Text Detection and Recognition in the Automotive Context

by

El Hebri Khiari

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M. degree in
Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

This thesis achieved the goal of obtaining high accuracy rates (precision and recall) in a real-time system that detects and recognizes text in the automotive context. For the sake of simplicity, this work targets two Objects of Interest (OOIs): North American (NA) traffic boards (TBs) and license plates (LPs).

The proposed approach adopts a hybrid detection module consisting of a Connected Component Analysis (CCA) step followed by a Texture Analysis (TA) step. An initial set of candidates is extracted by highlighting the Maximally Stable Extremal Regions (MSERs). Each sebsequent step in the CCA and TA steps attempts to reduce the size of the set by filtering out false positives and retaining the true positives. The final set of candidates is fed into a recognition stage that integrates an open source Optical Character Reader (OCR) into the framework by using two additional steps that serve the purpose of minimizing false readings as well as the incurred delays.

A set of of manually taken videos from various regions of Ottawa were used to evaluate the performance of the system, using precision, recall and latency as metrics. The high precision and recall values reflect the proposed approach's ability in removing false positives and retaining the true positives, respectively, while the low latency values deem it suitable for the automotive context. Moreover, the ability to detect two OOIs of varying appearances demonstrates the flexbility that is featured by the hybrid detection module.

Acknowledgements

First and foremost, I thank God for granting me the mental and physical abilties to write this thesis in a timely manner.

I thank my supervisor, professor Azzedine Boukerche. His technical support and expertise were paramount in providing me the necessary environment, as well as the much needed financial support. I also thank my close and dear friend, Amar Farouk Merah, for his infinite and unconditional help in helping me start my masters program as well as in helping me cope with this new environment with ease. I also thank my mentor, Dr. Abdelhamid Mammeri, for his consistent and uncompromising guidance throughout the duration of this work. Additionally, I thank my colleagues in PARADISE lab that sacrificed their personal time in order to help me advance in my work.

Last but not least, I send my deepest and most sincere gratitudes to my parents. With their passion and unconditional love, I was able to deal with every obstable, big and small, that life threw at me.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Problems

In a time where technology is at an all-time high and vehicles are at an abundance, the need to incorporate systems that assist drivers has never been higher. This task is fulfilled by Advanced Driving Assistance Systems (ADAS). Such systems consist of several modules, such as a pedestrian detection [61] [57], an animal detection [59], a lane detection [60], and a traffic sign recognition module [56] [58]. A vehicle would use said modules to assist the driver as well as gather information about the state of the road for it to shared with other nearby vehicles, using various Vehicular Ad-Hoc and Mobile Ad-Hoc networking technologies (VANETs and MANETs) [18] [4] [96] [14]. The purpose of this work is to add improvements to current ADAS by incorproating a text detection and recognition module. This module aims to extract text regions from important objects in the automotive context, read the detected texts and display them to the user in a time-efficient manner. In achieving this task, the driving experience of a user would be streamlined as the need to read important text-rich signs will be removed.

The field of text detection can be divided into two domain-specific sub-fields: document text detection, and text detection from natural scenes. The two domains impose different sets of challenges. Besides lighting effects, an issue found in nearly any object recognition sub-problem, little to no factors play a role in document text detection. Text detection from natural scenes, however, get affected by several more factors [101]. Some of these factors are factors are:

- *Latency*: For most natural scene text detection systems, minimizing delay is a priority. This forces the researcher to search for a tradeoff in achieving high accuracy rates at the expense of high latencies, and vice versa. This is not the case with document text detection, however, as such systems are almost entirely delay-independant.
- *Text appearance*: Documents generally have the same format, font and color, with slight differences in line spacing and margins that, however, will hardly have any

impact on the detection and classification processes. Text found in natural scenes, on the other hand, can be of any font, shape and color, thus complicating the task of the system.

- *Presence of obstacles*: The likelihood of a foreign object coming between the camera and the document is very low. This is a very common phenomenon in natural scenes, however, as the background is constituted primarily of unwanted objects (pedestrians, for example). The presence of such obstacles would negatively impact the process of detecting the objects of interest, as well as the process of transmitting the necessary information to nearby entities [1].

These challenges are the reason behind the increased attention this field has been getting in the past few years [97]. The above three points, however, do not fully cover the challenges found in text detection from natural scenes. There exist many different kinds of potentially useful text in a natural image, each of which may have additional challenges associated with it. One such kind, which is the primary concern of this work, is the text in the **Automotive Context**. The text in the automotive context consists of text that could be used to aid the driver of a vehicle. Such text could be the authorized speed limit in a highway, road-signs giving an order (STOP, for example), signs that describe the nature of the road ahead or signs giving directions to specific places.

It is important to note that text detection and recognition in the automotive context imposes a few additional challenges. These challengers are (Figure 1.1):

- *Road Conditions*: Unless an image stabilization system is used, the camera mounted on the vehicle will produce shaky images due to the roughness of the road.
- *State of road-signs*: Road-signs suffer from deterioration in quality with time from the sun's ultra-violet (UV) lights, thus affecting the text printed on it.
- *Lack of publicly available datasets*: While certain types of traffic signs have publicly available datasets tailored for them, many other text-rich signs (example Figure 1.2) have no dataset available for them. This renders the task of training a classifier, the most common approach in object recognition, almost impossible.
- *Real-time performance*: Unlike with stationary text detection and recognition, minimizing the delay to real-time values is a primary goal in the automotive context.

1.2 Motivation and Contribution

1.2.1 Motivation

Text detection and recognition in the automotive context remains a relatively untouched field of research due to the challenges mentioned. With difficulties arising from unmanageable factors, such as bad weather conditions, unwanted obstacles, blurry appearance of

Figure 1.1: Some of the challenges found in the automtoive context: presence of many unneeded objects, varying sizes, bad weather conditions, blurry texts

text, and lighting effects, tackling some of these challenges can be tedious. However, we believe that their effect(s) can be mitigated. As such, we are motivated to discover ways in which this can be achieved, whilst maintaining real-time performance.

Objects of Interst (OOIs)

Our Objects of Interst (OOIs) for this work are: North Amercian (NA) Traffic Boards (TBs) (Figure 1.2) and License plates (LPs). The following are the reasons for choosing these two objects:

- Detecting and recognizing traffic boards is an untouched and challenging task compared to traffic sign detection and recognition. The gap in the difficulty of detecting and recognizing both objects stems from the fact that traffic signs come in predetermined shapes and sizes [34], i.e. the number of different traffic sign types is small. Traffic boards, on the other hand, contain text that is dependant on the geographical location of the the board and therefore a character and/word recognizer needs to be utilized as opposed to a simple shape classifier for a traffic sign. These factors, in addition to the fact that traffic boards contain very informative texts, motivated us to tackle this problem.

- While license plate detection and recongition has been tackled previously, it still poses challenges in the automotive context due to the relatively small dimensions license plates come with. Moreover, real-time detection of license plates has not been a primary objective in most previous works. More importantly, the detection of license plates would serve as a flexbility measure for our system, i.e. a high degree of flexibility would be exhibited in being able to detect two objects of differing appearances, traffic boards and license plates, using the same framework.

Figure 1.2: No publicly available datasets for text-rich boards

1.2.2 Contribution

As proven in Section 5, this work succeeded in achieving high accuracies in detecting and recognizing text from the automotive context, all the while maintaining real-time performance. As Section 4 explains in details, this was achieved by integrating a hybrid detection module with a highly reputed Optical Character Recognizer (OCR). The detection module utilizes a novel texture analysis approach that separates text-rich objects from text-light objects. Therefore, with the right parameters in each step of the framework, our system can be used to detect and recognize texts that appear in many different objects in addition the OOIs mentioned above.

1.3 System Overview

A standard architecture for solving a general OR problem consists of two main steps, **detection** of the object(s) of interest and **recognition/classification** of the object(s), with additional **pre-processing** and **post-processing** steps. The pre-processing step aims to improve the results of the succeeding detection step through various image processing operations such as image filtering, image smoothing, resolution adjustments, etc. The detection step then extracts the coordinates of the object(s) of interest. The coordinates are fed to the recognition/classification step, where a classifier and/or recognizer is used. Finally, the post-processing step attempts to add final improvements to the results produced by the recognition step.

Our system adopts a similar approach. The pre-processing step attempts to alter the overall look of each frame in order to improve the performance of the upcoming steps.

Locating the coordinates of the OOIs is achieved in the detection step, which are then fed into the recognition step. A final post-processing step attempts to further improve the recongition results in order to provide the user with the most accurate texts possible. Figure 1.3 provides a simplified illustration. More details on the approach are given in Section 3, along with a more detailed flowchart.

