SRH University Heidelberg

Master's Thesis

An Approach to Digit Recognition Based on Histogram of Oriented Gradients

Author:	Inga Samaniego, Cesar Augusto
Matriculation Number:	11010516
Address:	Am Dorf 26 69124 Heidelberg Germany
Email Address:	cesar.ing23@gmail.com
Supervisor:	Prof. Dr. Milan Gnjatović
Begin:	01. January 2019
End:	26. August 2019

Statutory declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

………………………………………………..	………………………………………………..
Date	(Signature)

Abstract

This master thesis introduces an approach to digit recognition based on the Histogram of Oriented Gradients method and cosine similarity. The work describes a prototype system that implements the introduced approach and illustrates it for the case of car license plate digit recognition. The recognition accuracy of 93% is reported.

Kurzfassung

Diese Masterarbeit stellt eine Herangehensweise zur Erkennung von Ziffern vor, die auf einer Methode namens Histogram of Oriented Gradients und der Kosinus-Ähnlichkeit basiert. Die Arbeit beschreibt ein Modellsystem, das die eingeführte Herangehensweise anwendet und in diesem Fall die Erkennung der Ziffern von Autoplaketten veranschaulicht. Eine Erkennungsgenauigkeit von 93% wird vermeldet.

Contents

Chapter 1 Introduction 8

1.1. Statement of the problem 9
1.2. Scope and outline of the thesis 10
1.3. Goals of the thesis 12

Chapter 2 Image pre-processing 13

2.1. Gray scale conversion 13
2.2. Thresholding 19
2.3. Sobel filter 26

Chapter 3 Histogram of Oriented Gradients 34

3.1. Gradient feature technique 34
3.2. Gradient feature extraction 36
3.3. Normalization 39
3.4. Cosine similarity 39

Chapter 4 Datasets, comparison and results 41

4.1. Equipment and tools used for the collection and processing of the data 41
4.2. Image datasets 42
4.3. Description of the comparison 46
4.4. Results and confusion matrix 48

Chapter 5 Issues presented and discussion of the results 52

5.1. Data collection issues 52
5.2. Managing processing issues 53
5.3. Discussion of the test results 53

Chapter 6 Conclusions and future work 56

6.1. Conclusions 56
6.2. Further development and future work 57

List of Figures 58

List of Tables 60

Bibliography 61

Chapter 1

Introduction

License plate recognition (LPR) is a computer vision method used to identify vehicles via their license plates and this requirement has largely increased as the automotive industry is increasing continuously as well. Thus, an automatic LPR is a powerful way to deal with recognition for large numbers of vehicles.
Furthermore, a license plate consists of several parts e.g. country codes, city codes, images printed, diverse characters among digits. This way, the range of the domain of car license plates has been increased and the use of an automatic LPR is more required than before.

There are several methods and techniques for a LPR, and the present work introduces a specific technic for recognition of a particular domain of car license plates.
In this thesis, the method proposed for the digit recognition is based on the Histogram of Oriented Gradients (HOG) and it is focused on recognition of car license plate digits from fixed regions already containing individual digits. Such car license plates specifically belong to prototypes of German license plates.

In Chapter 1, a general overview of the present work is described while in Chapter 2, the pre-processing stage is studied. The pre-processing analyzes some procedures to improve the quality of the images before further processing and it provides facilities to adjust sizes, color conversion, binarization and filtering.
In Chapter 3 the HOG method is described and gradient features of images are obtained. This chapter also includes the normalization technique and the cosine similarity concepts that are used for comparison analysis.

Then, the concepts mentioned in Chapter 2 and Chapter 3 are applied and assessed in a source code written in *Processing* to compute the similarity between defined templates and hundreds of digit samples. The sequence of these stages are as shown in the block diagram from Figure 1.1.

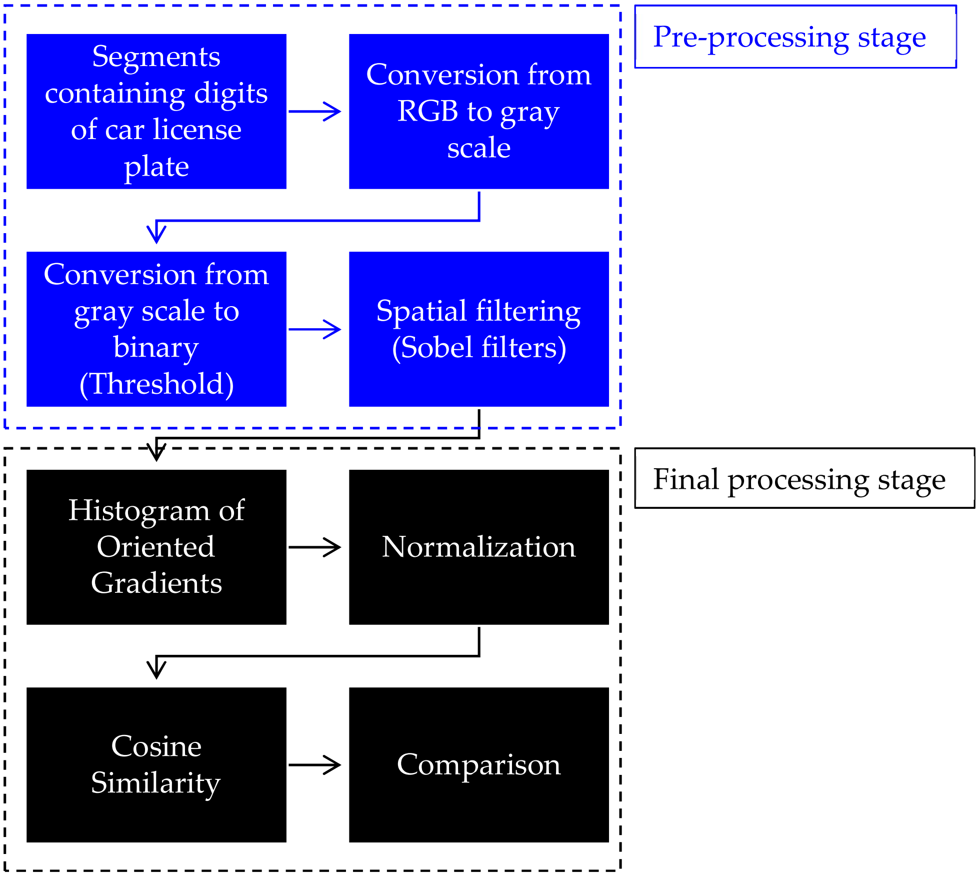

Pre-processing stage
Segments containing digits of car license plate
Conversion from RGB to gray scale
Conversion from gray scale to binary (Threshold)
Spatial filtering (Sobel filters)
Final processing stage
Histogram of Oriented Gradients
Normalization
Cosine Similarity
Comparison

Figure 1.1: Block diagram of the main stages of the process of digit recognition

In Chapter 4, the image datasets that were taken to test the system, the comparison to achieve the recognition and the results are presented. These full results are finally discussed in Chapter 5.

1.1. Statement of the problem

The scope of the problem of this thesis is based on the digit recognition and the accuracy obtained after applying the system proposed. In order to test the system, images containing digits of German car license plates were taken. These images present different factors such as dirt, sun light, shadow, camera flash, low resolution, low contrast, curved or sloped forms, which represent a challenge at the moment of processing them.

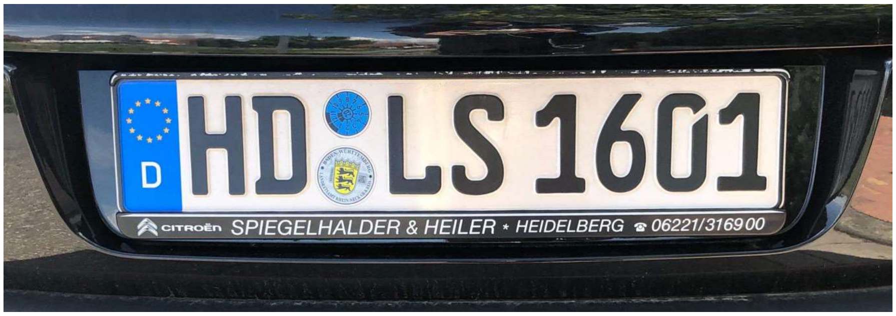


Figure 1.2: Picture of a German license plate.

1.2. Scope and outline of the thesis

The scope of this thesis is to introduce a prototype system that implements an approach to digit recognition for the case of German car license plates from segments already containing individual digits and report the recognition accuracy by making a test with 1,000 samples.
The main function of this system is to recognize which digit appears and to distinguish it from the others by computing a similarity after processing them. The images considered for the test stage shall be as the form illustrated in Figure 1.4. Those images are represented by the digits from *0* to *9*.

1. Input pattern: Standard pattern of German license plates as shown in Figure 1.3 from which the segments containing digits are obtained as depicted in Figure 1.4.

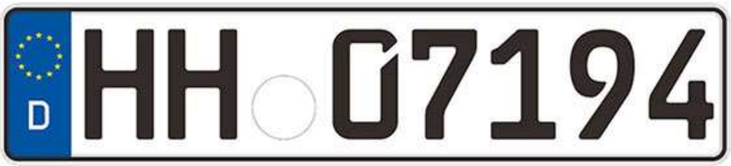


Figure 1.3: Standard German license plate

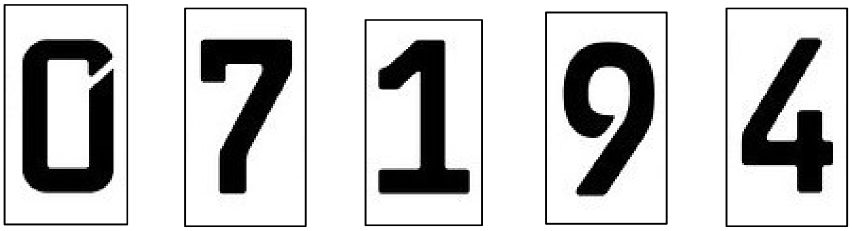


Figure 1.4: Cropped images per individual digit

2. Template and samples: As shown in Figure 1.5, templates from digit 0 to 9 were taken and compared with 100 samples per digit. Thus, there are 1,000 samples in total with 10 additional templates. Some samples are shown in Figure 1.6.

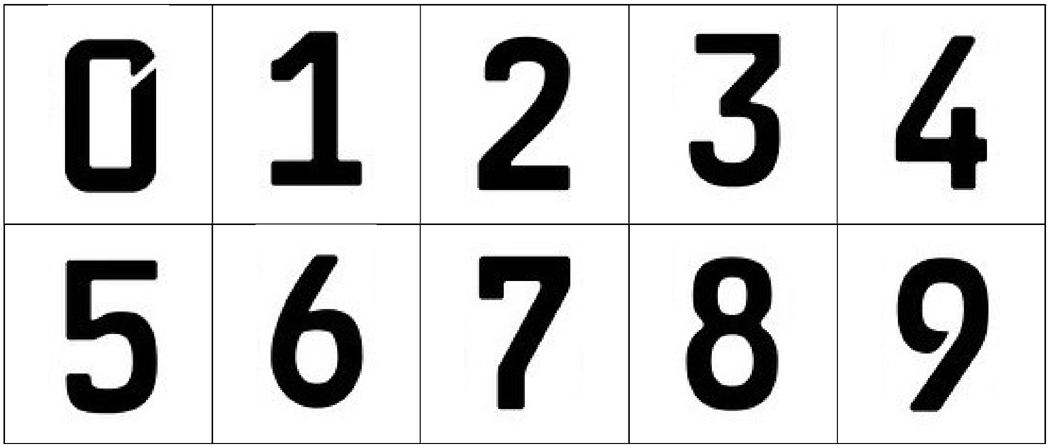


Figure 1.5: Templates of the digits, from zero to nine, considered for the comparison analysis.

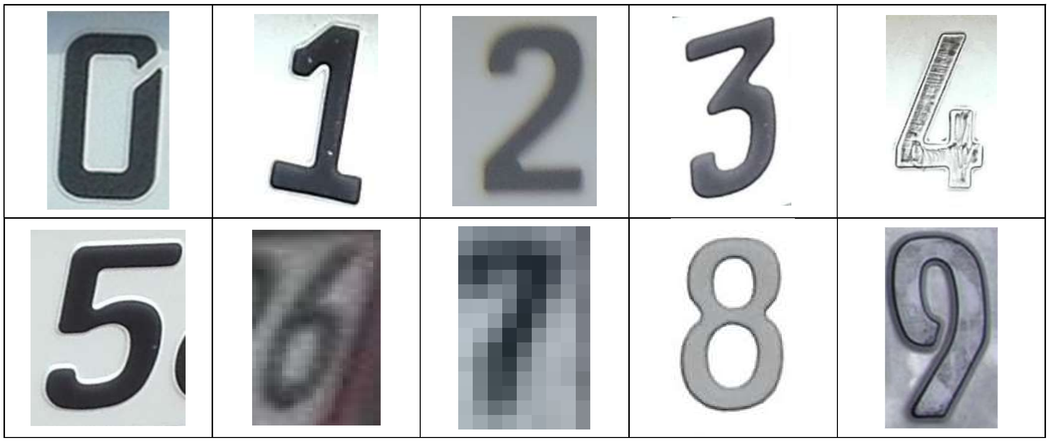


Figure 1.6: Some samples of the digits, from zero to nine, considered for the comparison analysis.

3. Results: The results are obtained from computing the algorithms and diverse considerations presented in this thesis. This computation is generated by a suitable software for image processing called *Processing* and the version chosen is 3.5.3. Further information regarding this software can be found in this reference: [P3].

1.3. Goals of the thesis

After having introduced the prototype system for recognition of digits, the goal is to report the recognition accuracy of 93% by analyzing the result of comparing all the ranges of the domain of samples with all the templates described in the preceding section.

Chapter 2

Image pre-processing

The pre-processing stage aims to improve the quality of an image basically by reducing or removing the noise and stabilizing the intensity of the images previous to its processing to get their features for further analysis. This pre-processing basically lies on the following steps: RGB to gray scale conversion, conversion from gray scale to binary and filtering.

2.1. Gray scale conversion

A gray scale image is basically defined as a gray monochrome digital image in which the contrast range is from black to white with one intensity value per pixel [SS00].

As mentioned in Section 1.2 of Chapter 1 all the images taken as samples, are color images. In order to start the pre-processing stage, the color images taken shall be converted into gray level images to reduce the computational cost and simplify the algorithms that are used.

The color-to-gray scale conversion is made by mathematical algorithms and there are several methods to achieve it, but techniques based weighted averages of the red, green and blue channels *(R, G* and *B)* are most frequently used. For this thesis, the Luminance algorithm is applied and it is described in the following equation [MK15]:

$$0.21 * R + 0.71 * G + 0.07 * B. \tag{2.1}$$

Before introducing such a conversion in detail, below some basic definitions that the pre-processing stage is based on.

