help with the resolution of the images and character detection in the later stages of their pipeline. As a first step, they also performed an unwrapping of the tire using Circular Hough Transform [4]. After, they focused on detecting only the DOT code by using their crafted features in order to keep a low memory footprint (the average sizes of their images were 500x2800 pixels) and extracted a Histogram of Oriented Gradients. This output they would feed in a Convolutional Neural Network based Multi-Layered Perceptron and would have as output regions of the image where text is present. The training was done with a synthetic data-set. Then, they would localize in the proposed areas the DOT code by using a deep neural network trained on a synthetic data-set of DOT foregrounds and different tire backgrounds. The same model was used also for character recognition and was trained on a 700,000 synthetic data-set of characters on black background to mimic the low contrast appearance of the embossed markings. They claim their pipeline obtained an accuracy of 80% in images that are considered acceptable to human standards, but make a note that an objective benchmark is not available. The lower quality images that would pose difficulties even to a human to recognize the text obtained an accuracy raging from 73

While the past paper's goal was for an industrial system that would have controlled conditions when performing the tire-markings recognition, there is also work in the field,

by Anton Katanaev et alia [1], for a consumer solution that would try to extract the tire specifications in order to help with ordering new ones. Their goal is to obtain the "ISO metric tire codes" that specify the type of tire, the width, aspect ratio, construction, diameter, load index and speed rating. Their first step was to collect a data-set of tire images by using internet scrappers and then filter through them using a classification model based off ResNet64. Compared to the previous work, Anton Katanaev et alia found the Circular Hough Transform unsuitable because of parameter tuning and opted for a segmentation approach for detecting the tire from the background. After this, they focused on image preprocessing to combat the different illumination in the images collected in the data-set and also perform circle correction for when the tires were appearing with an oval shape because of the camera angle. To reduce the space in which to perform the character recognition, the team also employed a step in which regions of text are searched in the image. They compared 14 pre-trained text detection models and chose the best performing one to provide supposed regions of text. After this, the proposed regions are fed into the best performing text recognition model they've tried: SEG OCR. This model obtained a character error rate of 0.16 on the original images, performing worst on the ones where the contrast was adjusted. As a final step to combat errors that might appear in the character recognition part and to identify a tire by its properties, they used a database of more than 15,000 tire characteristics for correcting incomplete or erroneous data. Because of the many learning models used, the deployment of their application is in cloud in order to leverage the computational power of AWS Cloud. On the user's end is a smartphone application to submit the captured tire image to the server to start processing and provide suitable tire candidates.

In our motivation we stated the desire of a system that would not require complex setups, a controlled light environment and that would be able to be self contained (without the need of a server for processing). This is why in my approach I preferred mathematical algorithms over learning models to accomplish the task and provide a robust system that is can perform in natural lighting conditions. TODO: sa introduc chestii despre acuratete dupa ce o calculez

# 3 THE PROPOSED SOLUTION

The solution is an image processing pipeline that would extract from the side picture of a tire the specifications written on it. These specifications are present in the form of embossed letters on the tire's exterior walls.

The system's input data are photos taken with a Cannon EOS 1300D that has a resolution of 5184 by 3456 pixels and a lens (TODO: specificatiile lentilei). While taking the pictures, the distance from the lens to the tire's side was approximately (TODO: distanta aproximativa pana la cauciuc cand fac poze). While capturing the images, I was careful to catch the entire wheel in the image because detecting half wheels or arcs would have proven difficult. One more adjustment I did was the camera placement in regards to the wheel's axle. I decided to be approximately in line with it in order for the wheel to appear circular in the image. If I wouldn't have done so, the tire would have had an oval shape. By controlling the distance between the camera and the wheel, as well as the camera position in regards to the wheel's axle, I consider I will be able to detect the tire in the image more reliably.