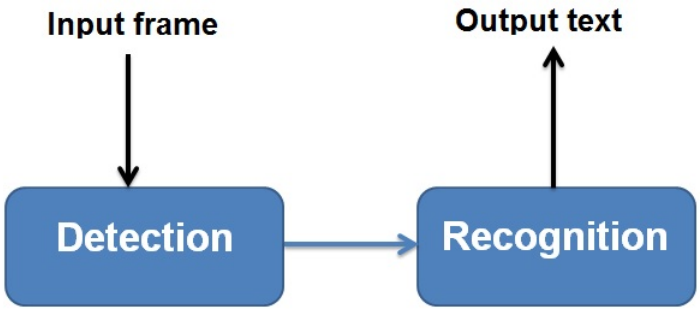


Figure 1.3: Simplified system overview

1.4 Thesis Outline

This thesis is outlined as follows:

- Chapter 2 provides an elaborate literature review of the field of text detection and recognition. It does so by listing and describing the most popular techniques used and the most notable works done in this field. Finally, it outlines the techniques and/or ideas that could be used in our system.
- Chapter 3 explains the proposed architecture. Firstly, it outlines the reasoning behind adopting a hybrid detection module, and, secondly, explains the purpose and logic of each step in the module.
- Chapter 4 provides details on the recognition module. ThThe purpose of selecting Tesseract, the OCR of choice, is firstly explained, followed by the purpose and logic of the post-processing step.
- Chapter 5 explains the processes that are taken to evaluate the performance of our system. It also includes comparisons with recent highly regarded works in the field of text detection and recognition.
- Chapter 6 concludes this thesis by providing a summary of the work.

Chapter 2

Related Works

2.1 Adopted Approaches

The works done in the literature of text detection follow one of two main classes: Contour-based approaches, or Texture-based approaches. Each class differs vastly from the other class, with advantages and disadvantages associated with each depending on the nature of the application. In some cases, both texture and contour-based features are used together, thus creating a **Hybrid** system. Before the two classes are elaborated upon, some commonly used techniques that are frequently used in addition to contour and/or texture-based approaches are briefly explained.

2.2 Commonly Used Techniques

These techniques are either used as a complimentary step to a contour and/or texture extraction technique, or as a vital step to the task of detection of text. They usually come in the form of a feature descriptor, feature classifier or a statistical model. The following subsections briefly explain the most commonly used feature descriptor, feature classifier and statistical model used in this field, based on the sheer number of times they were used in previous works.

2.2.1 Histogram of Oriented Gradients (HOG) - *Feature Description*

In 2005, Dalal and Triggs in [25] proposed a new set of feature descriptors for object recognition, initially designed for human detection, that they called *Histogram of Oriented Gradients (HoG)*. HoG works by computing the intensity gradients or edge directions of pixels within a divided region of the image, called "cell", and forming a histogram of the gradient orientations or edge directions. In other words, the gradient orientation occurrences of each cell are computed. As HoG operates on localized regions in the image, it

is invariant to geometric and photometric transformations thus gaining an advantage over contemporary feature descriptors such as *Scale-invariant feature transform (SIFT)* [51]. To improve accuracies, the computed local histograms are contrast-normalized so that invariance to illumination and shadowing is obtained. This is also done locally but with larger regions. Intensity values over a group of cells, called "blocks", will be measured and used to normalize the corresponding block. The resulting features can then be fed into a linear Support Vector Machine (SVM) thus producing a solid object detector/classifier. Figure 2.1 shows every step that is involved in producing an object (humans in this example) detector/classifier, taken from [25].
[more Figures]

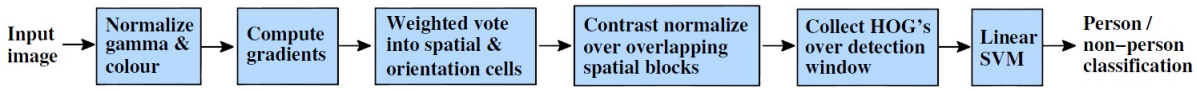


Figure 2.1: The steps involved in producing a human detector/classifier using HoG with a linear SVM, taken from [25]

2.2.2 Support Vector Machine (SVM) - *Feature Classifier*

A Support Vector Machine (SVM) is a *linear binary* classifier [23]. The 'binary' label means that SVM classifies a given input into one of two classes (in text detection, the two classes are 'text' and 'not text'). The 'linear' label refers to the linear decision function that the SVM constructs in order to classify a given input. This decision functions represents a hyper-plane that aims to separate the two classes in a high-dimensional space as much as possible. This is precisely done by optimizing the function $f(x) = w.\psi(x) + b$ in such a way that the distance between $\psi(x_i)$, the position of a point x, and the hyper-plane is maximized.

Despite being initially designed as a linear classifier, SVMs can also be used for *non-linear* classification, as suggested in [6]. This can be achieved by applying the *kernel trick* (first proposed in [2]), in which each dot product is replaced by a non-linear kernel (radial basis kernel, for example). As a result, a non-linear decision function is constructed that, similar to linear classification, will be used to classify a given input to one of the two classes. Figures 2.2 and 2.3)[1] illustrate the linear and non-linear classification processes.

Additionally, SVMs can be altered to carry out *multi-classification*. This can achieved through the transformation (specifically *reduction)* of the multi-class problem to a binary problem. Several methods exist for this transformation [28], such as carrying out a *one-versus-all* or a *one-versus-one* strategy.

[1]Taken from http://en.wikipedia.org/wiki/Support_vector_machine

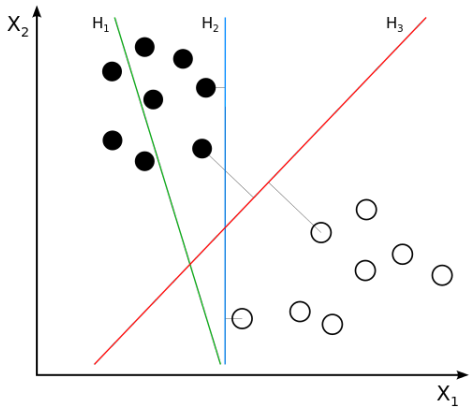


Figure 2.2: **Linear Classification**: hyper-plane H_1 fails to separate the classes, H_2 separates the classes but is far from optimal, H_3 classifies the two classes optimally

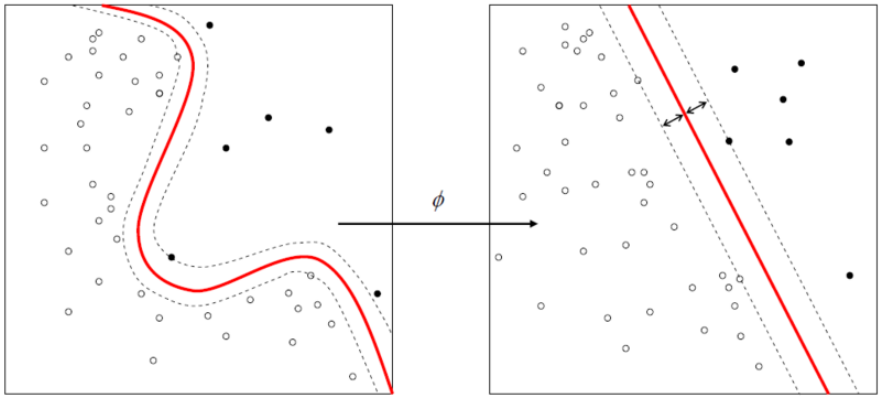


Figure 2.3: **Non-linear Classification**: application of a kernel machine

2.2.3 Conditional Random Field (CRF) - *Statistical Model*

Conditional Random Field (CRF) [45] is a *graphical probabilistic* model that is used to label sequential data in various fields such as natural language processing, bioinformatics, computer vision, etc. Rather than predicting single labels or values, as is done with most supervised machine learning algorithms (SVM, for example), CRF carries out *structured prediction* where it predicts structured objects.

Since CRF is a graphical model, a graph is used to represent the dependencies (Edges) between the random variables (Vertices) [13] [20]. Given set of observations $X = (x_1, x_2, ..., x_n)$, a set of random variables $Y = (y_1, y_2, ..., y_n)$, a graph $G = (V, E)$ where Y is indexed by V, the conditional probability (X, Y) is a CRF if the probability of Y that is conditioned by X follows the Markovian property:

$$P(y_i|X, y_i, j \neq i) = P(y_i|X, y_i, j \in N_i) \tag{2.1}$$

where N_i is the set of neighbors of x_i.