The term *digital image* can be defined as a two-dimensional image with a discrete array of intensity samples conveniently represented at each spatial point (x, y) [SS00]. It can also be explained as a matrix of size $M\ x\ N$, $a_{|MxN|}$, as described in Equation (2.2) [GM18].

$$\boldsymbol{a} = \begin{bmatrix} \boldsymbol{a_{0,0}} & \boldsymbol{a_{0,1}} & \cdots & \boldsymbol{a_{0,N-1}} \\ \boldsymbol{a_{1,0}} & \boldsymbol{a_{1,1}} & \cdots & \boldsymbol{a_{1,N-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{a_{M-1,0}} & \boldsymbol{a_{M-1,1}} & \cdots & \boldsymbol{a_{M-1,N-1}} \end{bmatrix}. \tag{2.2}$$

Each element of this matrix, called pixel, has information about its location in the image at (x, y) and the intensity value $a_{x,y}$:

$$\begin{aligned} a_{x,y} &= f(x, y), \\ x &\in \{0, 1, \dots, N-1\}, \\ y &\in \{0, 1, \dots, M-1\}. \end{aligned} \tag{2.3}$$

Additionally, the coordinate system in which the pixels of an image given are found, starts at the top left. The positive x-axis going from left to right and the positive y-axis from the origin downward [GM18].

Under the consideration mentioned above, images to be processed shall be represented as it is shown in Figure 2.1:

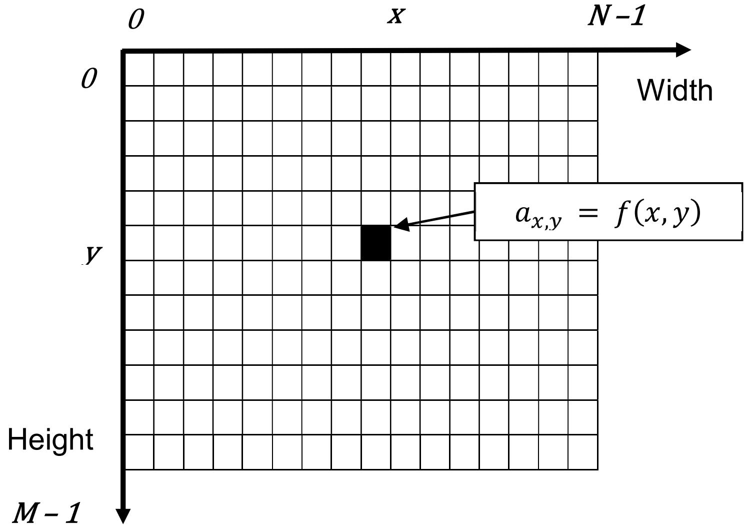


Figure 2.1: Matrix representation of an image and pixel location on the described system coordinates. (Image adopted from the next reference: [GM18])

As previously described, the input image to be processed is a color digital image, which is a three-dimensional color space (R, G and B). It means that it has 3 different matrices for each space. Following the matrix representation described in Equation (2.2), and assuming the color image of size *M x N,* the input color image can be represented as [MK15]:

$$I_{3|MxN|} = I(R,G,B), \tag{2.4}$$

where R, G and B are matrices as Equation (2.5) shows:

$$\begin{gathered}
\boldsymbol{R} = \begin{bmatrix} \boldsymbol{r_{0,0}} & \boldsymbol{r_{0,1}} & \cdots & \boldsymbol{r_{0,N-1}} \\ \boldsymbol{r_{1,0}} & \boldsymbol{r_{1,1}} & \cdots & \boldsymbol{r_{1,N-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{r_{M-1,0}} & \boldsymbol{r_{M-1,1}} & \cdots & \boldsymbol{r_{M-1,N-1}} \end{bmatrix}, \\
\boldsymbol{G} = \begin{bmatrix} \boldsymbol{g_{0,0}} & \boldsymbol{g_{0,1}} & \cdots & \boldsymbol{g_{0,N-1}} \\ \boldsymbol{g_{1,0}} & \boldsymbol{g_{1,1}} & \cdots & \boldsymbol{g_{1,N-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{g_{M-1,0}} & \boldsymbol{g_{M-1,1}} & \cdots & \boldsymbol{g_{M-1,N-1}} \end{bmatrix}, \\
B = \begin{bmatrix} b_{0,0} & b_{0,1} & \cdots & b_{0,N-1} \\ b_{1,0} & b_{1,1} & \cdots & b_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ b_{M-1,0} & b_{M-1,1} & \cdots & b_{M-1,N-1} \end{bmatrix}.
\end{gathered} \tag{2.5}$$

The formula from (2.1) now can be applied to the Equation (2.5) resulting in a new matrix of size *M x N* denoted as $\boldsymbol{a^g}_{|MxN|}$:

$$\boldsymbol{a^g = 0.21 * R + 0.71 * G + 0.07 * B}, \tag{2.6}$$

or,

$$\begin{aligned}
\boldsymbol{a^g} = \boldsymbol{0.21} * & \begin{bmatrix} \boldsymbol{r_{0,0}} & \boldsymbol{r_{0,1}} & \cdots & \boldsymbol{r_{0,N-1}} \\ \boldsymbol{r_{1,0}} & \boldsymbol{r_{1,1}} & \cdots & \boldsymbol{r_{1,N-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{r_{M-1,0}} & \boldsymbol{r_{M-1,1}} & \cdots & \boldsymbol{r_{M-1,N-1}} \end{bmatrix} \\
+\boldsymbol{0.71} * & \begin{bmatrix} \boldsymbol{g_{0,0}} & \boldsymbol{g_{0,1}} & \cdots & \boldsymbol{g_{0,N-1}} \\ \boldsymbol{g_{1,0}} & \boldsymbol{g_{1,1}} & \cdots & \boldsymbol{g_{1,N-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{g_{M-1,0}} & \boldsymbol{g_{M-1,1}} & \cdots & \boldsymbol{g_{M-1,N-1}} \end{bmatrix} \\
+\boldsymbol{0.07} * & \begin{bmatrix} \boldsymbol{b_{0,0}} & \boldsymbol{b_{0,1}} & \cdots & \boldsymbol{b_{0,N-1}} \\ \boldsymbol{b_{1,0}} & \boldsymbol{b_{1,1}} & \cdots & \boldsymbol{b_{1,N-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{b_{M-1,0}} & \boldsymbol{b_{M-1,1}} & \cdots & \boldsymbol{b_{M-1,N-1}} \end{bmatrix}.
\end{aligned} \tag{2.7}$$

And the result after convolving may be represented as:

$$\boldsymbol{a^g} = \begin{bmatrix} \boldsymbol{a^g}_{\mathbf{0,0}} & \boldsymbol{a^g}_{\mathbf{0,1}} & \cdots & \boldsymbol{a^g}_{\mathbf{0},\boldsymbol{N}-\mathbf{1}} \\ \boldsymbol{a^g}_{\mathbf{1,0}} & \boldsymbol{a^g}_{\mathbf{1,1}} & \cdots & \boldsymbol{a^g}_{\mathbf{1},\boldsymbol{N}-\mathbf{1}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{a^g}_{\boldsymbol{M}-\mathbf{1,0}} & \boldsymbol{a}_{\boldsymbol{M}-\mathbf{1,1}} & \cdots & \boldsymbol{a^g}_{\boldsymbol{M}-\mathbf{1},\boldsymbol{N}-\mathbf{1}} \end{bmatrix}, \tag{2.8}$$

However, for practical implementations the matrix given in Equation (2.2) is reformulated as a vector v [GM18]:

$$v^g = \underbrace{a^g{}_{0,0}, \dots, a^g{}_{0,N-1}}_{\text{First row}}, \underbrace{a^g{}_{1,0}, \dots, a^g{}_{1,N-1}}_{\text{Second row}}, \dots, \underbrace{a^g{}_{M-1,0}, \dots, a^g{}_{M-1,N-1}}_{\text{m}^{\text{th}}\text{ row}} \tag{2.9}$$

Where any element from the matrix $\boldsymbol{a}$ located at (x, y) is now represented with an index position at $y * N + x$ of vector v from (2.9) [GM18]:

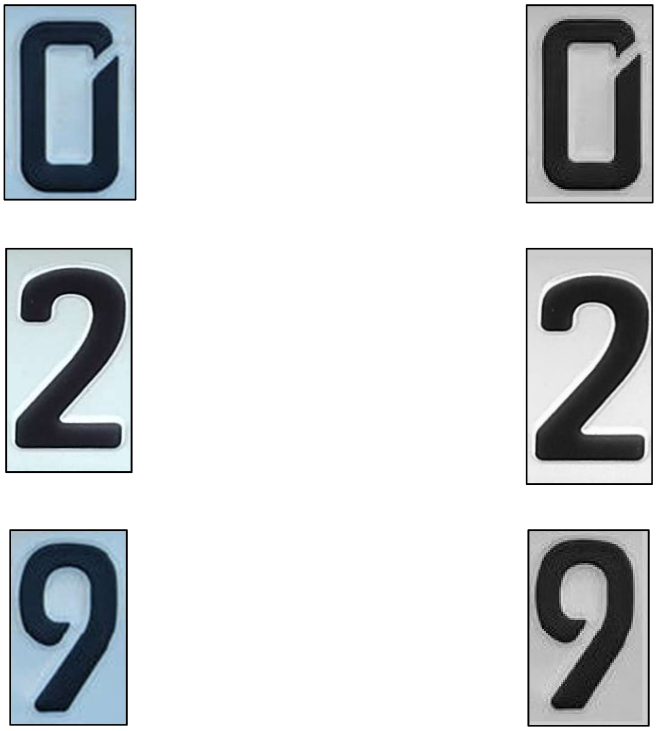
$$\boldsymbol{a^g}_{x,y} = \boldsymbol{v^g}_{y*N+x} \tag{2.10}$$

Therefore, the input color image $I_{3|MxN|}$ of size $M\ x\ N$ with three dimensional color space is now a monochromatic image $\boldsymbol{a^g}_{|MxN|}$ of size $M\ x\ N$ and conveniently represented as $\boldsymbol{v^g}_{y*N+x}$.

A gray level image is commonly stored in a 8-bit array, which means there are 2^8 possible intensity levels, so the gray scale output range is from *0* to *255* where *0* represents black and *255* for white. So, for the present work this consideration shall be kept.

Sample 1: In order to show the output from the computation made in *Processing*, it is taken the input image of digit *0, 2* and *9* shown in Figure 2.2 (a).

The Figure 2.2 shows the display window of *Processing*, where the gray scale conversion of the inputs given. Figure 2.2 (a) illustrates the input color images ($\boldsymbol{a}_{x,y}$) from the Sample 1, and (b) is the result of the computation or the output of the conversion ($\boldsymbol{a^g}_{x,y}$).



(a) (b)

Figure 2.2: (a) Original Image $a_{x,y}$ and (b) its gray scale image conversion $a^{g}{}_{x,y}$

2.1.1 Histogram of a gray scale image

As previously described the intensity levels of a gray scale image are within the range $[\mathbf{0}, \mathbf{255}]$. From this range, a gray level $\boldsymbol{r_k}$ can be selected at the pixel position $\boldsymbol{kth}$ and there might be several pixels with the same gray level, which is denoted as $\boldsymbol{n_k}$. Thus, in order to graphically represent these values, a histogram is plotted. A histogram is basically a distribution of those discrete intensity values and it can be presented as a function $\boldsymbol{h}(\boldsymbol{r_k}) = \boldsymbol{n_k}$ [GW18].

By convenience, the histogram is normalized by changing the range from [0, 255] to [0, 1]. To do that, each of its values is divided by the total number of pixels, which is denoted by n.

Calling the previous assumption, an input image with a size of $\boldsymbol{M\ x\ N}$ and the normalized histogram denoted as $\boldsymbol{p}(\boldsymbol{r_k})$ are considered. Then, $\boldsymbol{n} = \boldsymbol{M\ x\ N}$ and the normalized histogram $\boldsymbol{p}(\boldsymbol{r_k}) = \boldsymbol{n_k}/\boldsymbol{n}$ [GW18].

Therefore, the normalized histogram of a gray scale image can be presented by the Equation (2.11) [GM18].

$$p(r_k) = \frac{n_k}{\sum_{i=0}^{255} n_i} = \frac{n_k}{M.N}, \tag{2.11}$$

where $p(r_k)$ gives an estimation of the probability of occurrence of gray level r_k, so the summation of each of its elements is 1 [GM18]:

$$\sum_{i=0}^{255} p(r_k) = 1. \tag{2.12}$$

Subsequently, in order to show the results of the source code written in *Processing* for this section, the digit *0* is taken from the same *Sample 1* previously mentioned. The Figure 2.4 shows the display window of *Processing*, where the histogram of the gray level image is computed. Figure 2.4 (a) is the gray scale image and (b) is the result of plotting its histogram. Additionally, Figure 2.3 illustrates a histogram of gray levels of the image shown in Figure 2.4 (a), which was plotted in excel with the output values computed in *Processing*.