By desiring to not have a complex setup and not require supplementary light sources, the images were taken using the ambient light. This proved to increase the problem's difficulty quite considerably when it came to detecting the regions of text, because the contrast between the embossed letters and the background image got even smaller. Multiple approaches were considered in TODO section 3.2 Text Detection and in the end one proved fruitful in delivering acceptable results.

The system's steps are described in the following sections:

## 3.1 Tire Unwrapping

As a first step, I wanted to extract only the tire from the input image. This would help because the information I am interested in is located on it. I would also like to unwrap the tire from its disk shape (a circle with a smaller circle as a hole inside of it) into a rectangular one in order to have the writing from left to right rather than in all directions like if it was in a circle.

To better explain how this was accomplished, this step was split in multiple sub-steps:

### 3.1.1    Circle Detection

I detect where in the image are the circles representing the outer and inner borders of the tire and get their center's coordinates. The two detected circles might not have a common center because of the imperfections in taking the images, but I account and try to correct for that through a series of heuristics.

Because this is the beginning of the pipeline, I introduced a step to equalization the images and have some independence from the lighting conditions. Then I detected the first circles in the image using TODO: Hough Circle Transform and filtered its output through a series of TODO: heuristics.

## a)   Equalization

*Accomplishes:*

Lighting conditions are not controlled and this might result in low contrast in some areas of the image. Here we ensure a constant contrast across the entire image and that the entire histogram is used. This increases the robustness of the overall system.

*Reasoning:*

The pixel in a black and white image is a representation of how bright that particular part of the image should be (0 is for no light and 255 is for maximum brightness on a 8 bit sensor). If we take a photo in broad daylight, the values of all the pixels could all be over 200 lets say. The image would appear as being washed-out and lacking details (Figure 5a). If the photo is taken in a dark environment and all pixel values are under 100, the image appears again quite dark and lacking detail (Figure 5b). A solution is to use the space that remained unused in pixel values and stretch the existing pixel values to also cover those, thus increasing the contrast of the image and its level of detail (Figure 5c). This action is called equalization and can be used to enhance the contrast in an image. Furthermore, it spreads the pixel values across their entire value range so that a under-light and an over-light picture's histogram would look similar. The histogram is a plot that has on the $X$ axis all possible pixel values and on the $Y$ axis the number of times that value appears in an image.

TODO: Figure 5 (a, b, c)

A deviation from this standard equalization is Adaptive Histogram Equalization (AHE) that is not applied on the entire image and just on a small portion of it. When calculating the new value for a pixel, only a neighborhood around it is taken into consideration. Thus method while useful in bringing more contrast in to an image that has lighter and darker regions, it also increases the noise in the image. To counteract this, Contrast Limited AHE (CLAHE) was created. The addition is that it has a clipping upper limit and redistributes the pixels that

appear too often in the image into another ranges.

## Algorithm:

CLAHE, described in pseudo-code in Algorithm 1 in accordance to the implementation found in [5] and the explanation in the paper [3].

On $line1$ we split the image in multiple cells of size $k$ by $k$ because CLAHE is an adaptive algorithm. Then, for each of the cells we calculate their histogram to find out how many different pixel values they contain ($N_{gray}$). To perform the clipping, we need to calculate a few things:

$N_{avg} \leftarrow (k \times k) \div N_{gray}$, that is the number of pixels in the cell divided by how many pixel values are present in the cell;

$N_{CL} \leftarrow cl \times N_{avg}$, the clipping limit calculated based on input parameter $cl$;

$N_{\Sigma\_clip} \leftarrow \sum_{i=0}^{N_{gray}} max(0, H_s[i] - N_{CL})$, is the number of pixels that need to be clipped;

$N_{avg\_gray} \leftarrow N_{\Sigma\_clip} \div N_{gray}$, is how many pixels in average we will modify in addition to have a certain value.