2.3 Contour-based approaches

Most contour-based approaches consist of two steps: **Edge (contour) Detection** and **Connected Component (CC) Analysis**. The first step aims to extract the edges, or contours, from an image. The curvilinear continuity found between the extracted edges can then used to link edges as needed. An important thresholding intermediate step needs be to added, however, in order to eliminate any excess edges [55]. An example is illustrated in Figure 2.4.

The next subsections will elaborate more on Edge detection and Connected component analysis, listing the various edge detectors used in the literature and how CC analysis plays an important role in text detection.

2.3.1 Edge Detection [64]

An edge is the boundary that separates two 'different' regions in a given image. The difference between two regions is based on their distinct characteristics adhering to some feature(s), such as gray-level, color, texture, etc. Edge detection aims to find those edges that constitute the general outlook of an image through various mathematical procedures. This task, however, is anything but trivial since many unavoidable factors, such as illumination, size, positioning of objects, contrast between objects and the background, shadow, occlusion, etc., heavily impact the process of edge detection. As a result, the resulting segmented image will contain an excess of unneeded edges. Therefore, most counter-based strategies incorporate an additional pre-processing step that involves noise-reduction, illumination or smoothing operations. This, however, does not fully solve the problem, i.e. some unwanted edges will still be detected. In an attempt to prune these unwanted edges, an additional post-processing step can be used. This step can be a simple thresholding step, or it can find objects that take a specific shape or form (rectangle, triangle, etc.) or, in some applications, it can use line detection techniques (Hough transform, for instance) to transform broken edges into meaningful lines. In a nutshell, a standard, full, edge detection step consists of three steps: *Noise Reduction (pre-processing)*, *Edge points detection* and *Edge Localization (Post-processing)*.

Over the years, several edge detectors have been implemented. The following two are the mostly used ones:

- *Sobel Operator*
  The sobel operator operates firstly by convolving the image with an integer-valued filter, both vertically and horizontally. It then approximates the gradient of image intensity function, where, at each step, the result obtained is the corresponding gradient vector or its norm. The computational cost of the sobel operator is relatively inexpensive, but at the cost of crude gradient computations. Figure 2.5 shows an example of an image resulting from a sobel operator.

- *Canny Detector*
  Seen as one of the most powerful and effective edge operators to date, the canny detector surpassed most detectors and became the favorite among practitioners in object recognition. Firstly, the input image is smoothed with a Gaussian LPF, with specific values for its parameter σ. This value, if set large enough, will suppress most of the noise at the cost of rendering potentially-relevant edges weak. The local gradient of each point is then computed, resulting in ridges of large widths. A process known as 'nonmaximal suppression' is then used to thin those edges. Two thresholds, T_{low} and T_{high}, are then used to used to classify the remaining ridges to one of two classes of edges: *weak* edges ($> T_{low} and \leq T_{high}$) and *strong* edges ($> T_{high}$). As a final step, weak edges that are 8-connected to strong pixels are combined. Figure 2.4 shows an example of using a canny edge detector on an image.

Figure 2.4: Left: Original Image, Middle: Output with Canny edge detector, Right: Output with Canny edge detector + threshold

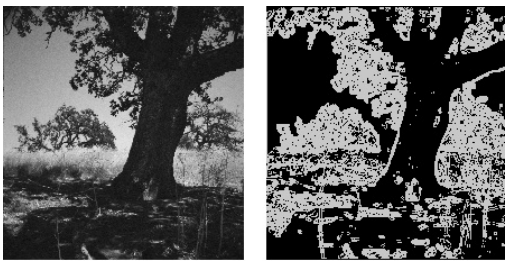


Figure 2.5: Left: Original Image (greyscale), Right: Output with Sobel operator

2.3.2 Connected Component (CC) Analysis

Connecting the components of an image has been a staple image processing step for over thirty years, to the extent that the terms "contour analysis" and "connected component analysis" can be use interchangeably. It takes a 2-dimensional or 3-dimensional image as input, converts it into a binary image and labels the pixels based on their pixel values [27]. According to [54]], the process of connected component labeling can be described as a transformation of the input image I, in its binary format, into a new image I' such that

1. All image elements (pixels) having a 0 (white) value in I will remain unchanged in I'

2. Every maximally connected subset of I of 1 (black) values will be labeled by a distinct positive integer in I'

A connected subset (component) consists of adjacent 1-value image elements, be it vertical or horizontal [82]. The *4-adjacent* subset is the most commonly used subset size, although *8-adjacent* can also be used [27]. A region is a *maximal 4-connected set* of 1-value image elements if, for pixels p and q, for a sequence of image elements $p = p_0, p_1, p_2, ..., p_n = q$, p_i is 4-adjacent to p_{i+1}, for $0 \leq i < n$ [82]. The same concept can be used with blocks, where two blocks P and Q are 4-adjacent if a pixel p in P and a pixel q in Q are 4-adjacent. Th same analogy can be used for 8-adjacent pixels (Figure 2.6)[2] and/or blocks.

In the field of text detection, the components mentioned above can be either words or characters. These words and/or characters each have distinct geometric properties where neighboring words/characters share similar spatial and geometric properties [77]. As a result, CC-based approaches in text detection follow three steps (steps 2 and 3 can be interchanged):

1. *CC Extraction*: Segment candidate text component.

2. *CC Analysis*: Filter out outliers, i.e. non-text components, through the use of user-defined rules (heuristics) that are based on the geometric and spatial properties of the components.

3. *Post-processing*: Group text blocks, i.e. words, on the same line.

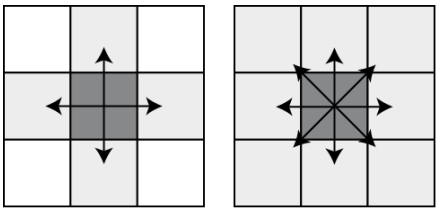


Figure 2.6: Left: 4-connectivity, Right: 8-connectivity [39]

Architectures that employ connected component algorithms differ in the way they extract the components and group them. The coming subsection explores the previous works that follow the CC-based architecture. (might remove)

[2]Taken from https://www.ualberta.ca/~ccwj/teaching/

2.3.3 Previous Works

In [50], Lui and Sarkar develop a system that consists of three main parts

1. A basic CC text detection framework,
2. an Intensity filter step, and
3. a Shape filter step.

The first step aims to extract the components and group them into words. The modified Niblack dynamic thresholding algorithm [72]14C is used to carry out the initial segmentation. The sizes of the components are then used to prune overly large and overly small components. The geometric and intensity guidelines used in the DAR community [32][33] are then used to group components to generate text regions. These guidelines are:

- Same x-axis alignment, based on the vertical distance between their *top* pixels,
- Similar intensities based on the distance between their intensity histograms,
- Similar font sizes based on their heights, and
- Close proximity based on their Euclidean distance.

The intensity filter is then used to filter out non-text components. This is done by computing the overlap between the intensity histograms of a component C_i and its adjoining area NC_i, $S(C_i, NC_i) = \sum_{k=0}^{255} min(h_{C_i}(k), h_{NC_i}(k))$. A threshold T_1 is then used to filter out any overlap that exceeds it. This is based on the authors' observation that text components show better contrast with their adjoining components than non-text components. Therefore, the smaller the overlap, the less similar the component and its adjoining area are, the less likely it is that text is present in this component. The process of thresholding is done iteratively, where in each iteration the threshold is lowered and more outliers are removed. This is repeated until no additional component is removed. The shape filter step then follows in order to prune non-text shapes. This step follows the assumption that a word is constituted from different objects (characters), and therefore a component that is formed from the same object will be disregarded. The authors used the inner distance shape model proposed in [47] to quantify the shape for each component H. The model is based on the largest *inner distance* of all pixel pairs within a component C_i, where the inner distance between two pixel is defined as the shortest distance between the two pixels inside C_i. The affinity between the components in a region R is then computed based on their distances, where another threshold T_2 is used. Any negative D values are disregarded, as well as values from regions with one or two components. This condition is taken from [32] where it was stated that most words consist of three or more objects (characters).
The authors tested their three-step approach on the dataset taken from the ICDAR 2003 challenge. They used performance measures taken from the challenge to evaluate their results [91], namely precision, recall and the combined rates. Their results were based on

249 images with varied resolutions and compared them to the best five algorithms listed in [91]]. Based on the results published in their paper, the results produced outperformed the best five algorithms from the ICDAR 2003 challenge, especially in the precision rate. The images used for testing were taken from natural scenes. However, the automotive context was not taken into consideration. The images used were taken from still camera shots and, therefore, the challenges found in the automotive context did not play a role. Moreover, the authors' approach was strictly text *detection* and did not incorporate any *recognition* step.