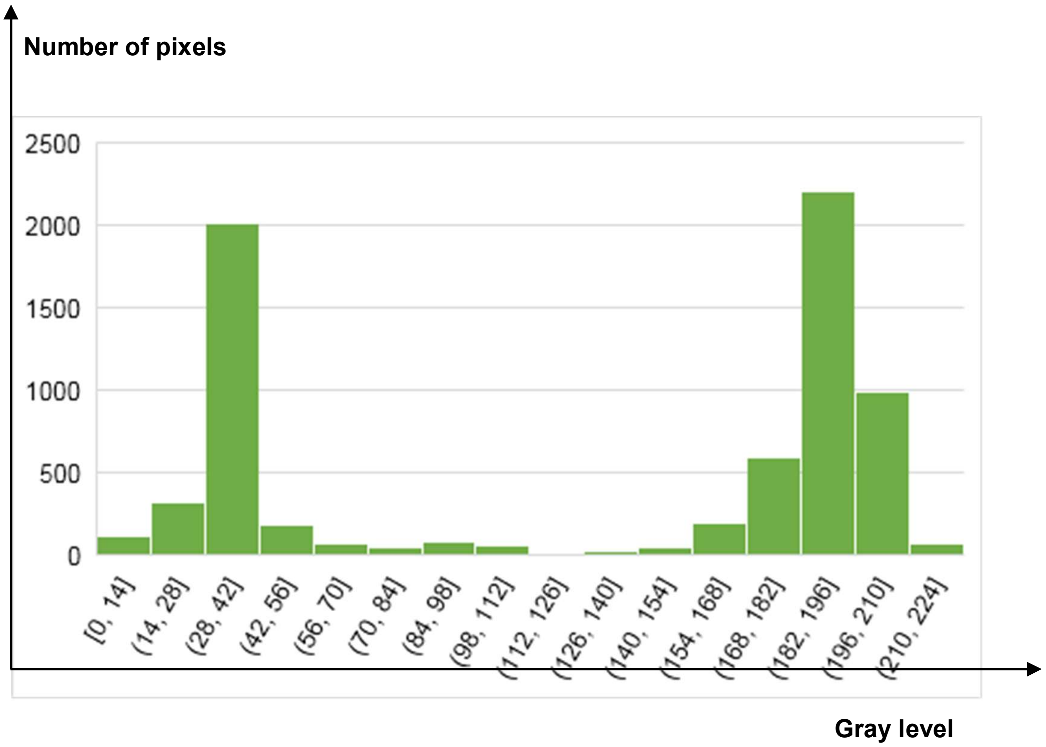
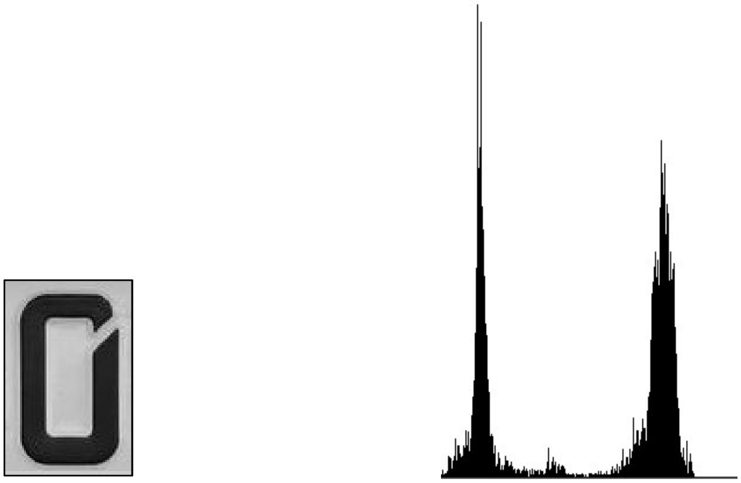


Figure 2.3: Non-normalized Histogram of gray levels of the digit *0* image shown in Figure 2.4 (a)



(a) **(b)**

Figure 2.4: (a) Gray scale image of the digit *0*, and (b) its histogram computed in *Processing*.

2.2. Thresholding

At this point, all the intensity values of the pixels are defined and plotted in a histogram, as described in the preceding section. The next step is now the binarization and to work with two groups properly segmented.

The thresholding process is basically used to create binary images by turning all pixels which are below a threshold value to zero and those greater than the threshold value to one [GW18].

For the same assumption, the gray level histogram shown in Figure 2.5 corresponds to the image shown in Figure 2.4 (a), $f(x,y) = \boldsymbol{a^{g}}_{\boldsymbol{x,y}}$, where $f(x,y)$ is the gray level of point (x,y).and it can be appreciated that there are two dominant groups of gray levels, which are divided by T.
After that, the process starts by selecting a threshold T that separates these groups. Any point (x,y) for which $f(x,y) > T$ is defined as an object point and if $f(x,y) \leq T$, the point is called a background point.
The thresholded image can be defined as $a^{t}{}_{x,y}$ and presented as follows [GW18]:

$$\boldsymbol{a^{t}}_{x,y} = \begin{cases} 1, & if\ f(x,y) > T \\ 0, & if\ f(x,y) \leq T \end{cases} \tag{2.13}$$

Thus, pixels turned into 1 correspond to objects whereas pixels turned into 0 correspond to the background.

An additional consideration is that when T depends only on $f(x, y)$ the threshold is known as global threshold and if T depends on the spatial coordinates x and y, then, the threshold is called $dynamic$ or $adaptive$ [GW18]. For the purpose of this thesis, it is considered the automatic global threshold, which is explained below.

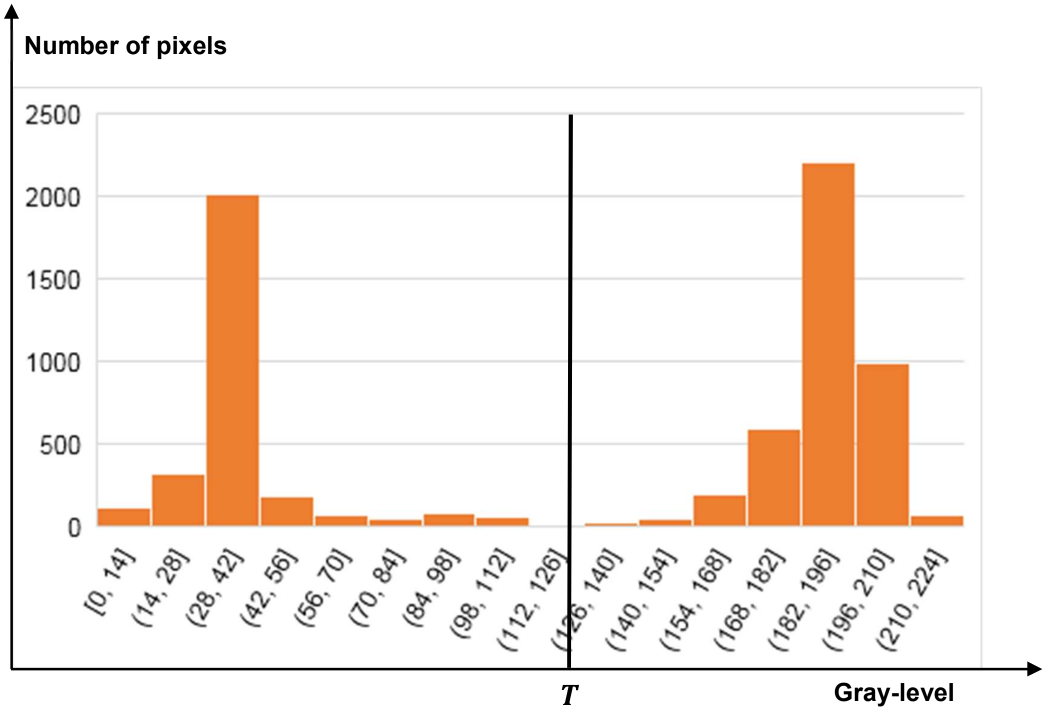
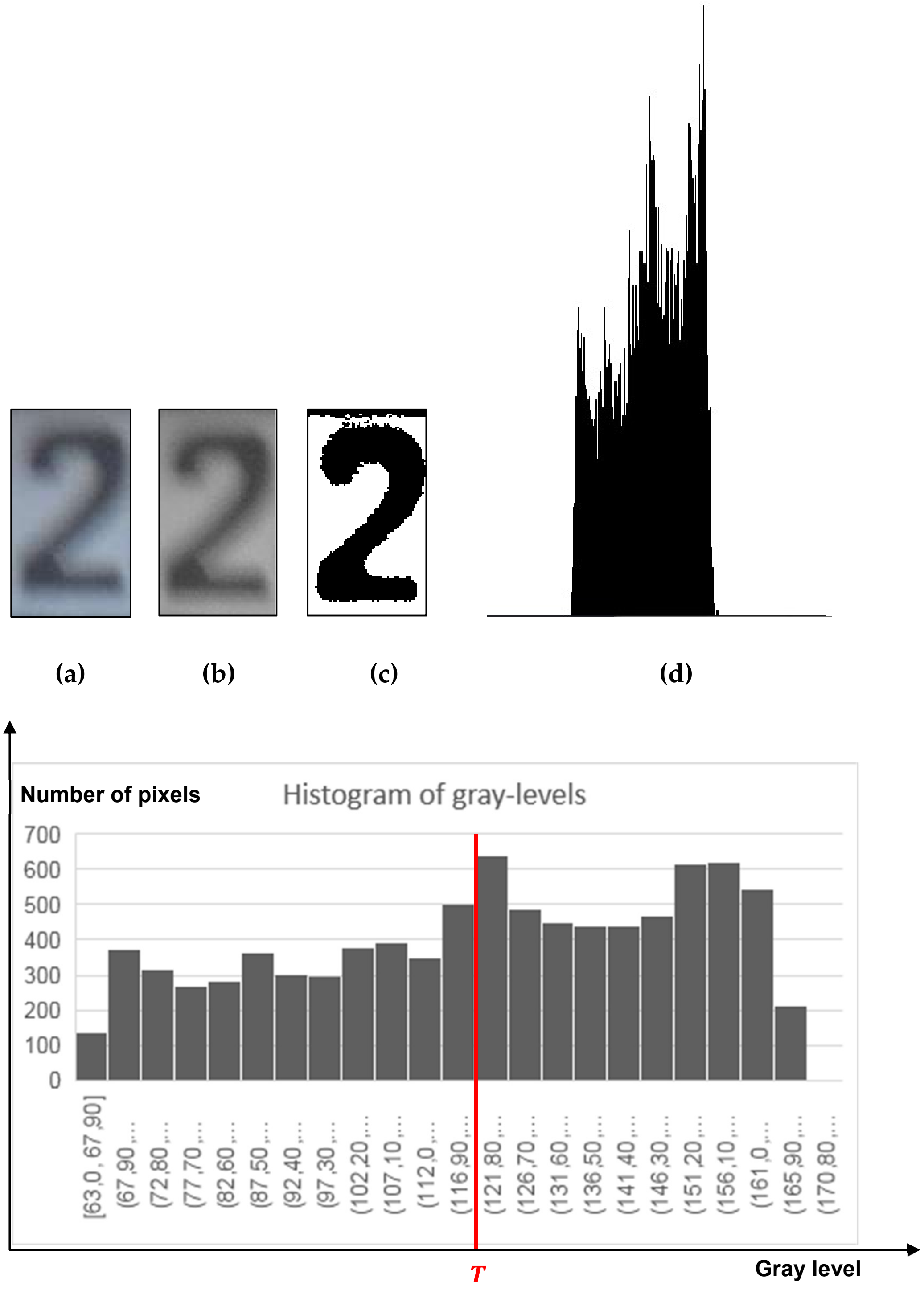


Figure 2.5: Gray level histogram that can be partitioned by a single threshold

Figure 2.6 shows an example of thresholding. A threshold value T that is a midway between the maximum and minimum gray levels of an input image is chosen as illustrated in Figure 2.6 (e) (plotted in excel), which is around 122. Figure 2.6 (a) shows the input image, which is a color image of the digit 2, Figure 2.6 (b) is its gray scale image, Figure 2.6 (c) is its corresponding thresholded image, and Figure 2.6 (d) its histogram of gray levels.

This threshold achieved in Figure 2.6 (c), is a segmentation, which was obtained by eliminating the shadows and leaving only the objects of interest. These objects in this case are darker than the background, so any pixel with a gray level less than T was turned into black (0), and any pixel with a gray level greater than T was turned into white (255).



(a) **(b)** **(c)** **(d)**

(e)

Figure 2.6: (a) Input color image of digit 2, (b) its gray scale image, (c) its threshold value with a midway, (d) its histogram computed in *Processing*, and (e) its histogram showing gray levels plotted in excel

Nevertheless, this method is basic and one unintentionally might choose a threshold value *T*, which leads to undesirable results as shown in Figure 2.7 (c) and (d).

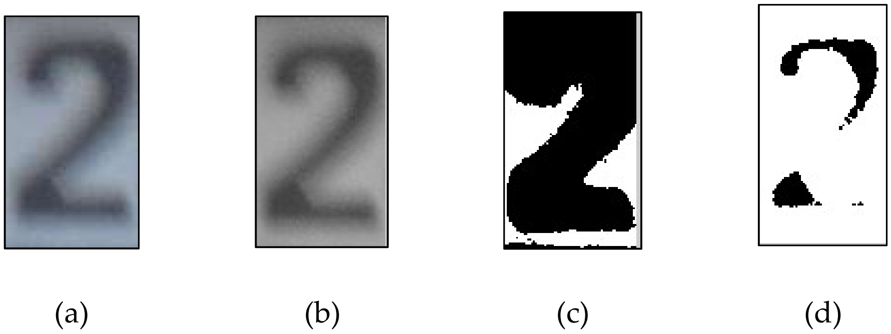


Figure 2.7: (a) Input color image, (b) gray scale of the input image, (c) result of a applying a higher threshold value *T* than desired, and (d) result of a applying a lower threshold value *T* than desired

In order to select a proper threshold value, it is needed to use a technique, which can select a suitable threshold value to properly select the object and background point so that the objects are not impacted.

2.2.1 Automatic global Thresholding

The automatic global thresholding, as its name suggests, is an automatic process to estimate a suitable threshold value for each image [GW18].
The most basic technique of all thresholding methods is to divide the image histogram by using a single global threshold T, as illustrated in Figure 2.6, however it can happen to have different distributions of intensities and using a basic thresholding might not be enough and convenient.

Histograms of gray levels for images like Figure 2.8 (a), follow a similar feature since they have their dominant gray levels near the white and black side as shown in Figure 2.8 (b). But as mentioned, images might have distinct dominant intensity levels grouped in different regions and selecting a single threshold value might lead to undesirable outputs as illustrated in Figure 2.7 (c) and (d).

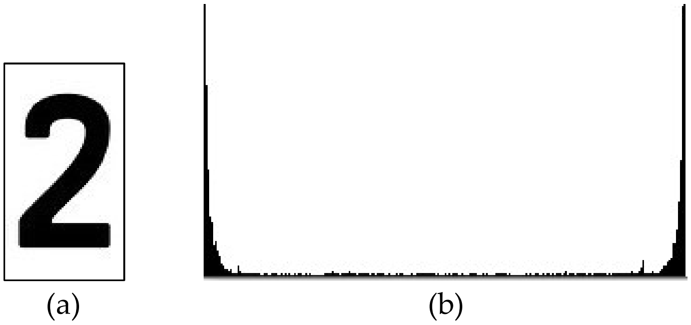

(a) (b)

Figure 2.8: (a) gray scale image of the digit 2, and (b) its histogram computed in *Processing*.

To show the problematic situation when choosing the same threshold value ***T***, the result of applying such value is illustrated in Figure 2.9 (c). The output of the digit *7* is not a desirable result.

One can play with different threshold values for each image up to get an "optimal" result. However, it can take a long work if hundreds of samples are tested, how in the situation of this thesis. To avoid such work, it is convenient to use an automatic process to select a threshold value, which is the reason for applying the automatic global thresholding.