Then, we iterate through all the pixel values in the histogram and create a new one. If the pixel count is $\geq$ than $N_{CL}$, we will clip that pixel value. Else, we will add to the pixel count $N_{avg\_gray}$ and if we exceed the clipping limit, we floor to it and take note of this in the form of $N_r emain$. After this step, the histogram would look like in figure 1.
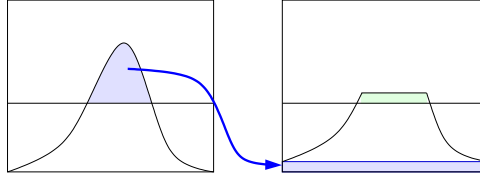


Figure 1: the histogram after first clipping, source: TODO wikipedia

The limit we excede with will be redistributed after this on $line5$. Here, we go through the pixel values in the histogram again and at a simmilary random step we add one more pixel to the count that overflowed earlyer. The step is computed by the formula: $step \leftarrow N_{gray} \div N_{remain}$.

Before starting to compute the next cell, we adjust the current one acoording to the new histogram we just computed: $H_{res}$.

After all the cels were modified and equalized, we can stisch them back together. But because we had a cell by cell approach to calculating the histogram, border pixels might have very different values. This is why an extra step of bilinear interpolation is performed on the border pizels of each of the cells.

---
**Algorithm 1** CLAHE
---
**Require:** $img$ be a $X \times X$ matrix; $cl\epsilon[0,1]$; $k$

**Ensure:** $k \mid X$

  1: $S \leftarrow (img$ split into $k \times k$ cells$)$

  2: **for** $s$ in $S$ **do**

  3:     $H_s \leftarrow histogram(s)$

  4:     $H_{res}, N_{remain} \leftarrow clip(H_s, cl)$

  5:     $H_{res} \leftarrow redistribute(H_{res}, N_{remain})$

  6:     $s \leftarrow adjust\_cell(s, H_{res})$

  7: **end for**

  8: $img \leftarrow$ interpolate cells $S$
---

*Results:*

    After equalization is applied, if the input image's histogram looked like TODO: Figure, now it looks like TODO: Figure.

# b)   Hough Circle Transform

*Accomplishes:*

*Reasoning:*

*Algorithm:*

*Results:*

# c)   Heuristics

*Accomplishes:*

*Reasoning:*

*Algorithm:*

*Results:*

### 3.1.2 Convert to Polar Coordinates

### 3.2 Text Detection

### 3.2.1 Segmentation

### 3.2.2 Segment Binarization

### 3.2.3 Filtering for Text

### 3.2.4 Voting

### 3.3 Text Recognition

*a) TEMPLATE*

*Accomplishes:*

*Reasoning:*

*Algorithm:*

*Results:*

# 4 IMPLEMENTATION DETAILS

# BIBLIOGRAPHY

[1] Katanaev Anton, Ukhmylov Yaroslav, Chekmareva Alfiya, and Khalili Roman. How to identify vehicle tires using deep learning visual models. `https://blog.griddynamics.com/how-to-identify-vehicle-tires-using-deep-learning-visual-models/`. Last accessed: June 2022.

[2] Wajahat Kazmi, Ian Nabney, George Vogiatzis, Peter Rose, and Alex Codd. An efficient industrial system for vehicle tyre (tire) detection and text recognition using deep learning. *IEEE Transactions on Intelligent Transportation Systems*, 22(2):1264–1275, 2021.

[3] J. MA, X. Fan, S.X. Yang, X. Zhang, and X. Zhu. Contrast limited adaptive histogram equalization based fusion for underwater image enhancement. *Preprints 2017*, 2017030086, 2017.

[4] OpenCV-Docs. Hough circle transform. `https://docs.opencv.org/3.4/d4/d70/tutorial_hough_circle.html`. Last accessed: June 2022.

[5] Karel Zuiderveld. C ansi implementation of contrast limited adaptive histogram equalization. `http://www.realtimerendering.com/resources/GraphicsGems/gemsiv/clahe.c`. Last accessed: June 2022.