In [73], Ezaki et al. designed a system for extracting text from natural images for the visually impaired. Their approach consists of three main steps:

- Text detection from natural image,
- character recognition, and
- text-to-Speech synthesis

The authors target *small* (height $<= 30$) and *large* characters differently. For small characters, the erosion operation (morphological operation) is first performed followed by a top-hat processing. Top-hat processing is carried out by computing the difference in intensity values of the original and output eroded image; similar to what was done in [50] as mentioned earlier. After the top-hat processing is done, the resulting image is binarized and the CCs are extracted. To only detect text, the authors follow the assumption that western texts follow horizontal patterns and apply a simple filter to extract such areas. This approach cannot be used to extract large characters (height > 30), however, as it would be computationally infeasible. To tackle this issue the authors tested three methods, each of which is applied after the image is zoomed into the required area. These methods are:

- *Character extraction from Edge image*
  The Sobel edge operator is applied to each color channel from the RGB image. The maximum value of each pixel is then used to combine the three resulting images. Otsu's method [75] is then used to binarize the final image, from which CCs are extracted. If characters are clumped or when the background is non-uniform, however, some text-containing components are discarded (result from the selection rules), thus rendering this method unreliable.

- *Character extraction from Reverse Edge image*
  This method is similar to the previous one with the additional step of reversing the binarized image. It fails when the characters are surrounded by connected edges and the printed characters are not broken.

- *Color-based character extraction*
  This method takes a different approach from the methods mentioned so far. Rather than extracting CCs using morphological and edge operators, this method uses the fact that characters within a word are almost always of the same color. Therefore, the

authors applied Otsu's binarization method on each of the RGB channels of the image, resulting in a total number of 8 possible colors for each pixel. The three resulting binary images are separated, then have CCs extracted from them independently.

With following the assumption that characters of a word share similar dimensions (in addition the the assumption previously mentioned), the authors were able to locate large characters using some simple constraints. The constraints they used are:

- $0.1 < W_i/H_i < 2$ (Aspect Ratio)
- $W_i * H_i > 50$ (Area)
- $\Delta y < 0.2 * Max(H_i, H_j)$
- $\Delta x < 2 * Max(W_i, H_j)$
  Δy *and* Δx *are the distances between the centers of the areas between one component* C_i *and the succeeding component* C_j

The authors used 504 images containing text from natural scenes taken from *Trial* Section of the ICDAR 2003 Robust Reading Competition [91]. Precision and recall rates, and their combined f-value were used to evaluate each of the three methods mentioned. Due to the existence of many large characters, the morphological-based method gave less than satisfactory results with an f-value of 0.28. The edge-based (no inversion) approach produced the highest f-value of 0.62. The authors the attempted to investigate the results of combining the different methods using the OR operator. As expected, all f-values were fixed to the highest value of 0.62, but with noticeable changes in the precision and recall rates. Priority was given to the combination that gave the highest recall rate, which resulted from combining all four methods.

In [100], Zhang et al designed Conditional Random Fields (CRF)-based system to detect and extract text from scene images. The system consists of three stages: CC extraction, CRF classification and CC filtering, all of which are executed twice over two iterations. Following the common the claim that texts within an image tend to be of different color than the background, the authors binarized every image using the Niblack algorithm [72]. The CCs are then extracted from the binarized image and filtered according to some size and aspect ratio constraints. The same two constraints in addition to the *Average Gradient* of each CC are used as features for the classification stage. The CRF is modeled with two *Neighborhood Relationship Graphs (NRG)*, a strict NRG and a relaxed NRG. The first iteration utilizes the strict NRG anlabels CCs as "text" or "non-text". The "text" CCs are sent to strict OCR module that gives confidence values that indicate whether a given CC is a text line or not. If a CC is not a considered text line, it is labeled as an "uncertain" CC and fed to the second iteration where the same steps are executed but with a relaxed NRG and OCR filter. The results of the both iterations are used to give estimated text regions. While this CRF scheme proved fairly effective at extracting multiple text lines, the precision and recall rates produced are not up to par with the winner of the ICDAR 2005 competition.

In [35], Greenhalgh and Mirmehdi designed a real-time system for detecting traffic signs. Their system consists of the two classical steps: detection and recognition. The detection starts by using *Maximally Stable Extermal Regions (MSER)*, which takes an image from different viewpoints and attempts to find the correspondence between them [66]. To do so, a given RGB image is first normalized into the desired color channel; for this paper three colors were of concern, white, blue and red. Figure 2.7 shows shows how a red-blue normalized image looks like. For the white color, since no channel exists for it, the MSER regions are found on the greyscale equivalent. For all colors, a thresholding step follows to determine which components retain their shape thus knowing what components to disregard. Feature such as height, width, area, aspect ration, etc., are used to further prune unwanted components. The *Histogram of Oriented Gradients (HOG)* features of the detected traffic signs are then used, with the help of the Sobel operator, for recognition. By training a many-to-one linear binary Support Vector Machines (SVM) on the HOG features, multi-class classification was achieved. The dataset used for training was taken from U.K. Department for Transportation. For each of the 131 images used, 1200 geometric distortions were created in order to create a solid dataset that properly represents the various conditions traffic signs are found on roads. The precision, recall and F1 measures produced from 3 videos with 14 traffic signs in total were 86.8%, 80.7% and 0.84 respectively.

In [42], Koo and Kim focused more on deciding whether a given CC or region contains and/or is a text candidate. To do so, they divided their work into their steps: *candidate generation*, *candidate normalization* and *non-text filtering*. Candidate generation was done through MSER extraction (as done in [35]) for its low computation cost [43]. An ICDAR 2011 [85] ground truth was augmented to train an AdaBoost classifier that will be used to cluster the CCs based on their adjacency relations. Afterwards, for the candidate normalization step, the images are geometrically normalized and then binarized. The binarization is based on estimated text and background colors. The text color is computed by averaging the CCs color while the background color is computed by averaging the color of an entire block. The third step, non-text filtering, is carried out by extracting features from overlapping blocks and feeding them to a trained multi-layer perceptron. For each block, a decision (text or no text) will be made. The decisions for all blocks will then be integrated in order to produce a final decision for each block. The authors tested their system on the ICDAR 2005 and 2011 datasets, with $f-measure$ values of 0.7452 and 0.708 respectively.

2.4 Texture-based Approaches

In this approach, the texture of an image is used for segmentation based on the spatial distribution of colors or intensities over regions within the image. The image texture is simply a set of highly descriptive metrics that are computed from an image. What makes textural features particularly interesting is the ability for them to be described with various variables, such as coarseness, direction, contrast, etc [38]. However, despite containing rich

Figure 2.7: Left: Original image, Right: Red-Blue Normalized image, taken from [35]

information, defining texture and analyzing it has been a challenging task for computers since it is an innate quality found on all surfaces and are structured specifically for humans to perceive [37].
. According to [92], texture analysis methods fall into one of two broad categories, *Statistical* where primitive pixel-based features (size, position) are used as features, and *Syntactic* where broader properties (brightness, shape) are used. The following three subsections briefly explain some of the more commonly used texture analysis techniques that have been used previously.

2.4.1 Co-occurrence

Using co-occurrence features was one of the first statistical approaches to texture analysis. Back in 1979, Haralick in [36]] used second-order statistics gathered from *Grey Level Co-occurrence Matrix (GLCM)* [38]. The statistics found in a GLCM provide information on the frequency of certain pixel values and their positioning in a given image. More precisely, the value found in an element (i, j) of a GLCM contains the *probability* of an i-value pixel being located next to a j-value pixel. Ultimately, a GLCM is a neat arrangement of the frequencies of different pixel brightness that occur in an image [83].
The advantage of using second-order statistics over first-order statistics is the exploitation of spatial pixel properties by taking into account the relation between the location of two pixels. While third-order statistics are able to exploit the spatial relations between three pixels, their complexity and computational cost deterred practitioners from implementing them [83].

2.4.2 Tamura

In [94], Tamura et al selected three features out of six that, based on psychological measurements for human subjects, they deemed as the most important features for describing

texture. These features are [38]:

- *Coarseness*
  Seen as the most important feature, coarseness identifies the largest size in which texture exists. *[Include mathematical details]*

- *Contrast*
  The dynamic range of grey levels are captured along with the polarization of white and black pixels. *[Include mathematical details]*

- *Directionality*
  This feature aims to measure the total directionality of a region, and therefore is considered a global property. *[Include mathematical details]*

When each of these three features is computed for each pixel, a *Coarseness-Contrast-Directionality (CND)* distribution is formed. An image presented in this distribution is hence referred to as a *Tamura* image.