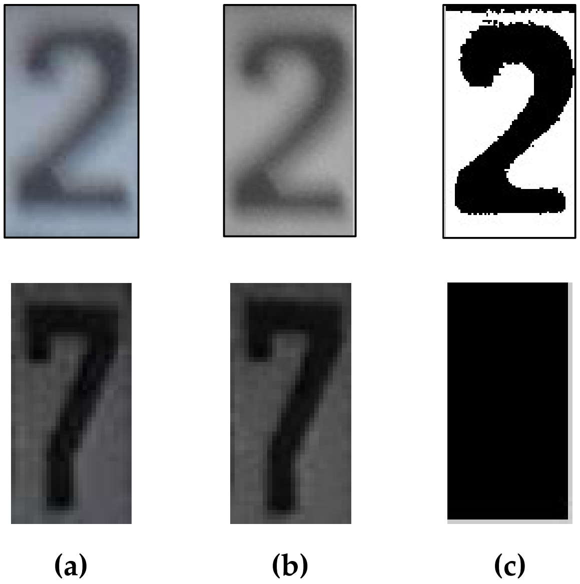

(a) (b) (c)

Figure 2.9: (a) Input images of digit 2 and 7, with display windows from *Processing* of: (b) its Gray scale Image, (c) thresholded images when applied the same midway *T* assumed for the digit 2 as illustrated in Figure 2.6.

In order to achieve this automatic process, the following algorithm is applied to obtain T automatically [GW18]:

1. Select an initial threshold value T.
2. Likewise for a basic single thresholding process, the image is divided into groups of pixels G_1 and G_2 by using T as illustrated in Figure 2.10, where G_1 consists of all pixels with gray level values greater than T and G_2 consisting of pixels with values less than or equal to T.

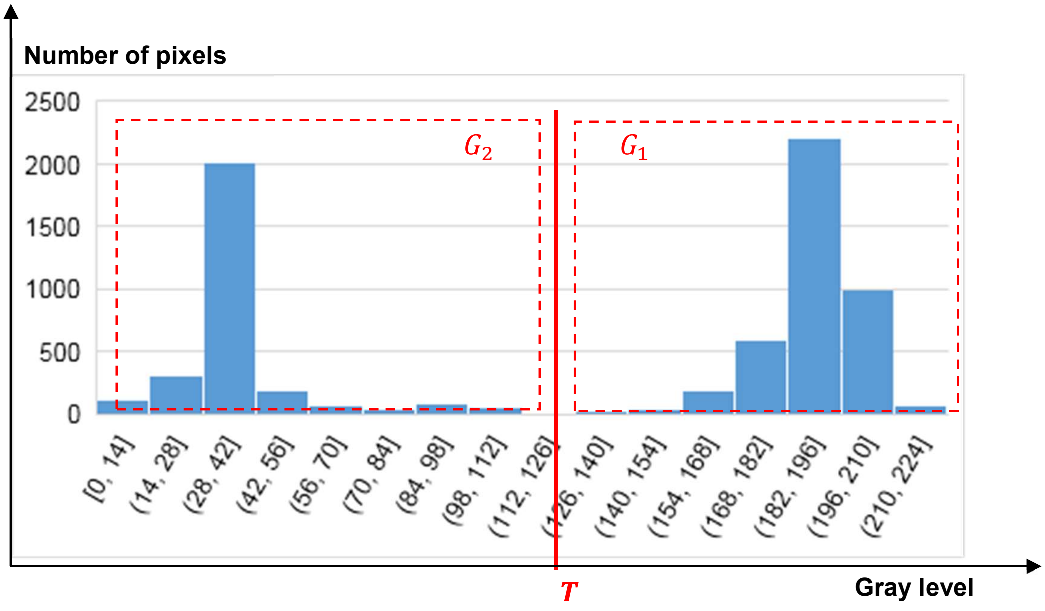


Figure 2.10: Gray level histogram that can be partitioned by a single threshold

3. Calculate the average gray level values existing in G_1 and G_2 denoted as u_1 and u_2. Considering the new representation of the image given in Equation **(2.10)**, u_1 and u_2 are defined as follows:

$$u_1 = \frac{\sum_{y=0}^{N}\sum_{x=0}^{M} r_{y*N+x}}{n_{G_1}}, if\ r_{y*N+x} > T, \tag{2.14}$$

$$u_2 = \frac{\sum_{y=0}^{N}\sum_{x=0}^{M} r_{y*N+x}}{n_{G_2}}, if\ r_{y*N+x} \leq T. \tag{2.15}$$

Where:

n_{G_1} is the number of pixels existing in G_1,
n_{G_2} is the number of pixels existing in G_2,
M is the width of the image,
N is the height of the image,

r_{y*N+x}, gray level at index $y * N + x$ of the represented image $\boldsymbol{v^g}_{y*N+x}$.

4. Calculate the new threshold value by applying the next formula:

$$T = \frac{1}{2}(u_1 + u_2). \tag{2.16}$$

5. Finally, if necessary, the steps from 2 to 4 shall be repeated until the difference in T in the next iterations is smaller than a predefined parameter T_0.

When it is observed that the background points and the objects occupy similar areas in the image, it is a recommendable to use the average gray level of the image as the initial threshold value T. However, if the object area is smaller than the background area, then there will be a group of dominant pixels in the histogram and the average just proposed is not a good initiation [GW18]. In the scope of this thesis, this situation barely happens since the input image is cropped and fixed to the object itself. Thus, the algorithm proposed for this section is used in the thresholding computation.

Another consideration is that the parameter T_0 shall stop the computation of the automatic threshold value T when the iteration is taking a long time. By setting a low value T_0 the iteration of the algorithm is much more accurate, but less efficient. On the other hand, if T_0 is a high value, the accuracy will be reduced, but the speed of the iteration is higher. In this thesis, it is considered a value, which balances both accuracy and efficiency at $T_0 = 40$ when computing the algorithm written in *Processing*.

Taking the same input images from Figure 2.9, Figure 2.11 illustrates an example of applying the automatic threshold algorithm. The Figure 2.11 (a) shows the original images of digit *2* and *7*, Figure 2.11 (b) represents the gray scale images, and Figure 2.11 (c) the result obtained when applied the iterative algorithm and the average gray level with $T_0 = 40$.

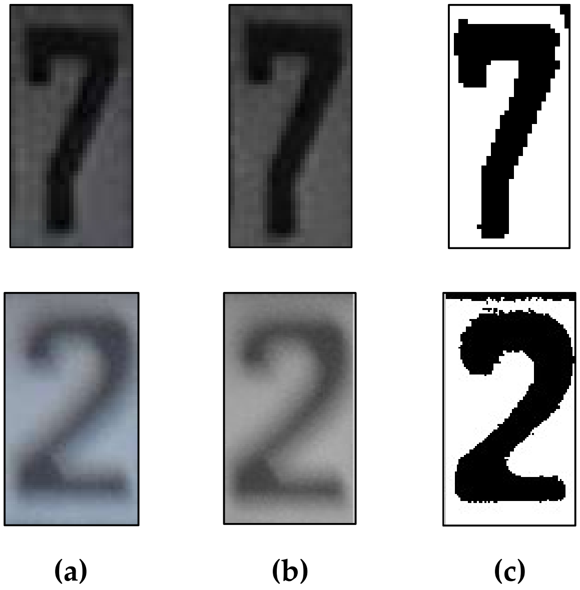


Figure 2.11: (a) Input images, with display windows of *Processing* of: (b) gray scale images, (c) thresholded images of digit 2 and 7 when applied the automatic global thresholding,

2.3. Sobel filter

Continuing with the pre-processing stage, after thresholding the image, a method to find the edges of the already processed image up to this point is applied. Edge detection is one of the most important techniques for processing an image since by detecting an edge, important information of the image can be found.

A Sobel filter or Sobel operator computes a two-dimensional spatial gradient measurement that uses derivate approximation to emphasize edges. When it finds regions with high gradients of the image, they are considered as edges. [GG13].

2.3.1 Spatial filtering

The core of the Sobel filter concept is the term *spatial gradient*, thus it is crucial to shortly define it.

The basic idea of spatial filtering is the application of a specific sub image of defined size called filter. This filter is also known as a mask, or a kernel. [GM18].

For the purpose of this section, a sub image of dimension **3 *x* 3** is considered as illustrated in Figure 2.12.

$w(-1,-1)$	$w(-1{,}0)$	$w(-1{,}1)$
$w(0,-1)$	$w(0{,}0)$	$w(0{,}1)$
$w(1,-1)$	$w(1{,}0)$	$w(1{,}1)$

Figure 2.12: A filter of dimension *3 x 3* [GM18].

Then, the process of linear spatial filtering is the following:

- The filter $w(i,j)$ presented in Figure 2.12 is moved from pixel to pixel in an input image $f(x,y)$ as shown in
- Figure 2.13.

- The response $\boldsymbol{R}$ of the filter at any pixel (x,y) of the image $f(x,y)$, is described in the next equation [GM18]:

$$\boldsymbol{R} = \sum_{i=-1}^{1}\sum_{j=-1}^{1} w(i,j)f(x+i,y+j)\,. \tag{2.17}$$

Where:

$$w(i,j) \in \mathbb{R}, for\ \{i,j\} \subset \{-1,0,1\}.$$

It is important to keep in mind that the coefficient $w(0,0)$ is placed at the same position as the pixel (x,y) of the image $f(x,y)$. It is also considered the neighborhood of the pixel (x,y) so that matrices of the same size can be convolved, which in this case is *3 x 3*. Thus, an 8-neighbourhood of pixel (x,y) is taken into account as shown in

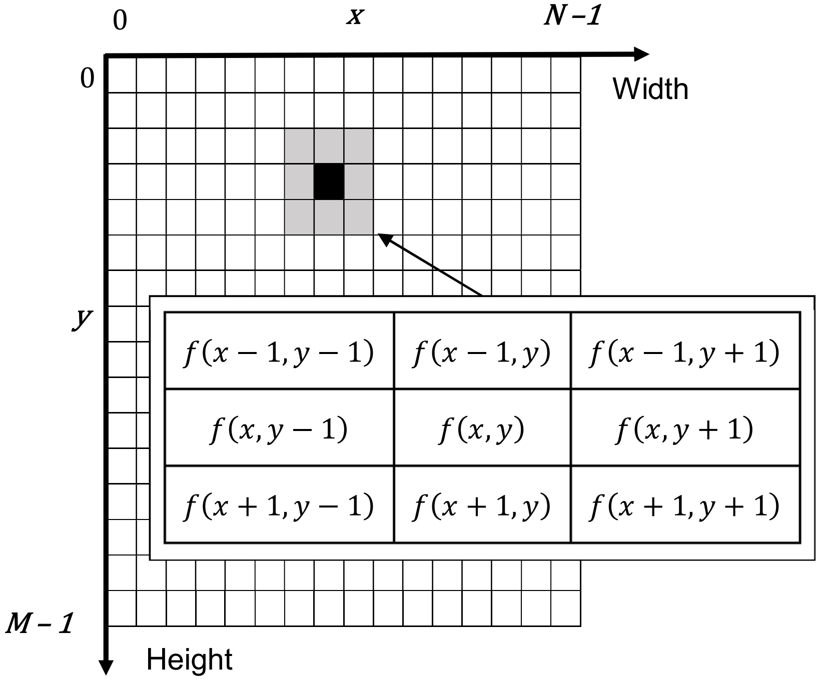
Figure 2.13 and the Equation (2.17) can be as it follows [GM18]:

$$\begin{aligned}\boldsymbol{R} = {} & w(-1,-1)f(-1,-1) + w(-1,0)f(-1,0) + \\ & w(-1,1)f(-1,1) + w(0,-1)f(0,-1) + w(0,0)f(0,0) + \\ & w(0,1)f(0,1) + w(1,-1)f(1,-1) + w(1,0)f(1,0) + \\ & w(1,1)f(1,1).\end{aligned} \tag{2.18}$$

Each pixel (x,y) will have its corresponding 8-neighbourhood which will be convolved with the filter, then it will be generated a new resultant matrix $g(x,y)$ that is set to a corresponding value $\boldsymbol{R}$ [GM18]:

$$g(x,y) = \boldsymbol{R}. \tag{2.19}$$

Figure 2.13 illustrates an 8-neighbourhood of an image $f(x, y)$ with its center point at pixel (x, y).



$f(x-1, y-1)$	$f(x-1, y)$	$f(x-1, y+1)$
$f(x, y-1)$	$f(x, y)$	$f(x, y+1)$
$f(x+1, y-1)$	$f(x+1, y)$	$f(x+1, y+1)$

Figure 2.13: Eight-neighborhood of the image $f(x, y)$ with its center point at pixel (x, y) (Image adopted from the next reference: [GM18]).

Considering the coordinate system of the image $f(x, y)$, it is important to keep in mind that if the center of the filter is applied at the point (0,0) or point $(M-1, N-1)$, the first row and first column as well as the last row and last column of the filter fall outside the image. To solve that, there are several options, but for this thesis the following is pursued:

- The pixels from the first row and first column as well as the last row and last column of the image $f(x, y)$ can be skipped in order to prevent the filter from falling outside the range of the image. Then, the starting center point of the filter $w(i, j)$ is at point $(0, 0)$, while the final one at $(M-2, N-2)$. Thus, the processed image will have less size than the original one at $(M-2)\ x\ (N-2)$.

2.3.2 Gradient

The gradient for the same image $\boldsymbol{f(x, y)}$ at coordinates $\boldsymbol{(x, y)}$ is defined as follows [GM18]:

$$\nabla f(x,y) = \begin{bmatrix} \dfrac{\partial f(x,y)}{\partial x} \\ \dfrac{\partial f(x,y)}{\partial y} \end{bmatrix} = \begin{bmatrix} g_x(x,y) \\ g_y(x,y) \end{bmatrix}, \tag{2.20}$$

and the gradient magnitude is given by:

$$\nabla f(x,y) = \text{mag}\ (\boldsymbol{\nabla f(x,y)}) = \sqrt{\left[\left(\frac{\partial f(x,y)}{\partial x}\right)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2\right]}, \tag{2.21}$$

$$\|g(x,y)\| = \sqrt{\left[g_x(x,y)^2 + \ g_y(x,y)^2\right]}. \tag{2.22}$$

The Equation (2.22) may be approximated as [GM18]:

$$\|g(x,y)\| \approx |g_x(x,y)| + \ |g_y(x,y)|, \tag{2.23}$$

Where $|g_x(x,y)|$ and $|g_y(x,y)|$ are as follows:

$$\begin{aligned} |g_x(x,y)| & \\ &= \big|\big(f(x+1,y-1) + 2f(x+1,y) + f(x+1,y+1)\big) \\ &\quad - \big(f(x-1,y-1) + 2f(x-1,y) + f(x-1,y+1)\big)\big| \end{aligned} \tag{2.24}$$

$$\begin{aligned} |g_y(x,y)| & \\ &= \big|\big(f(x-1,y+1) + 2f(x,y+1) + f(x+1,y+1)\big) \\ &\quad - \big(f(x-1,y-1) + 2f(x,y-1) + f(x+1,y-1)\big)\big| \end{aligned} \tag{2.25}$$

2.3.3 Sobel operator

After having presented spatial filtering and gradient, it is now pertinent to introduce the Sobel operator itself. This operator is basically defined as a pair of *3 x 3* convolution filters as shown in Figure 2.14, which are known as *Horizontal Sobel Operator* and *Vertical Sobel Operator* since they respond over the orientation given in their names themselves [GG13].

-1	-2	-1
0	0	0
1	2	1

(a)

1	0	1
-2	0	2
-1	0	1

(b)

Figure 2.14: Sobel filters: (a) horizontal, and (b) vertical [GM18].

Having introduced the Sobel filters in Figure 2.14, they can now be applied in the preceding equations. To do a more general formula, Equation (2.18) is slightly redefined and then the convolution operation of the input image at pixel (x, y) with the filter *3 x 3* is as follows:

$$\begin{aligned} g(x,y) \\ = f(x-1,y-1) * w(-1,-1) + f(x-1,y) * w(-1,0) + f(x-1,y+1) \\ * w(-1,1) + f(x,y-1) * w(0,-1) + f(x,y) * w(0,0) + f(x,y+1) \\ * w(0,1) + f(x+1,y-1) * w(1,-1) + f(x+1,y) * w(1,0) \\ + f(x+1,y+1) * w(1,1). \end{aligned} \tag{2.26}$$

Now, the vertical Sobel filter can be replaced in Equation (2.26):

$$\begin{aligned} g_x(x,y) \\ = f(x-1,y-1) * -1 + f(x-1,y) * -2 + f(x-1,y+1) * -1 \\ + f(x,y-1) * 0 + f(x,y) * 0 + f(x,y+1) * 0 + f(x+1,y-1) * 1 \\ + f(x+1,y) * 2 + f(x+1,y+1) * 1 \end{aligned} \tag{2.27}$$

and similarly, by applying the horizontal Sobel filter:

$$\begin{aligned} g_y(x,y) \\ = f(x-1,y-1) * -1 + f(x-1,y) * 0 + f(x-1,y+1) * 1 \\ + f(x,y-1) * -2 + f(x,y) * 0 + f(x,y+1) * 2 + f(x+1,y-1) * -1 \\ + f(x+1,y) * 0 + f(x+1,y+1) * 1. \end{aligned} \tag{2.28}$$

The equations (2.27) and (2.28) are now combined to find the magnitude of the gradient at each point (x, y) of the new function $g(x, y)$ in the equation (2.23):

$$\|g(x,y)\| \approx |g_x(x,y)| + |g_y(x,y)|,$$

and the angle of orientation of the edge is defined as follows:

$$\theta(x,y) = arctan\left(\frac{g_y(x,y)}{g_x(x,y)}\right). \tag{2.29}$$

Therefore, for the considered input image of dimension MxN, the resultant matrix of Equation (2.27) is:

$$g_x(x,y) = \begin{bmatrix} g_x(0,0) & g_x(1,0) & \cdots & g_x(N-2,0) \\ g_x(0,1) & g_x(1,1) & \cdots & g_x(N-2,1) \\ \vdots & \vdots & \ddots & \vdots \\ g_x(0,M-2) & g_x(1,M-2) & \cdots & g_x(N-2,M-2) \end{bmatrix}, \tag{2.30}$$

while the resultant matrix of Equation (2.27) is:

$$g_Y(x,y) = \begin{bmatrix} g_Y(0,0) & g_Y(1,0) & \cdots & g_Y(N-2,0) \\ g_Y(0,1) & g_Y(1,1) & \cdots & g_Y(N-2,1) \\ \vdots & \vdots & \ddots & \vdots \\ g_Y(0,M-2) & g_Y(1,M-2) & \cdots & g_Y(N-2,M-2) \end{bmatrix}, \tag{2.31}$$

the combination described in Equation (2.23) to compute the magnitude is:

$$\|g(x,y)\| = \begin{bmatrix} \sqrt{g_x(0,0)^2 + g_y(0,0)^2} & \cdots & \sqrt{g_x(N-2,0)^2 + g_y(N-2,0)^2} \\ \sqrt{g_x(0,1)^2 + g_y(0,1)^2} & \cdots & \sqrt{g_x(N-2,1)^2 + g_y(N-2,1)^2} \\ \vdots & \ddots & \vdots \\ \sqrt{g_x(0,M-2)^2 + g_y(0,M-2)^2} & \cdots & \sqrt{g_x(N-2,M-2)^2 + g_y(N-2,M-2)^2} \end{bmatrix}, \tag{2.32}$$

and its angle of orientation described in Equation (2.29) is:

$$\theta(x,y) = \begin{bmatrix} arctan\left(\frac{g_y(0,0)}{g_x(0,0)}\right) & \cdots & arctan\left(\frac{g_y(N-2,0)}{g_x(N-2,0)}\right) \\ arctan\left(\frac{g_y(0,1)}{g_x(0,1)}\right) & \cdots & arctan\left(\frac{g_y(N-2,1)}{g_x(N-2,1)}\right) \\ \vdots & \ddots & \vdots \\ arctan\left(\frac{g_y(0,M-2)}{g_x(0,M-2)}\right) & \cdots & arctan\left(\frac{g_y(N-2,M-2)}{g_x(N-2,M-2)}\right) \end{bmatrix}. \tag{2.33}$$

To do a summary of the processed images,

Figure 2.15 shows the last processes applied up to this point, where the following should be taken into account:

- ${a^t}_{x,y}$, is the thresholded image.

- Since for the present method described the first row and the first column as well as the last ones are skipped, $f(x,y)$ shall be set to ${a^t}_{x+1,y+1}$.

- After applying the filters $w(i,j)$, the dimension of the gradient vector $g(x,y)$ is $(M-2)\ x\ (N-2)$.

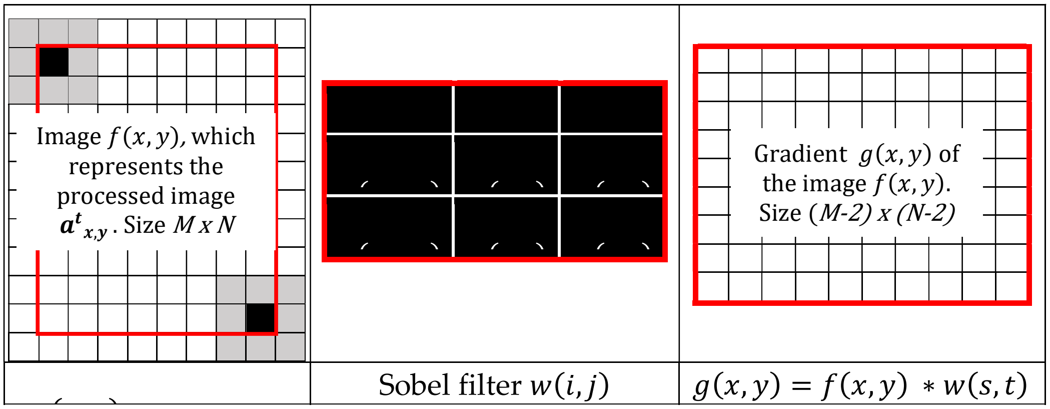
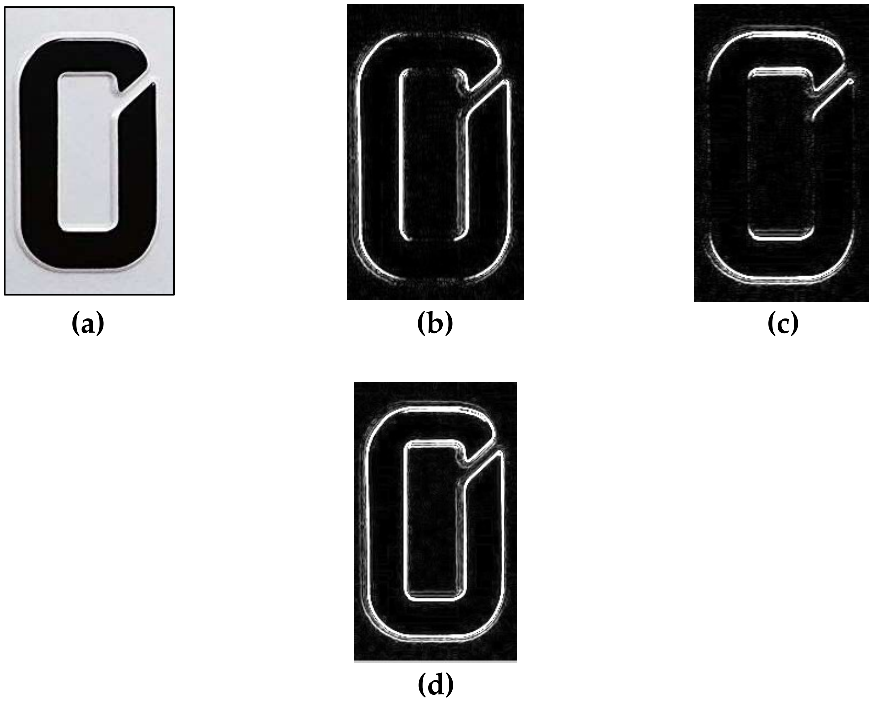


Image $f(x,y)$, which represents the processed image $\boldsymbol{a}^t_{x,y}$. Size $M\ x\ N$
Gradient $g(x,y)$ of the image $f(x,y)$. Size $(M\text{-}2)\ x\ (N\text{-}2)$
Sobel filter $w(i,j)$
$g(x,y) = f(x,y) * w(s,t)$

Figure 2.15: Graphical summary of the processed image up to this section

In Figure 2.16, the display windows of *Processing* are exhibited, showing the application of Sobel filters. Figure 2.16 (a) represents the gray scale image, Figure 2.16 (b) the output image after applying the horizontal Sobel filter, (c) the output image after applying the vertical Sobel filter, and (d) the output image after combining horizontal and vertical Sobel filters.



(a) **(b)** **(c)**

(d)

Figure 2.16: Display windows of *Processing*, showing the application of Sobel filters: (a) input image (b) horizontal, (c) vertical, and (d) horizontal and vertical combined

And applying the Sobel filter on the image from the previous examples shown in Figure 2.11, the result is illustrated in the figure below:

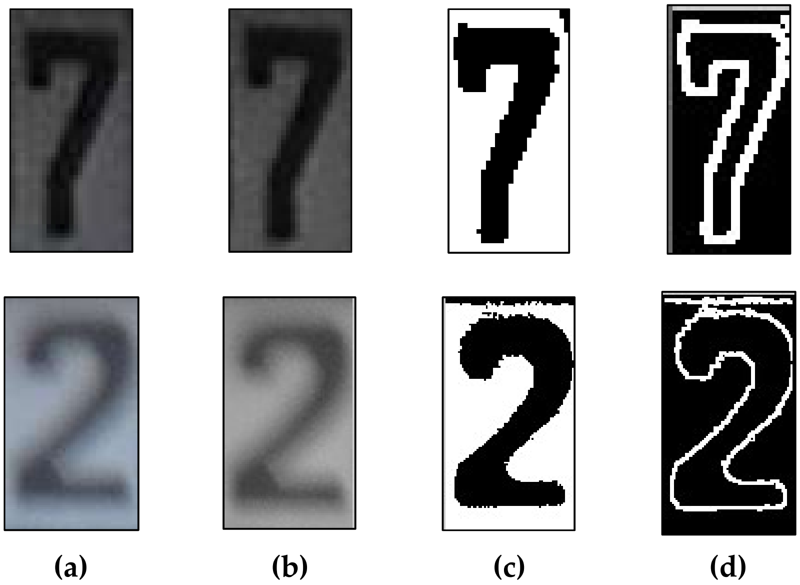


Figure 2.17: (a) Input images of the digit 2 and 7, with display windows of *Processing* of: (b) their gray scale images, (c) their thresholded images, and (d) their output when applied Sobel filter.

Chapter 3

Histogram of Oriented Gradients

In Chapter 2 it has been studied the pre-processing stage, which basically consists in the binarization of the image and their corresponding gradients that haven been investigated in the preceding chapter.

Up to this point images have been analyzed considering all their pixels, but now a technique is presented, which works with gradient features of those images that group pixels.

The Histogram of Oriented Gradients (HOG) is based on the evaluation of well-normalized local histograms of image gradient orientations in a dense grid. This grid is implemented by dividing the current pre-processed image into small spatial cells and each cell consists of accumulating local histograms of gradients directions [DT05].

3.1. Gradient feature technique

This section works with the resultant gradient vectors presented in Section 2.3 of Chapter 2. So, the main idea a gradient feature technique is to rearrange these gradients vectors in a new array called chain code directions. [GM18].

A chain code is basically a set of n elementary vectors $e_0, e_1, \dots, e_{n-1}$ rotated in increments of $\frac{2\pi}{n}$, where $n \in \mathbb{N}$.

Figure 3.1 (a) shows eight chain code directions, and (b) n chain code directions [GM18].

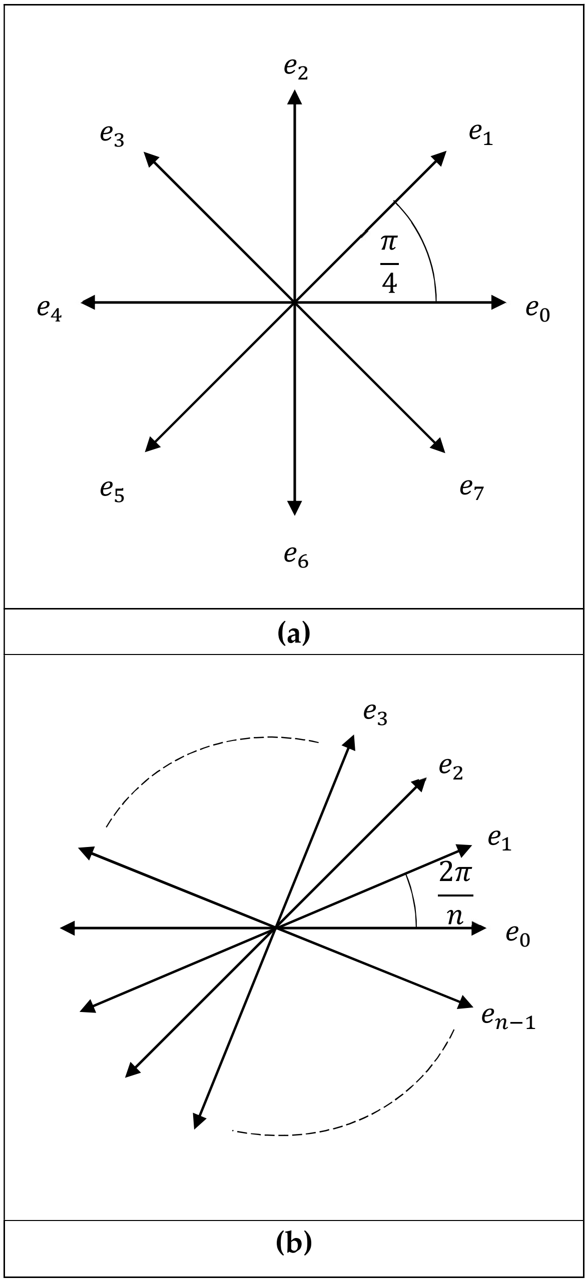


Figure 3.1: (a) eight chain code directions, and (b) n chain code directions.