2.4.3 Gabor Filter

This linear filter is regarded as one of the most popular filters for texture analysis. It owes its popularity to its ability to accurately depict the responses of the human visual system. It does so by enabling filtering in both the frequency and the spatial domains [38]. Turner was one of the first practitioners to implement a gabor filter, where in [95] he enabled multi-channel filtering by using a bank of filters at different orientations and scales. This way he was able extract the necessary orientation and frequency information to produce the required texture features from the image. A sample gabor-filtered image is shown in Figure 2.8 *(Some extra details)*

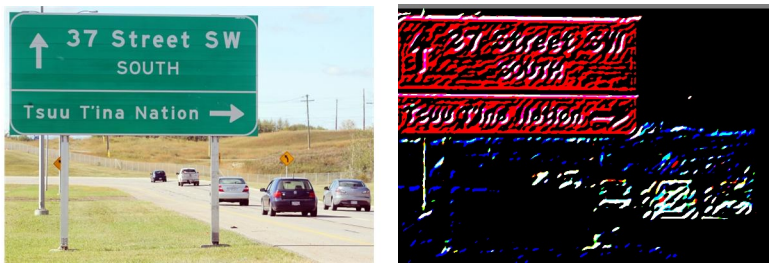


Figure 2.8: Left: Original Image, Right: Output with Gabor filter

2.4.4 Local Binary Pattern (LBP)

LBP has been widely considered as one of the most powerful features for texture classification ever since it was introduced by Ojala et al in [74]. In simple terms, LBP uses the

central pixel within a cell to threshold the neighboring 8 pixels thus giving an 8-bit binary number, as shown in Figure 2.9. A histogram for each is cell is formed. The histograms are then concatenated to create a feature vector, which can be used in a classifier such as SVM.

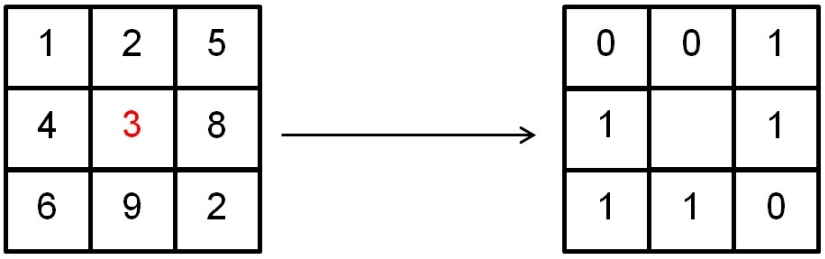


Figure 2.9: LBP for one cell, resulting binary pattern: 00110111

2.4.5 Previous Works

An example of a straightforward, simple and fairly effective texture-based approach to detecting text from natural scenes was done in [62]. It consists of only two steps. The first step uses a combination of wavelet transform and gabor filter to extract sharpened edges for texture information. The second step works detects the text areas by applying wavelet entropy to retrieve the average energy of the gabor-filtered image. The ICDAR 2003 dataset plus other images were used for testing the system. The results showcased highly satisfactory detection rates but less than satisfactory false positive rates; an issue that is highly common in this field.

In [3], Angadi and Kodabagi adopted a standard preprocessing-detection-postprocessing methodology to detect text from natural scenes. The pre-processing step involved the use of the Discrete Cosine Transform (DCT) to suppress background objects such as trees, vehicles, etc. The DCT is applied with a high-pass filter on 8×8 blocks. The resulting image is then used for the detection phase, which contains two steps: feature extraction and classification. The detection process starts by dividing the given image to 50×50 blocks, where 8 features will be extracted from each block. The features are *contrast* and *homogeneity* on four orientations, 0^o, 45^o, 90^o and 135^o. Two empirically-obtained thresholds T_1 and T_2, one for contrast and one for homogeneity, are then used to determine whether a block is a text region or not. The post-processing step then follows to add refinements to the detected regions. The refinements aim to detect undetected text regions. This is performed by computing the average contrast for blocks adjacent to a detected block. Another threshold (also empirically determined) is used to determine whether the entire adjacent block should be merged, or only specific columns and rows should be merged, or whether the block should not be merged at all. The system was tested on 100 low-resolutions images and high accuracies were obtained. No dataset was mentioned, however, and the results could be different if a bigger set of images was used.

In [41], Kim et al proposed a texture-based approach to detecting texts from images. The texture features of choice were simply the grey levels of the raw pixels of the image; a straightforward and cheap approach since no extra features need to be computed, but comes at the cost of large feature vectors that will be fed to the classifier, a Support Vector Machine (SVM). The authors decided to use SVM as the classifier for two reasons: SVMs work well in high-dimensional spaces, and they are able to incorporate feature extractors to aid classification by using their inputs as features. To train the classifier, the authors opted for the approach proposed in [93] where the non-text patterns are added through the positives outputs resulting from an SVM trained on non-text images. The results of the classification stage are not sufficient to locate the text regions, and therefore two extra steps, *Chip generation* and *Chip fusion*, are added. The former uses a continuously adaptive mean shift algorithm (CAMSHIFT) to locate the text locations. The chip fusion step then attempts to fix any overlapping of regions that could occur as a result of text detection in different resolutions (to detect texts with various sizes). The results produced were in the desirable range, particularly in the processing time. However, the approach is not suited for small texts which is the primary target in the automotive context. Moreover, while the images used were fairly complex, the images found in the automotive context exhibit far greater complexity as obstacles of many kinds and colors will be present.

In [86], Hanif and Prevost designed a wearable system for the visually impaired to detect text from natural scenes. For the texture analysis step, they opted for the statistical GLCM method where they selected 6 out of the 14 features defined by Haralick [36]. These features were divided into three groups: the *contrast* group (contrast, dissimilarity, homogeneity), the *orderliness group* (energy and entropy) and the *correlation* feature was left on its own. For the classification step, two classes of classifiers were tested: generative classifiers and discriminative classifiers. For the former class, the authors used a combination of mono- and multi-Gaussian classifiers for the two classes *text* and *non-text*. As for the discriminative class, three values for the number of hidden units (N_h) were used (20, 40 and 60) to get three discriminative classifiers. As many authors did before, Hanif and Prevost used the ICDAR 2003 dataset for training and testing. They selected 100 images for testing with various size font, shapes and colors. The results obtained were promising for the number of images used, which is small to draw any concrete conclusion on the effectiveness of this approach in this application.

Minetto et al proposed a new gradient-based descriptor for detecting single line text regions in [68]. The proposed descriptor is a variation of the original HOG descriptor [25] and is specifically designed to detect text lines (specifically Roman letters) based on texture features, hence the name *T-HOG*. The authors tested T-HOG in three applications in text detection: as a post-filter, as a detector and as a post-filter in OCR algorithms. As a post-filter, T-HOG and R-HOG (original HOG) were added to a text detector called "SnooperText" that was proposed in [67] by Minetto et al. The SnooperText+T-HOG combination was compared to that of Tian et al [99], Ephstein et al [30] (for their high reported f-scores) and several others detectors from the ICDAR 2003 and 2005 competitions. The results showed that the SnooperText+T-HOG combination, with the right

parameters, produces at least equally good results to that in [99] and [30]. In addition to this, the results prove that this combination clearly outperforms SnooperText with no post-filter. As a text detector, T-HOG can be used in tandem with an SVM using a sliding window approach. However, the authors neglected to evaluate its results since it proved to be highly expensive, despite the ability of T-HOG to detect text. As a post-filter for OCR algorithms, T-HOG works on removing false positives from a given image in order to improve the results given by the OCR.

The "SnooperText" tool mentioned in the previous work is explained in more details in [69]. SnooperText is a texture-based tool that is specifically designed to detect text in urban areas. The tool first segments the given image to get candidate characters by using the *toggle mapping* segmentation scheme mentioned in [84], which is a modified version of the scheme in [31]. The candidate character are then filtered with size and aspect ration constraints first, and a shape descriptor second. The shape descriptor used is based on Fourier moments, pseudo-Zernlike moments and polar descriptor, all of which are fed into an intermediate SVM classifier which is then fed to a final SVM classifier. The extracted and filtered characters are then grouped into text lines and/or words based on geometric criteria mentioned in [80]. From these extracted text lines, further filtering is applied in order to eliminate spurious text lines regions that might have resulted from non-character segments. This step is carried out using the aforementioned T-HOG descriptor [68]. A final multi-scale processing step is implemented in order to make up for the poor performance on varying fonts. For testing, the authors used 4 datasets: the iTowns Project's image collection (ITW)[65], a public benchmark from the Google Street View images by Wang et al.(SVT)[3], the benchmark used by Ephstein et al.(EPS) in [30] and a subset of the ICDAR 2005 dataset. The precision rate obtained exceeded the state-of-the-art approaches in the literature with a value of 0.74. Despite SnooperText's clear effectiveness, it still shows signs of vulnerability to small, distorted or cursive fonts.