For the purpose of testing the system proposed, it is considered $n = 8, 16 \; and \; 32$.

For this section, the gradient vector of pixel (x, y) that was introduced in Subsection 2.3.2 of Chapter 2 is recalled:

$$\boldsymbol{\nabla f(x,y)} = \begin{bmatrix} \dfrac{\boldsymbol{\partial f(x,y)}}{\boldsymbol{\partial x}} \\ \dfrac{\boldsymbol{\partial f(x,y)}}{\boldsymbol{\partial y}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{g_x(x,y)} \\ \boldsymbol{g_y(x,y)} \end{bmatrix}. \tag{3.1}$$

Its magnitude defined as:

$$\|\boldsymbol{g(x,y)}\| = \sqrt{[\boldsymbol{g_x(x,y)^2} + \boldsymbol{g_y(x,y)^2}]} \approx |\boldsymbol{g_x(x,y)}| + |\boldsymbol{g_y(x,y)}|, \tag{3.2}$$

and the gradient direction:

$$\boldsymbol{\theta(x,y)} = \boldsymbol{arctan}\left(\frac{\boldsymbol{g_y(x,y)}}{\boldsymbol{g_x(x,y)}}\right). \tag{3.3}$$

3.2. Gradient feature extraction

The gradient feature extraction refers to the representation of the pixel(x, y) by a gradient feature vector as follows [GM18]:

$$\left(\boldsymbol{a_0(x,y)}, \boldsymbol{a_1(x,y)}, \dots, \boldsymbol{a_{n-1}(x,y)}\right), \tag{3.4}$$

where n is the number of chain code directions, $a_i(x, y)$ is the feature vector, and $0 \leq i \leq n - 1$.
This feature vector is obtained from decomposing the pixel's gradient vector along chain code directions, where n is the number of chain code directions [GM18].

For convenience and for further explanations, the gradient feature vector can be represented as follows:

n	0	1	2	$\dots$	$n-3$	$n-2$	$n-1$
$a_i(x,y)$	$\boldsymbol{a_0(x,y)}$	$\boldsymbol{a_1(x,y)}$	$\boldsymbol{a_2(x,y)}$	$\boldsymbol{\dots}$	$\boldsymbol{a_{n-3}(x,y)}$	$\boldsymbol{a_{n-2}(x,y)}$	$\boldsymbol{a_{n-1}(x,y)}$

Figure 3.2: Representation of the array of feature vectors.

There are methods to achieve the decomposition of gradient vectors and for the present thesis it is used the method presented in Subsection 3.2.1.

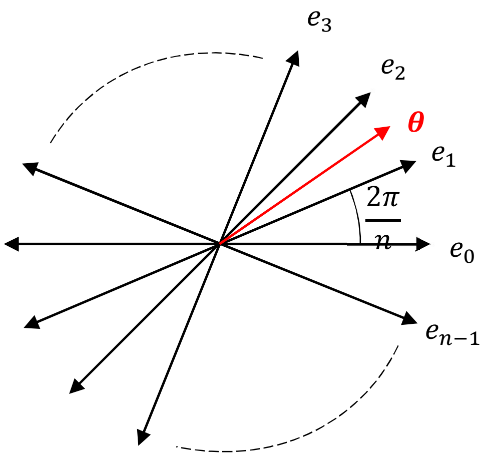
3.2.1 Decomposition of a Gradient Vector

The decomposition of the gradient vector is basically to place the module $\|g(x,y)\|$ according to its gradient direction or orientation $\boldsymbol{\theta(x,y)}$.
If the gradient direction $\boldsymbol{\theta(x,y)}$ of pixel $g(x,y)$ is aligned with the direction of vector e_j or lies between e_j and $e_{(j+1)|\,mod\,n}$, the coefficients of the feature vector describing the pixel are [GM18]:

$$\boldsymbol{a_i(x,y)} = \begin{cases} \|\boldsymbol{g(x,y)}\|, \boldsymbol{i=j} \\ \boldsymbol{0, i \neq j} \end{cases}, \tag{3.5}$$

where $0 \leq i \leq n-1$.

For example, the preceding explanation can be graphically depicted as follows:



(a)

n	0	1	2	…	$n-3$	$n-2$	$n-1$
$a_i(x,y)$	$\boldsymbol{a_0(x,y)}$ $\boldsymbol{=0}$	$\boldsymbol{a_1(x,y)}$ $\boldsymbol{= \|g(x,y)\|}$	$\boldsymbol{a_2(x,y)}$ $\boldsymbol{=0}$	…	$\boldsymbol{a_{n-3}(x,y)}$ $\boldsymbol{=0}$	$\boldsymbol{a_{n-2}(x,y)}$ $\boldsymbol{=0}$	$\boldsymbol{a_{n-1}(x,y)}$ $\boldsymbol{=0}$

(b)

Figure 3.3: (a) n chain code directions, and (b) representation of the gradient direction $\boldsymbol{\theta(x,y)}$ with its magnitude $\|\boldsymbol{g(x,y)}\|$ placed in the array of feature vectors.

Since the gradient direction $\boldsymbol{\theta(x,y)}$ lies between e_2 and e_1 as illustrated in Figure 3.3 (a), then $e_j = e_1$, thus $a_1(x,y) = \|g(x,y)\|$and the rest feature vectors are set to 0 as presented in Figure 3.3 (b).

3.2.2 Downsampling

With the feature vectors obtained in Equation (3.4), each pixel (x,y) is now defined as:

$$\big(a_0(x,y), a_1(x,y), \dots, a_{n-1}(x,y)\big),$$

which has a dimension of n.

For this case, the current dimension of the input image is $(M-2)\ x\ (N-2)$ as described in

Figure 2.15. Thus, the dimension of its corresponding accumulated feature vector is $(M-2)\ x\ (N-2)\ x\ n$ [GM18]. This feature vector can also be expressed as the following matrix:

$$a(x,y) = \begin{bmatrix} a(0,0) & a(1,0) & \cdots & a(n-1,0) \\ a(0,1) & a(1,1) & \cdots & a(n-1,1) \\ \vdots & \vdots & \ddots & \vdots \\ a(0,d) & a(1,d) & \cdots & a(n-1,d) \end{bmatrix}, \tag{3.6}$$

where, $d = (M-2) * (N-2)$.

In this process it is desired to obtain two feature vectors that describe their corresponding image rather than those images themselves for a convenient comparison. In order to achieve that, the gradient vector shall not depend on the image in terms of dimension [GM18].

To start the process of downsampling of the gradient vector, the next steps are listed [GM18]:

1. A grid is created by dividing the current image in $g_x\ x\ g_y$, forming $g_x * g_y$ cells or blocks.

2. Each cell or block obtained in the previous step, which is defined as B, is described with a feature vector of dimension n as follows:

$$(b_0, b_1, \dots, b_{n-1}), \tag{3.7}$$

where:

$$b_i = \sum_{(x,y)\in B} b_i(x,y), \tag{3.8}$$

and $0 \le i \le n-1$.

3. The image is now described with a feature of a constant dimension $g_x * g_y\ x\ n$, independently of the image dimension. Therefore, in this thesis it is not needed to resize the image.

   Then, the final feature gradient vector can be expressed in the following matrix:

$$b(x,y) = \begin{bmatrix} b(0,0) & b(1,0) & \cdots & b(n-1,0) \\ b(0,1) & b(1,1) & \cdots & b(n-1,1) \\ \vdots & \vdots & \ddots & \vdots \\ b(0,dg) & b(1,dg) & \cdots & b(n-1,dg) \end{bmatrix}, \tag{3.9}$$

   where $dg = g_x * g_y$.

3.3. Normalization

In this process called normalization, the range of pixel intensity values, which were obtained after extracting the gradient feature extraction of the image and divided in blocks as described in Subsection 3.2.2, is changed in order to have consistency in such dynamic range of data. For this purpose, the downsampled image $b(x,y)$ of dimension $g_x * g_y\ x\ n$ is normalized as follows [MA19]:

- the block B is now denoted as M with a new vector of dimension n:

$$(m_0, m_1, \dots, m_{n-1}), \tag{3.10}$$

where

$$m_i = \frac{b_i}{\sum_{j=0}^{g_x * g_y} \sum_{i=0}^{n-1} b(j,i)^2}. \tag{3.11}$$

3.4. Cosine similarity

The cosine similarity computes a measure of similarity between two vectors. Thus, the values of similarities are in the range of *0* to *1*.

For the case of this thesis, the cosine similarity computed is between two feature gradients that describe their respective input image.

Then, the final step is to compare feature gradient vectors after being normalized as described in Section 3.3.

To define the cosine similarity, the two vectors to be compared can be represented as follows [MA19]:

$$\begin{aligned}\mathrm{P} &= (p_1, p_2, \ldots, p_n\,),\\ \mathrm{Q} &= (q_1, q_2, \ldots, q_n\,),\end{aligned} \tag{3.12}$$

then, the cosine similarity equation is:

$$\sin(\mathrm{P}, \mathrm{Q}) = \cos(\theta) = \frac{P.Q}{\|P\|\|Q\|} = \frac{\sum_{i=1}^{n} p_i q_i}{\sqrt{\sum_{i=1}^{n} {p_i}^2}\sqrt{\sum_{i=1}^{n} {q_i}^2}} \tag{3.13}$$

Chapter 4

Datasets, comparison and results

In this chapter, a general description of the datasets either templates or samples used for testing the approach of digit recognition proposed is presented. Furthermore, the comparison of the results of applying the algorithm implemented in *Processing* is described.

4.1. Equipment and tools used for the collection and processing of the data

In order to get and process the datasets used through the whole process of the present work, different computational tools and equipment have been used and they have made possible to achieve the goals proposed in this thesis. So, in this section they are briefly described.

4.1.1 Equipment and tools used for the dataset collection

All the samples used for testing the system proposed were taken with digital cameras and smartphones with enough capability for the present work have been collected. To be able to achieve that dataset collection, the following equipment has been employed:

- Canon digital camera, model PC1817, camera of 12-megapixels with zoom 50x.
- Smartphone Nokia 6, model TA-1021, camera of 16-megapixels.
- Smartphone Huawei, model P8, camera 13 -megapixels.

4.1.2 Software and computer used for processing the datasets

The time of processing or execution time is indeed a very important data to know when a system is introduced. In this case, it highly depends on what kind of software is used for processing the dataset and what kind of computer is used to execute such a process.
The computer used for executing the system for recognition has the following specifications:

- Laptop Intel 7, core i7-6700HQ, 8 CPUs, 2.6GHz, 16GB RAM,

and the software used for computing the algorithms and display the results is called *Processing*:

- *Processing*: It is a flexible software used for image processing based on Java and/or other kinds of programming languages [P3].

4.2. Image datasets

As mentioned in Chapter 1, there is an image dataset used as template and another ones used as sample. From these datasets comes the segments of images containing digits for the processing stage, which were used to test the system proposed in this thesis.

4.2.1 Template dataset

The dataset of the template images was taken from a website specialized in German license plates [LP]. For convenience, these images were cropped and fixed to the dimension of around 60 x 120 pixels.

As described in Section 1.2 of Chapter 1, the template images correspond to the digits from *0* to *9* as shown in the figure below. These images are roughly black and white only and practically with no noise. Furthermore, the object occupies more space than the background. All of these features of the input images, for this thesis, are considered as the ideal scenario or as the template images for the purpose of this thesis.

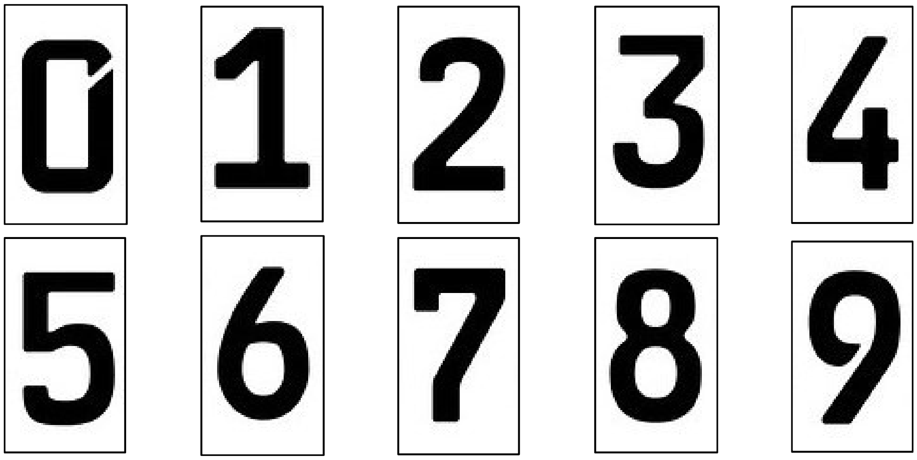


Figure 4.1: Templates of the digits, from zero to nine, considered for the comparison analysis.

Thus, the corresponding domain of template images of digits is $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}, \mathbf{5}, \mathbf{6}, \mathbf{7}, \mathbf{8}, \mathbf{9}\}$ and its respective range is $\{\mathbf{1}\}$ for each element of the domain. Therefore, there are 10 template images in total.

4.2.2 Sample dataset

As for the dataset of the sample images, it was mentioned in Chapter 1, that the samples were specifically taken from German license plates, which were photographed around the city of Heidelberg, Germany, The figure bellow shows a small extract of the collection of the pictures taken before they were cropped for the processing stage.

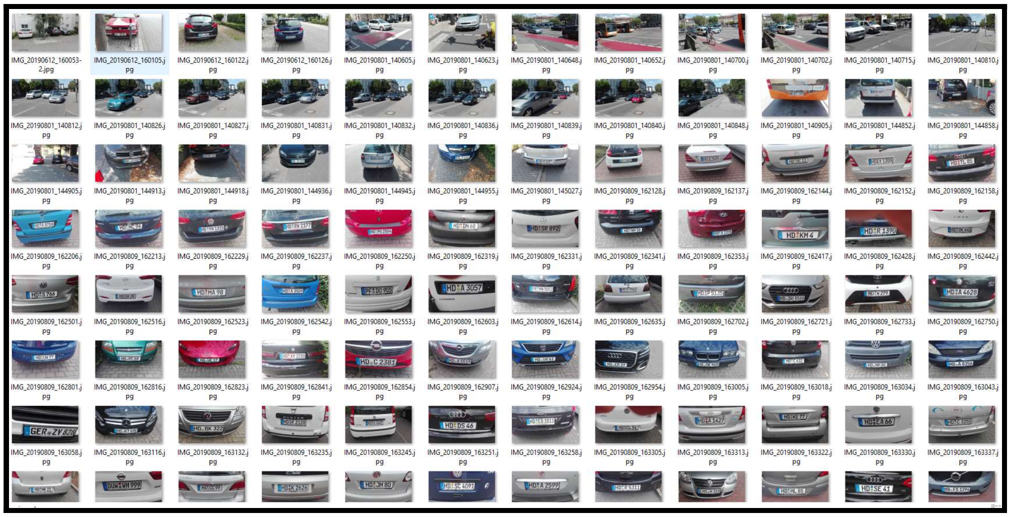


Figure 4.2: Small extract of the collection of pictures of German car license plates.

Those images were also classified regarding their special challenging features found at the moment of collecting like dirt, sunshine, shadowing, shadow, blurs, low resolution, inclination, flashlight. Some of these images are illustrated in the figure below:

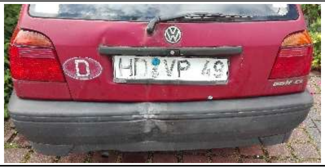
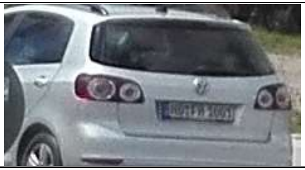
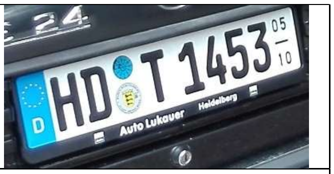
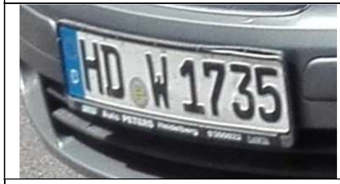
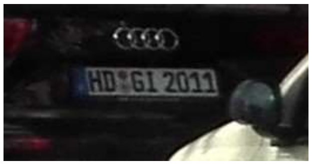
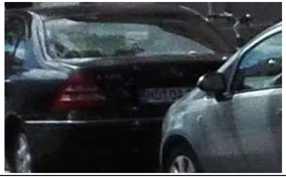
		
Sunshine	Blurry image	Sloped image
		
Curved image for bad angle shooting	Low contrast	Image with objects interfering

Figure 4.3: Pictures of license plates with specific features.

Similarly, those images were also cropped just focusing on the object of interest and despite having the features shown in Figure 4.3**,** they were kept for the analysis of comparison.

After cropping, images were saved and properly labeled so that the software could easily read them.

The number of digits considered is one hundred samples per digit, which makes a total of one thousand samples without including the template images. Thus, the corresponding domain of sample images of digits is $\{\mathbf{0},\mathbf{1},\mathbf{2},\mathbf{3},\mathbf{4},\mathbf{5},\mathbf{6},\mathbf{7},\mathbf{8},\mathbf{9}\}$ and its respective range is $\{\mathbf{100}\}$ for each element of the domain. Therefore, there are 1,000 sample images in total.

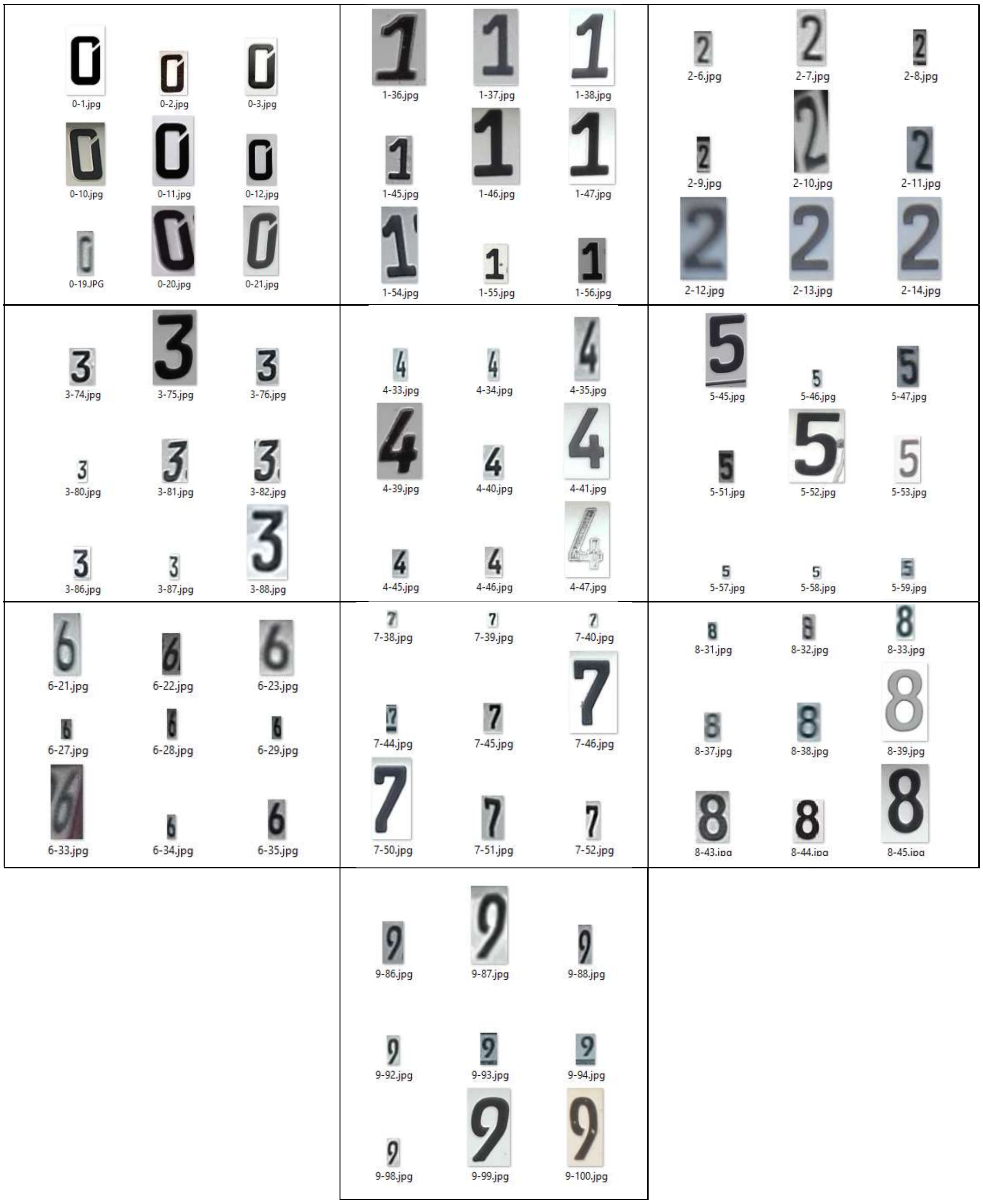


Figure 4.4: Small extract of the dataset of the cropped images from the collection of pictures shown in Figure 4.2 used as samples and grouped as per their corresponding digit.

The images presented in Figure 4.4 were conveniently labeled and saved in a file so that they can conveniently be read and processed by the software.

4.3. Description of the comparison

After having described the template images as well as the sample images in the preceding section, the next stage was to process them based on the concepts introduced in Chapter 2 and Chapter 3. In Section 3.4 of Chapter 3 the concept of cosine similarity was presented in order to compare two images, and then the process of comparison was repeated for all of the sample images with each template following the next steps:

1. First, one sample of the domain of samples, $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, is taken.
2. Then, the sample taken is compared with each template of the digits from *0* to *9*.
3. For each comparison, the cosine similarity between them is computed by applying the Equation (3.13) presented in Section 3.4 of Chapter 3.
4. Finally, the maximum value of the similarities computed is chosen and its corresponding template is classified as the digit recognized.

Figure **4.5** illustrates an example of the comparison made for one sample image testing its respective cosine similarity with each template and finally marking the highest similarity that indicates the sample has been recognized as the digit that the corresponding template image represents, which in this case is the digit *2*.

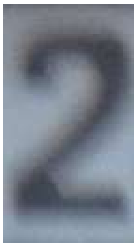
Sample image	Templates	Similarity	Observation
		0.57135856	
		0.52736086	
		0.86390525	Sample recognized as digit 2, since it is the highest similarity
		0.72321343	
		0.49615905	
		0.5947266	
		0.40831712	
		0.61892307	
		0.5966018	
		0.6344742	

Figure 4.5: An example of the comparison between a sample image with all the template images of digits from *0* to *9*.

4.4. Results and confusion matrix

A confusion matrix is basically a table that shows the recognition performance that was described in the preceding section.
To construct this matrix, the comparison described in Section 4.3 was repeated for all the samples considered for testing and then the results were sorted as illustrated from Table 4.2 to Table 4.10. The first column represents the samples per digit and the first row the template of digits, while the main diagonal the number of samples of the digit from the corresponding row recognized as the template of the digit from the respective column. Furthermore, matrices for each configuration of parameters were made and illustrated in the tables from Table 4.2 to Table 4.10. Such configurations were established by varying the dimension of the grid presented in Subsection 3.2.2 of Chapter 3 and the number of chain code directions described in Section 3.1 of Chapter 3 as follows:

Table 4.1: Set of configurations used for testing.

N° Configuration	**N° rows of the grid** g_x	**N° columns of the grid** g_y	**N° of chain code directions** n
1	2	3	8
2	3	3	8
3	2	2	8
4	2	3	16
5	3	3	16
6	2	2	16
7	2	3	32
8	3	3	32
9	2	2	32

For example, in Table 4.2 is shown that 97 samples of the number *0* were recognized as digit *0*, while 3 samples were recognized as digit *8*.

Table 4.2: Confusion Matrix for $g_x = 2, g_y = 3, n = 8$

Sample / Template	0	1	2	3	4	5	6	7	8	9
0	**97**	0	0	0	0	0	0	0	3	0
1	1	**87**	0	0	0	0	0	0	1	11
2	1	8	**72**	3	0	0	0	9	1	6
3	0	0	0	**84**	0	1	0	0	15	0
4	1	0	0	0	**80**	0	14	0	5	0
5	0	0	0	1	0	**94**	0	0	5	0
6	0	0	0	0	0	0	**100**	0	0	0
7	0	0	0	0	0	0	0	**98**	0	2
8	2	0	0	0	0	0	0	0	**98**	0
9	1	0	2	0	0	0	0	0	2	**95**

Table 4.3: Confusion Matrix for $g_x = 2, g_y = 2, n = 8$

Sample / Template	0	1	2	3	4	5	6	7	8	9
0	**95**	0	0	0	0	0	0	0	5	0
1	1	**79**	0	0	20	0	0	0	0	0
2	0	5	**81**	0	2	0	0	6	3	3
3	0	0	0	**81**	1	0	7	0	10	1
4	5	3	1	0	**72**	2	8	0	6	3
5	0	0	0	7	0	**88**	2	0	3	0
6	0	0	0	0	0	0	**98**	0	2	0
7	0	0	0	0	2	0	0	**98**	0	0
8	0	0	2	0	0	0	0	0	**98**	0
9	3	0	4	0	0	0	0	0	0	**93**

Table 4.4: Confusion Matrix for $g_x = 3, g_y = 3, n = 8$

Sample / Template	0	1	2	3	4	5	6	7	8	9
0	**79**	2	0	0	0	0	1	0	18	0
1	0	**99**	0	0	1	0	0	0	0	0
2	1	15	**77**	0	1	0	0	5	0	1
3	0	0	0	**91**	0	0	0	0	9	0
4	0	0	0	0	**85**	0	11	0	4	0
5	0	0	0	3	0	**90**	0	0	7	0
6	0	0	0	0	0	0	**96**	0	4	0
7	0	6	0	0	0	0	0	**94**	0	0
8	0	0	0	0	0	2	0	0	**98**	0
9	1	0	0	0	0	0	0	2	2	**95**

Table 4.5: Confusion Matrix for $g_x = 2, g_y = 3, n = 16$

Sample / Template	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	**99**	0	0	0	0	0	0	0	1	0
1	2	**96**	0	0	0	0	0	0	0	2
2	0	2	**83**	3	0	0	0	4	2	6
3	0	0	0	**84**	0	1	1	0	14	0
4	1	3	0	1	**76**	0	15	0	4	0
5	0	0	0	1	0	**94**	0	0	5	0
6	0	0	0	0	0	0	**100**	0	0	0
7	0	0	0	0	0	0	0	**100**	0	0
8	0	0	2	0	0	0	0	0	**98**	0
9	3	0	2	2	0	0	0	0	2	**91**

Table 4.6: Confusion Matrix for $g_x = 2, g_y = 2, n = 16$

Sample / Template	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	**95**	0	0	0	0	0	0	0	5	0
1	0	**91**	0	1	7	0	0	0	1	0
2	0	0	**82**	0	0	0	1	1	6	10
3	0	0	0	**80**	0	0	9	0	11	0
4	8	4	1	0	**67**	1	8	0	11	0
5	0	0	0	4	0	**91**	1	0	4	0
6	0	0	0	0	0	0	**100**	0	0	0
7	2	0	0	0	0	0	0	**98**	0	0
8	0	0	2	0	0	0	0	0	**98**	0
9	3	0	4	0	0	0	0	0	2	**91**

Table 4.7: Confusion Matrix for $g_x = 3, g_y = 3, n = 16$

Sample / Template	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	**86**	4	0	0	0	0	1	0	9	0
1	0	**100**	0	0	0	0	0	0	0	0
2	1	4	**84**	1	0	0	0	4	4	2
3	0	0	0	**88**	0	0	0	0	12	0
4	0	0	0	0	**84**	0	10	0	6	0
5	0	0	0	3	0	**90**	0	0	7	0
6	0	0	0	0	0	0	**98**	0	2	0
7	0	2	0	0	0	0	0	**98**	0	0
8	0	0	0	0	0	2	0	0	**98**	0
9	1	0	2	0	0	0	0	0	2	**95**

Table 4.8: Confusion Matrix for $g_x = 2, g_y = 3, n = 32$

Sample / Template	0	1	2	3	4	5	6	7	8	9
0	**100**	0	0	0	0	0	0	0	0	0
1	3	**96**	1	0	0	0	0	0	0	0
2	0	0	**89**	1	1	0	0	2	3	4
3	0	1	0	**83**	0	0	5	0	11	0
4	2	4	1	1	**72**	0	13	0	7	0
5	0	0	0	1	0	**95**	0	0	4	0
6	0	0	0	0	0	0	**100**	0	0	0
7	2	0	0	0	0	0	0	**98**	0	0
8	0	0	2	0	0	0	0	0	**98**	0
9	3	0	2	2	0	0	0	0	2	**91**

Table 4.9: Confusion Matrix for $g_x = 2, g_y = 2, n = 32$

Sample / Template	0	1	2	3	4	5	6	7	8	9
0	**97**	0	0	0	0	0	0	0	3	0
1	0	**96**	1	1	1	0	0	0	1	0
2	0	0	**80**	0	0	0	0	0	5	15
3	0	0	1	**81**	2	0	6	0	10	0
4	5	3	0	0	**65**	0	17	1	9	0
5	0	0	0	3	0	**91**	0	0	6	0
6	0	0	0	0	0	0	**100**	0	0	0
7	2	0	0	0	0	0	0	**98**	0	0
8	2	0	2	0	0	0	0	0	**96**	0
9	2	0	5	0	0	0	0	0	2	**91**

Table 4.10: Confusion Matrix for $g_x = 3, g_y = 3, n = 32$

Sample / Template	0	1	2	3	4	5	6	7	8	9
0	**89**	4	0	0	0	0	1	0	6	0
1	0	**100**	0	0	0	0	0	0	0	0
2	0	2	**91**	0	0	0	0	2	4	1
3	0	1	0	**82**	1	0	4	0	12	0
4	0	0	0	0	**84**	1	5	0	10	0
5	0	0	0	4	0	**89**	0	0	7	0
6	0	0	0	0	0	0	**100**	0	0	0
7	0	2	0	0	0	0	0	**98**	0	0
8	0	0	2	0	0	0	0	0	**98**	0
9	1	0	2	0	0	0	0	0	3	**94**

Chapter 5

Issues presented and discussion of the results

In this Chapter, the issues presented during the collection of data and processing of the images are discussed. Likewise, the analysis and discussion of the results are presented.