2.5 Hybrid Approaches

A notable hybrid-based work was done by Liu and Ikenaga in [49]. Their scheme consisted of four steps: Edge detection, Candidate text region generation, text region verification and a final binarization step. The first step (edge detection), seen as a pre-processing step, aims to extract edge information of as many text region as possible while filtering non-text regions. The authors refrained from giving too many details on the edge detector used as they state that it is diffrent from most existing isotropic edge detectors. They, however, mentioned that some of the detector's ideas are described in [46]. The edge detector is applied on median-filtered image, and the resulting edge pixels are sent to the second stage. The second step uses the edge pixels and applies CC analysis to retrieve text contours. Some non-text contours are also obtained, however, and therefore an additional filtering step needs to be added. Three criteria are used this filtering step: the average gradient magnitude of the pixels, the gradient direction variance of the pixels and the number of

[3] http://vision.ucsd.edu/~kai/svt/

pixels. The text verification step (third) attempts to further eliminate more false positives. The Haar wavelet was selected for this task for its orthogonality, symmetrical features and low computation cost [71]. The final binarization stage is used to make the extracted text regions readable by the an OCR. The binarization is preceded by color clustering step so that the binarization produced is of the best quality. Due to the invariance to size, orientation, language and color exhibited by the features selected, the proposed scheme is able to achieve good recall rates (up to 93.5%) and false alarm rates (down to 3.2%) with various languages (English, Japanese, Chinese), character font sizes and orientations.

In [77], Pan et al designed a hybrid system to detect and localize text from natural scenes. As is the case with most systems, Pan et al's system starts with a pre-processing stage. The aim of this stage is to detect the text-containing regions by assigning confidence values and scale maps to each region, which will then be used to segment the image into components. The confidence values are an approximation of the probability of text being contained in a particular region. This is achieved by extracting the HoG features of each region and feeding them to a Waldboost classifier [90]. This process is done on different scales of 1.2 steps. The confidence values and scale maps are then projected back into the original image, which will binarized using Niblack's algorithm [72] and segmented accordingly. The CC stage then follows, where a Conditional Random Field (CRF) [45] with supervised parameters, using an Artificial Neural Network (ANN), is used. The CRF's goal is to classify a given component as being a "text" component or a "non-text" component. The component-specific unary features such as aspect ratio, compactness, normalized width and height, etc., are exploited in addition to the component-component binary features such as shape difference, spatial distance, overlap ratio, etc. Finally, a text grouping (post-processing) stage is added to connect components into text regions and cut off between-line edges. A Minimum Spanning Tree (MST) based on a learned distance metric is used to connect the components into text regions. The authors then consider the edge cutting problem as a learning-based energy minimization problem, where a greedy algorithm is used to determine the optical edge labels. This process is aided by extracting 6 features from each text line: line regression error, cut score, line height, etc. Tests were carried out on the ICDAR 2005 dataset and the resulting F1 score surpassed the one stated in [53]. A text recognition step is yet to be incorporated, however, although the authors did mention that it is one of their goals for their future work.

A combination of Sobel and Canny edge detector with texture features were used to detect text from video in [88] by Shivakumara et al. The main idea behind this work was to use edge and texture features to differentiate between high contrast regions (text) and low contrast regions (background) through the use of edge detectors and thresholds. The classification between low and high contrast regions follows the authors' observation that high contrast videos produce more Sobel components than Canny, and vice versa for low contrast videos. For the high contrast videos, i.e. videos that contain text, a Sobel feature is again applied to produce a four-directional edge map. From this map, statistical features are extracted to capture the texture property of the text. Namely, mean, standard deviation, energy, entropy, inertia and local homogenity were extracted. Additional features such as straightness and cursiveness were used to deal with the ineviTable false positives.

The system was tested on a manually created dataset rather than an ICDAR dataset, and was compared with three works in the literature [48], [98] and [63]. The results showed that the proposed scheme outperformed these 3 works, but did show some failures in detecting some text lines.

An MSER-based approach in a hybrid system was proposed in [87] by Shi et al. Two types of MSERs are extracted in the first step, dark-on-light MSERs and light-on-dark MSERs. The following step constitutes the bulk of the system, where each MSER component will be labeled as being a text or a non-text component. To perform this task, an undirected graph is constructed with the extracted MSERs as its nodes. The authors then define a cost function and treat the labeling problem as a segmentation problem. Their goal is to minimize the cost function by using textual and geometric features. The textual features were used to create a *unary* cost function, with features such as regularity, uniform stroke width, gradient features, etc. The geometric features were used to measure the distances between the neighboring nodes in order to create a *pairwise* cost function. The pairwise cost function is beneficial in that it complements the unary cost function by exploiting information not used in the unary cost function. After the main cost function is constructed, the max-flow/min-cut algorithm [22] is used for the minimization task for its high speed and satisfactory performance. To further reduce any possible false positive, HOG features from each text-labeled component are extracted and fed into a Random Forest classifier [44]. The ICDAR 2011 dataset was used for evaluating the system. The precision rate in particular was very satisfactory. The recall rate, however, was not of the quality the authors hoped for due to the illumination disturbances found in some images which leads to the incorrect removal of some MSERs.

A hybrid approach for text detection for Farsi/Arabic characters was proposed in [70] by Moradi and Mozaffari. It is labeled as a hybrid approach because it combines a contour-based step, edge detection, to the texture classification process. They used the Sobel operator, with the help of filtering, to extract edges of horizontal, vertical, diagonal and anti-diagonal orientations. To facilitate the upcoming texture classification step, an *artifical corner extraction and refinement* (separated to two steps) step was incorporated to increase the number of corners from four to six. Afterwards, an intensity picture is created using the Discrete Cosine Transform (DCT) with coefficients taken from [24]. The equation in [52] is used to compute the quadratic weights, which will be used with the coefficients to compute the texture intensity of each block. For the texture segmentation step, the authors opted to use LBP for its discriminative abilities. However, some issues arise when using a standard LBP for this application (large feature vector, for example) and this goaded the authors to adopt a variation of LBP. They used a *horizontally emphasized LBP (HELBP)* to extract horizontal text strokes, and a *vertically emphasized LBP (VELBP)* to extract vertical text strokes. The use of a diagonal (and anti-diagonal) emphasized LBP was deemed unnecessary since the number of horizontal and vertical strokes found in Farsi/Arabic texts exceeds the number of strokes found in the diagonal directions. The feature vectors produced are the fed to an SVM with a Radial Basis function (RBF) kernel. As a final step, horizontal and vertical profile analyses were done to locate the text bounding boxes. This was done by computing the Mean Texture Intensity (MTI) for

each region. Since there is no available dataset for such an application, the authors had to create their own dataset using 50 (720×576) video clips of various contrasts, brightness, lighting, etc., from various sources. The results obtained were reasonably high (91.38% recall, 87.22% precision) and, with the right parameter adjustments, can achieve high recall and precision rates on English and Chinese texts.

2.6 Summary

This Section aims to provide a summary of all the works mentioned thus far in a tabular format. The summary of each paper will based on the following:

- **Approach Class**- Whether the work's main idea follows a Contour-based approach, a Texture-based approach or both (Hybrid)
- **Pre-processing**- What algorithm/features, if any, were used for this step
- **Detection**- What algorithm/features were used for this step
- **Recognition**- What algorithm/features, if any, were used for this step
- **Post-processing**- What algorithm/features, if any, were used for this step
- **Dataset(s)**- What dataset(s) were used for testing the proposed system
- **Comments**- Brief strengths and weaknesses of each work with relation to the Automotive context in particular.

After the Table, the algorithms/features that seem most suiTable for the Automotive context will be listed with brief explanations to why they were selected. As is the case with any problem, *accuracy* will be one of the criteria for selecting the appropriate approach. However, *delay* is another equally important criterion that needs to be taken into consideration since the focus of this research is in the automotive context; having real-time performance with minimal delay is a requirement.