5.1. Data collection issues

The first stage of this thesis was related to the data collection used for testing. Some images from this collection presented specific artefacts, which influenced the results. These issues were mainly as listed below:

1. The flashlight of the camera that caused white marks on the digit itself.
2. Bleaching by the sun that led to a low contrast of the image.
3. Shadowing that produced by the angle of the camera position and also by the lack of illumination.
4. Sloped and/or curved samples due to the position of the vehicles and the shooting angle when taking photos. This caused that the cropped segments intended to contain individual digits also included parts of the neighbor digit, borders or other artefacts (See sample 3 from

5. **Figure 5.1**).
6. Blurry images due to a long shooting distance that caused low resolution.

7. Diverse marks on the license plate due to dirt.

8. Objects interfering on the license plate itself.

These issues just listed (except for sloped or curved images) caused noise in the image evaluated. Therefore, the results highly depended on the amount of noise presented in the images coming from this stage of collecting. These results are further explained in the next sections.

5.2. Managing processing issues

During the processing stage, the main issues that were dealt with were the following:

1. The value of the parameter T_0 was set to 40 as a balance between accuracy and efficiency of the algorithm. The execution of the program is less efficient when this parameter is set to a smaller value, since the iteration process takes more time, which was sometimes, depending on a given image, unacceptably high. On the other hand, when the value of this parameter was too high, the accuracy was affected. Therefore, it was necessary to find a balance.

2. In a single threshold technique, a threshold value was manually chosen for simple visual inspection and this manual set should be made for every case if aiming for a good result, but it takes a long time of work. In order to avoid this situation, the automatic global threshold was implemented.

3. The gradient feature vector allowed to analyze the image from block to block of vectors rather than pixel by pixel. This allowed for comparing images of different sizes, i.e. there was no need to resize the images before comparing them.

5.3. Discussion of the test results

The test results described in Chapter 4, are subsequently analyzed and listed:

1. Input images that have features as the sample N° 1 and as the sample N° 5 (sloped and curved digits) from

2. **Figure 5.1**, were mostly successfully recognized as long as there is not so much noise caused by other factors.
3. Images containing parts of neighbor digits and/or borders with significantly high black intensity level, as sample N° 3 and N° 8 from Figure **5.1**, mostly resulted in recognizing wrong digits.

4. Images with low resolution, as sample N° 2 from
5. **Figure 5.1**, performed a good result when compared. However, very low resolution, as sample N° 4 from the same figure, has led to classify the wrong recognition.

6. Very low contrast of the image, as shown in
7. **Figure 5.1**, sample N° 6, also led to wrong recognition.

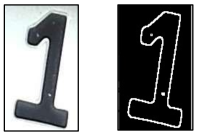
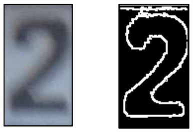
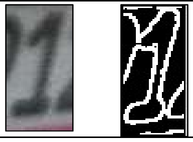
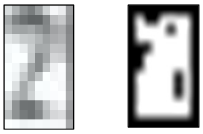
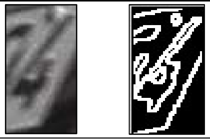
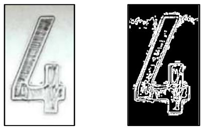
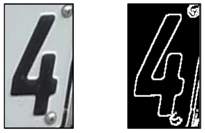
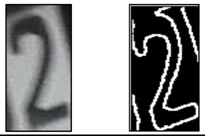
N° of sample	**Sample Image/Edges**	**Feature**	**Highest value of similarity**	**Recognition result**
1		Sloped image	0.7494907	Correctly recognized.
2		Low resolution	0.86390525	Correctly recognized.
3		Parts of neighbor digit	0.84135	Confused as *0*
4		Very low resolution	0.68056345	Confused as 3
5		Curved digit	0.7247493	Correctly recognized.
6		Low contrast	0.8701854	Confused as 0
7		Another objects on digit	0.8611992	Correctly recognized.
8		Sloped and black borders	0.7081161	Confused as 8

Figure 5.1: Samples with specific features showing their corresponding recognition result.

8. Despite the value T_0 for an automatic global thresholding was set to a relatively high value of 40, the system performance was good, is discussed below.

9. As for the confusion matrices, the main diagonal of these matrices shows the amount of successfully recognized digits and they have shown good results.
   The best recognition accuracy of 93% was achieved for the configuration $g_x = 3, g_y = 3, n = 32$, while the lowest recognition accuracy of 88% was obtained for the configuration $g_x = 2, g_y = 2, n = 8$. The summarize of those results are given in Table 5.1.

10. For the purpose of illustration, the average processing time per image for the best configuration $g_x = 3, g_y = 3, n = 32$ was 2.98 milliseconds, with standard deviation of 4 milliseconds.

Table 5.1: Recognition accuracy for each test configuration.

Configuration	Recognition accuracy
$g_x = 2, g_y = 3, n = 8$	91%
$g_x = 2, g_y = 2, n = 8$	88%
$g_x = 3, g_y = 3, n = 8$	90%
$g_x = 2, g_y = 3, n = 16$	92%
$g_x = 2, g_y = 2, n = 16$	89%
$g_x = 3, g_y = 3, n = 16$	92%
$g_x = 2, g_y = 3, n = 32$	92%
$g_x = 2, g_y = 2, n = 32$	90%
$g_x = 3, g_y = 3, n = 32$	93%

Chapter 6

Conclusions and future work

6.1. Conclusions

1. The present thesis has introduced an approach for digit recognition based on Histogram of Oriented gradients and cosine similarity, specifically focused on digits from car license plates that were taken in Germany.

2. The average gray level of the image was selected as the initial threshold value T, since the area of the object of interest has occupied more area than the background. Thus, the dominant pixels corresponded to the object itself. This situation has mostly happened because the images used were conveniently cropped and fixed as this configuration.

3. The iteration parameter T_0 for the automatic global thresholding has properly been chosen so that there is a balance between accuracy and efficiency for the execution of the algorithm.

4. A single global threshold technique is not convenient for the case of this thesis due to the high amount of samples taken, since this technique requires to set a manual threshold value, thus it requires long time of work. Hence, in this case, the automatic global threshold was more

efficient in terms of time and therefore more suitable to apply, because it automatically computes a proper threshold value for every case.

5. By using gradient features of the image, there was no necessity to resize the images for processing. Measuring the similarity from small regions to small regions of the image rather than the whole image pixel by pixel made the basis of the comparison.
6. Unsuccessful recognitions were mostly presented when images contained parts pf other objects with similar contrast levels with as the object of interest itself.
7. The recognition accuracy of 93% was reported.

6.2. Further development and future work

1. In order to overcome issues caused by the positioning of images (curved and sloped) algorithms for rotation can be applied to get better results.
2. The extension of this work to recognize a whole license plates, which includes the recognition of alphabetical characters and symbols recognition, is feasible.
3. Further filters to reduce noise coming from features described in Section 5.1 of Chapter 5 or others can be applied at the pre-processing stage in order to improve the final results.

List of Figures

Figure 1.1: Block diagram of the main stages of the process of digit recognition 9
Figure 1.2: Picture of a German license plate. 10
Figure 1.3: Standard German license plate 10
Figure 1.4: Cropped images per individual digit 10
Figure 1.5: Templates of the digits, from zero to nine, considered for the comparison analysis. 11
Figure 1.6: Some samples of the digits, from zero to nine, considered for the comparison analysis. 11
Figure 2.1: Matrix representation of an image and pixel location on the described system coordinates. (Image adopted from the next reference: [GM18]) 14
Figure 2.2: (a) Original Image ax, y and (b) its gray scale image conversion agx, y 17
Figure 2.3: Non-normalized Histogram of gray levels of the digit *0* image shown in Figure 2.4 (a) 18
Figure 2.4: (a) Gray scale image of the digit *0*, and (b) its histogram computed in *Processing* .. 19
Figure 2.5: Gray level histogram that can be partitioned by a single threshold 20
Figure 2.6: (a) Input color image of digit *2*, (b) its gray scale image, (c) its threshold value with a midway, (d) its histogram computed in *Processing*, and (e) its histogram showing gray levels plotted in excel 21
Figure 2.7: (a) Input color image, (b) gray scale of the input image, (c) result of a applying a higher threshold value *T* than desired, and (d) result of a applying a lower threshold value *T* than desired 22
Figure 2.8: (a) gray scale image of the digit *2*, and (b) its histogram computed in *Processing*. . 23
Figure 2.9: (a) Input images of digit *2* and *7*, with display windows from *Processing* of: (b) its Gray scale Image, (c) thresholded images when applied the same midway *T* assumed for the digit *2* as illustrated in Figure 2.6. 23
Figure 2.10: Gray level histogram that can be partitioned by a single threshold 24
Figure 2.11: (a) Input images, with display windows of *Processing* of: (b) gray scale images, (c) thresholded images of digit *2* and *7* when applied the automatic global thresholding, ... 26
Figure 2.12: A filter of dimension *3 x 3* [GM18]. 27
Figure 2.13: Eight-neighborhood of the image $f(x, y)$ with its center point at pixel (x, y) (Image adopted from the next reference: [GM18]). 28
Figure 2.14: Sobel filters: (a) horizontal, and (b) vertical [GM18]. 30
Figure 2.15: Graphical summary of the processed image up to this section 32
Figure 2.16: Display windows of *Processing*, showing the application of Sobel filters: (a) input image (b) horizontal, (c) vertical, and (d) horizontal and vertical combined 32
Figure 2.17: (a) Input images of the digit *2* and *7*, with display windows of *Processing* of: (b) their gray scale images, (c) their thresholded images, and (d) their output when applied Sobel filter. 33
Figure 3.1: (a) eight chain code directions, and (b) n chain code directions. 35
Figure 3.2: Representation of the array of feature vectors. 36
Figure 3.3: (a) n chain code directions, and (b) representation of the gradient direction $\theta x, y$ with its magnitude gx, y placed in the array of feature vectors. 37
Figure 4.1: Templates of the digits, from zero to nine, considered for the comparison analysis. 43
Figure 4.2: Small extract of the collection of pictures of German car license plates 43
Figure 4.3: Pictures of license plates with particular features. 44
Figure 4.4: Small extract of the dataset of the cropped images from the collection of pictures shown in Figure 4.2 used as samples and grouped as per their corresponding digit. 45

Figure 4.5: An example of the comparison between a sample image with all the template images of digits from *0* to *9*..47
Figure 5.1: Samples with specific features showing their corresponding recognition result.... 55

List of Tables

Table 4.1: Set of configurations used for testing. 48
Table 4.2: Confusion Matrix for $gx = 2, gy = 3, n = 8$ 49
Table 4.3: Confusion Matrix for $gx = 2, gy = 2, n = 8$ 49
Table 4.4: Confusion Matrix for $gx = 3, gy = 3, n = 8$ 49
Table 4.5: Confusion Matrix for $gx = 2, gy = 3, n = 16$ 50
Table 4.6: Confusion Matrix for $gx = 2, gy = 2, n = 16$ 50
Table 4.7: Confusion Matrix for $gx = 3, gy = 3, n = 16$ 50
Table 4.8: Confusion Matrix for $gx = 2, gy = 3, n = 32$ 51
Table 4.9: Confusion Matrix for $gx = 2, gy = 2, n = 32$ 51
Table 4.10: Confusion Matrix for $gx = 3, gy = 3, n = 32$ 51
Table 5.1: Recognition accuracy for each test configuration. 55

Bibliography

[GW18] Rafael C. Gonzales and Ricahrd E. Woods, *Digital Image Processing (Third edition).* University of Tennessee, Tennessee, United States, 2002.

[GG13] Samta Gupta and Susmita Ghosh Mazumdar. *Sobel Edge Detection Algorithm.* International Journal of Computer Science and Management Research, Vol 2, Issue 2. Bhilai, India, 2013.

[DT05] N. Dalal and B. Triggs. *Histograms of Oriented Gradients for Human Detection* – in: Proc. of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). Montbonnot 38334, France, 2005.

[MA19] Milan Gnjatović, Nemanja Maček and Saša Adamović. *A Non-Connectionist Two-Stage Approach to Digit Recognition in the Presence of* Noise – in: the Proc. of the 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom), Naples, Italy, 2019.

[GM18] Milan Gnjatovic. *Lectures Notes in Image Processing.* SRH Hochschule Heidelberg, Engineering and Architecture Department, Germany, 2018.
Available: http://gnjatovic.info/imageprocessing/

[SS00] Linda Shapiro and George Stockman. *Computer Vision.* The University of Washington and Michigan State University, United States, 2000.

[MK15] Samuel Macdo, Givanio Melo and Judith Kelner. *A comparative study of grayscale conversion techniques applied to SIFT descriptors.* SBC Journal on Interactive Systems, volume 6, number 2, Brazil, 2015.

[P3] *Processing* [Online]. Available: https://processing.org/. Access date: 24.06.2019.

[LP] German license plates [Online]. Available: https://picclick.com/NEW-European-German-License-Plates-for-BMW-VW-292634582888.html. Access date: 16.08.2019.