Table 2.1: Summary of previous works

Paper	**Approach Class**	**Pre-processing**	**Detection**	**Recognition**	**Post-processing**	**Comments**
[41] 2003	Texture	None	Grey-level values+SVM	None	Chip generation (CAMSHIFT) & Chip fusion	•No recognition step •Not suited for small texts
[76] 2004	Contour	Top-hat processing & Otsu's binarization	Geometrical & Size constraints	None	None	•No details for the recognition step • Small-large character classification unnecessary for AC
[49] 2006	Hybrid	Edge detection (detector details not given)	CC analysis: *Gradient & geometrical features*	None	Texture analysis: *Haar Transform* & Binarization	•No recognition step • Not real-time • Effective combination of CC and texture analysis
[86] 2007	Texture	None	CRF with two NRGs & an OCR Module	None	None	• Testing set too small to draw any concrete conclusions •No text recognition step
[50] 2008	Contour	Niblack [72]	Geometric & intensity guidelines used in the DAR community [33]	None	Intensity Filter & Shape filter	• Images taken from still camera shots, i.e. stable and not blurred •No text recognition step •Not real-time
[3] 2010	Texture	Discrete Cosine Transnform (DCT)	Contrast & Homogeneity features	None	Average contrast for undetected adjacent blocks	•Unconvincing dataset • Post-processing step adds robustness •No recognition step •Not real-time

[67] 2010	Hybrid	Sobel & Canny edge detectors	Texture Features: *Mean, standard deviation, entropy, inertia and homogeneity*	None	Additional features: straightness & cursiveness	•Convincing dataset •Separates between low and high contrast videos •No details regarding execution times • No recognition step
[62] 2011	Texture	Wavelet transform & Gabor filter	Wavelet Entropy	None	None	•Simple & straightforward •Good detection rate •Less than satisfactory false detection rate •No recognition step •Not tested for real-time
[77] 2011	Hybrid	HOG+Waldboost classifier & Niblack [72]	CRF with supervised parameters, using an ANN	None	MST & Greedy approach for an energy minimization problem	•Combines a CC approach (CRF) with texture features (HOG) • Multilingual support •No recognition step (though mentioned for future work)
[100] 2011	Contour	Niblack [72]	Size & Aspect ratio constraints	CRF with two NRGs & an OCR Module	Intensity filter	•Double iterations could be time consuming •Unconvincing f-score •Lack of details on the OCR module used
[68] 2012	Texture	None	T-HOG (not recommended)	Tesseract	T-HOG	•Excellent as a post-detection or as a post-OCR filter (post-processing) •Inefficient as a detector •Not real-time

[87] 2012	Hybrid	None	MSER extraction & max-flow/min-cut algorithm [22]	None	HOG+Forest	•MSER extraction computationally inexpensive •Effective removal of false positives •Subpar ecall rate •No recognition step •Not tested for real-time
[35] 2012	Contour	Greyscale, Color normalization & Size features	MSER regions extraction	HOG+SVM	None	•MSER extraction computationally inexpensive •Real-time •Tailored for text-less traffic signs
[70] 2013	Hybrid	Sobel edge detector & artificial corner extraction and refinement	DCT & LBP(*HELBP & VELBP*)+SVM	None	MTI	•Multilingual support •Use of HELBP & VELBP improves detection •Use of video clips •No recognition step
[69] 2013	Texture	Image segmentation through *toggle mapping*[31]	Fourier moments, pseudo-Zernlike moments & polar encoding + SVM	None	Geometric criteria & T-HOG	•T-HOG specialized for characters •Can be used with an OCR •Still vulnerable to certain fonts •Not tested for real-time
[42] 2013	Contour	None	MSER+Adaboost & a Multilayer Perceptron	None	Size normalization	•MSER extraction computationally inexpensive •No recognition step •Not real-time

Two important aspects are commonly lacked in the works mentioned, aspects that are essential to designing a robust text recognition system for the Automotive context. The first is the lack of a clear *Recognition* step, where an input image (whether a full image or an ROI) is given and the appropriate text is accurately recognized and displayed. Among the mentioned works, [100] is the only work that attempts to incorporate a solid *text* recognizer. [68] does include an OCR but it was only added to test the proposed algorithm (T-HOG) with an OCR tool, i.e. the OCR is not part of the system. [35] uses an object recognizer to recognize certain kinds of traffic signs, not a *text* recognizer. The second is the lack of any real-time testing. While [70] used a decent number of video clips for testing, the videos were taken from time-insensitive clips. [35] is the only paper that included real-time performance and testing part of its objectives, and this was shown by the use of manually-taken video clips from a vehicle.

Two ideas taken from the previous works can be of great use in our objective. The first is the contour-based blob detection algorithm **MSER**. As mentioned in the Table, MSER is known for its low computational cost; a feature that is of the utmost importance when dealing with real-time scenarios. The second is the OCR tool mentioned in [68] "Tesseract" [89]; a Google-sponsored open-source OCR tool that is renowned for its highly accurate results. One of its strongest advantages is that it has already been trained numerous times in the past by several developers and therefore the need to train a new text recognizer is eliminated.

2.7 Datasets

Several datasets exist in the field of text detection; some are designed for document analysis, some for natural scenes images and some designed on a subset of either (the GTSRB and GTSDB datasets, for example, for traffic signs[4]). Most previous works (as seen in the previous sections), however, use the datasets provided in the *ICDAR 2003*, *ICDAR 2005* and/or *ICDAR 2011* competitions.

1. **ICDAR 2003 [91]** [5] **& ICDAR 2005 [53]** [6]
   These datasets contain images for four fairly similar yet different applications:
   - *Robust Reading* - reading complete words from camera-captured natural scenes
   - *Robust Word Recognition* - reading single words from natural scenes
   - *Robust Character Recognition* - reading single characters from natural scenes
   - *Text Location* - locating text found in natural scenes

   Each dataset is divided into three parts:

[4] http://benchmark.ini.rub.de/?section=home&subsection=newss
[5] http://algoval.essex.ac.uk/icdar/Datasets.html
[6] http://algoval.essex.ac.uk:8080/icdar2005/index.jsp?page=intro.html

- *Sample* - provided to allow users to develop early impressions on the functionality of their system
- *Trial* - further divided into *TrialTrain* to train one's system, and *TrailTest* to produce publishable results. However, a system should be trained on the entire set for competitions.
- *Competition* - used to produce results for ICDAR competitions that will be used to evaluate the performance of a system.

Some images of this dataset are shown in Figures 2.10, 2.11 and 2.12.

Figure 2.10: Text Location data set, 'Sample' class

Figure 2.11: Text Location data set, 'TrialTrain' class

Figure 2.12: Text Location data set, 'TrialTest' class

2. **ICDAR 2011** [85][40] [7]
   Six years after its predecessor competition, ICDAR 2011 returned to test systems for text detection from both natural scenes and documents. This competition, however,

[7] http://robustreading.opendfki.de/

consists of three (rather than 4) applications: Text Localization (location), Text Segmentation and Word Recognition, each of which has a dataset that is divided into *Train* and *Test* classes. Another difference this competition has is the availability of both camera-captured images and *digitally-born* images, i.e. digitally created web and e-mail images. Digitally-born images differ from camera-captured images in that they are subject to information loss from compression algorithms when transmitted online. Figures 2.13 and 2.14 show some images found in the text localization dataset (some images are found in the 2003/2005 text location datasets).

Figure 2.13: Text Localization data set, 'Train' class

Figure 2.14: Text Localization data set, 'Test' class

Chapter 3

Proposed Architecture

3.1 Introduction

As outlined in Section 2, a subtle switch has been developing in the way that text detection has been getting tackled over the past years; a switch from a Connected Component Analysis (CCA) approach or Texture Analysis (TA) approaches to hybrid approaches. This switch is justified by the end results of the hybrid works as the overall precision, recall, f-scores and (sometimes) delay have been experiencing notable improvements. In addition, hybrid approaches provide the researcher with more flexibility in both the way he tackles the problem at hand and in the different kinds of objects (text in this case) he would be able to detect and, possibly, recognize.

The performance improvements resulting from hybrid approaches can be attributed to the complimentary nature that is exhibited between CCA approaches and TA approaches. CCA approaches are reputed by their ability to carry out the task at hand at desirable speeds and notorious for their relatively lacking accuracies. TA approaches, on the other hand, are reputed by their relatively high accuracies and notorious for their less than desirable execution times. Therefore, by combining the two approaches, it is possible to have a system that is able to achieve desirable accuracies within reasonable execution times. The challenge, however, is deciding when and how to use each approach in such a way that neither accuracy nor speed is compromised.

3.2 Proposed Approach

Given the advantages that accompany a hybrid approach, we also decided to adopt such an approach. A flowchart of our approach is shown in Figure 3.1.

The first step of our architecture is Connected Component Analysis (CCA). The purpose for using this step is to provide the next step, Texture Analysis, with a significantly

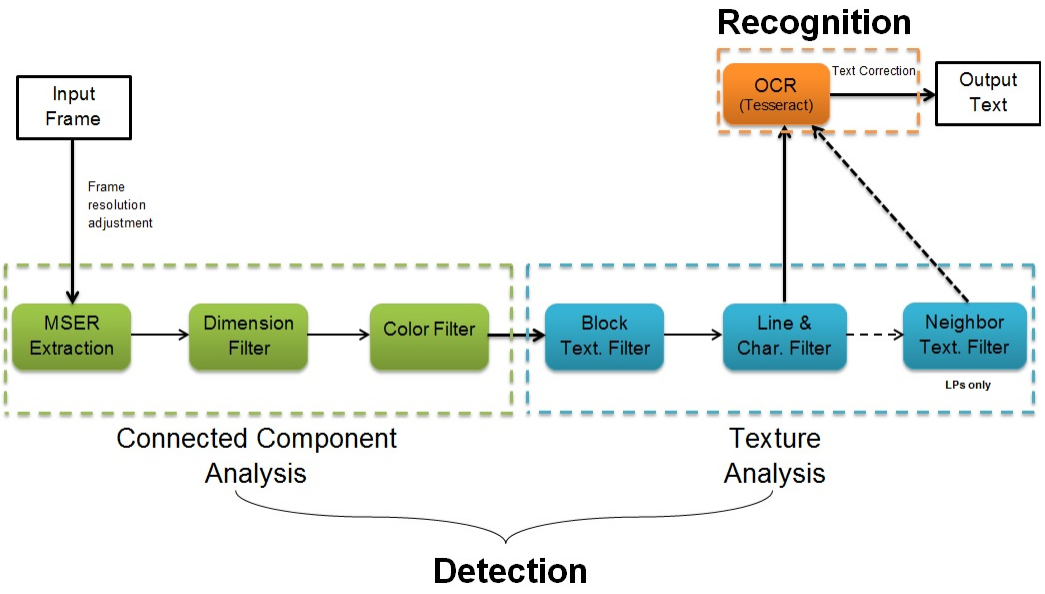


Figure 3.1: Flowchart of the proposed architecture

reduced set of objects. The following step uses novel manually-designed texture filters that are able to separate text-rich objects from nontext-rich objects. The final step, the recognition module, utilizes a highly regarded Optical Character Recognizer (OCR) to read the texts written on the objects passed on by the previous step. (The frame resolution adjustment is discussed in Section 5.3.2). At the end of this chapter, Figures 3.22 and 3.23 summarize the detection module by displaying a sample frame after each CCA and TA step is applied.

3.3 Connected Component Analysis (CCA)

As highlighted in the Section 2, maintaining reasonable execution times is of paramount importance in the automotive context. As CCA is reputed for its low execution times as opposed to TA approaches, it behooves us to include it as our first step. The following sections outline the various steps used. Section 5.2 provides numerical details that justify the use of each step. A flowchart for this step is shown in Figure 3.2.

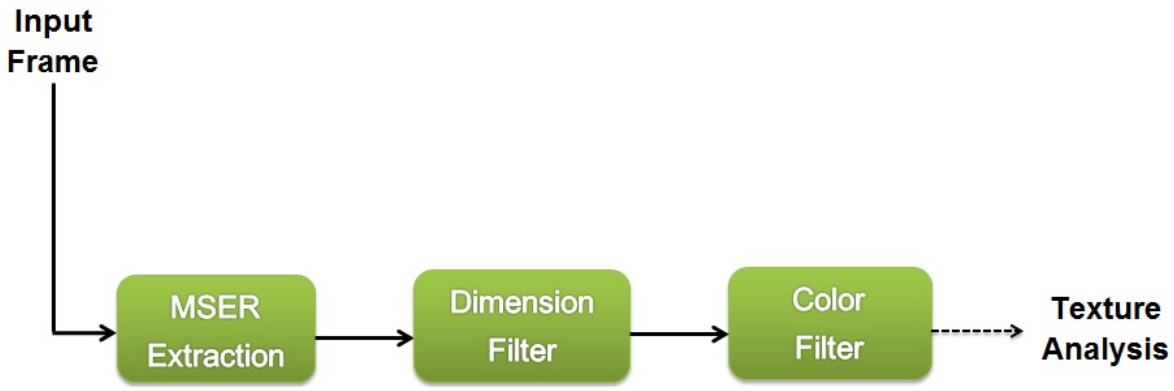


Figure 3.2: Flowchart of the CCA step

3.3.1 Maximally Stable Extremal Regions (MSERs)

Background and Motivation

Proposed by Matas et al. in 2002 [66], the Maximally Stable Extremal Region (MSER) is a blob detection approach that aims to find correspondances between two different views of an image. The process of extracting the extremal regions is achieved by thresholding all sorted pixel in a given image using various thresholds. The list of connected components is maintained by means of a union-find algorithm. A 'maximally stable' region is identified when a region, or component, remains unchanged in size through a wide range of thresholds.

The popularity that MSER has been gaining is owed to its performance speed. The first step, sorting, has a complexity of $O(n)$ and the second step, the union-find, has a compelxity of $O(log(logn))$ thus resulting in a nearly linear worst-case performance of $O(nlog(logn))$. Moreover, MSER exhibits the following properties:

- Invariance to affine transformation of image intensities
- Mult-scale detection
- Since only the regions that remain relatively unchanged during the thresholding process, Stability is achieved

However, MSER is not known for producing accurate results on its own and therefore we are required to add a subsequent step to filter out unwanted objects. Figure 3.3 provides an illustration of a sample MSER output.

3.3.2 Heuristic Approach

To filter out unnecessary objects from the set of MSERs, we decided to adopt a Heuristic approach. This choice is based on two reasons:

Figure 3.3: Left: original frame, Right: same frame with the MSERs highlighted

- Most previous works incorporated a classifier for this step for its proven effectiveness. This, however, is not an option for our work since there currently is no publicly available well-representative dataset for our text-rich objects of interest. Although a few license plate datasets are available online, they are either small in size or come from regions where the license plates look noticeably different from NA plates. Therefore, the task of effectively training a classifier is rendered not possible.

- Heuristics help provide a solution to a given problem through simple means at the expense of sub-optimal performances, and have been used in different applications, such as [12], [15], [5], [17] and [9]. Our objects of interest exhibit dimension and color features that allows for a heuristic approach to be effective yet fast, as this section explains. Additionally, the next step in our framework, the TA step, will impose more severe conditions on the remaining set of ROIs. In other words, optimal results are not a necessity at this step. Therefore, with the issue of optimal performance at this step out of the equation, a heuristic approach is well-suited for our framework.

Dimension Heuristics

The MSERs obtained come in various dimensions. Our OOIs also come in different dimensions depending on the distance from the camera to the object, thus giving us the freedom to control the range from which objects are detected. That is, by restricting the dimensions to small numbers, our system will be able to filter out all close-range and most mid-range objects and keep the desired faraway objects, and the same can be done for the other two modes, as shown in Figure 3.6. The option to detect objects in all distances at once is possible, but not recommended for the following reasons:

- The number of filtered out objects would get reduced significantly, i.e. more false positives would be present, thus rendering the task for the subsequent steps harder. This is shown in Figure 3.7.

- Unlike most previous works that are concerned with detection only, our work aims to recognize the text that is written on the OOIs. To do so, we need to ensure that the detected object is clear enough for the recognition module to carry out its task effectively. Since the text contained in far-away objects is nearly impossible to read

(even for humans), the choice to exclude faraway objects, i.e. disable long-range mode, is justified. A sample far-away license plate is displayed in the bottom-right image in Figure 3.5.

The necessary dimensions were determined by manually cropping the OOIs from images and recording their dimensions, as shown in Figures 3.4 and 3.5. The constraints we found applicable for our application are the *height*, *width*, *area* and *aspect ratio*. More numerical details on how they are obtained are provided in Section 5.2.1.

Figure 3.4: An example of how we determined the dimension constraints for traffic boards (sizes are in pixels from a 900×600 frame)

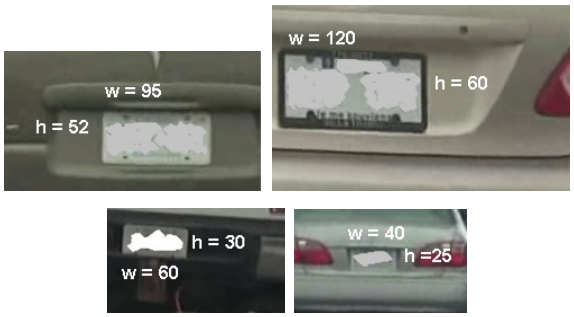


Figure 3.5: An example of how we determined the dimension constraints for license plates (sizes are in pixels from a 900×600 frame)

(a) A frame in which only faraway objects are detected

(b) A frame in which only mid-ranged objects are detected

(c) A frame in which only close-ranged objects are detected

Figure 3.6: Selecting different dimension constraints leads to detecting objects located at different distances from the vehicle

Figure 3.7: Attempting to detect objects from all ranges could result in false positives

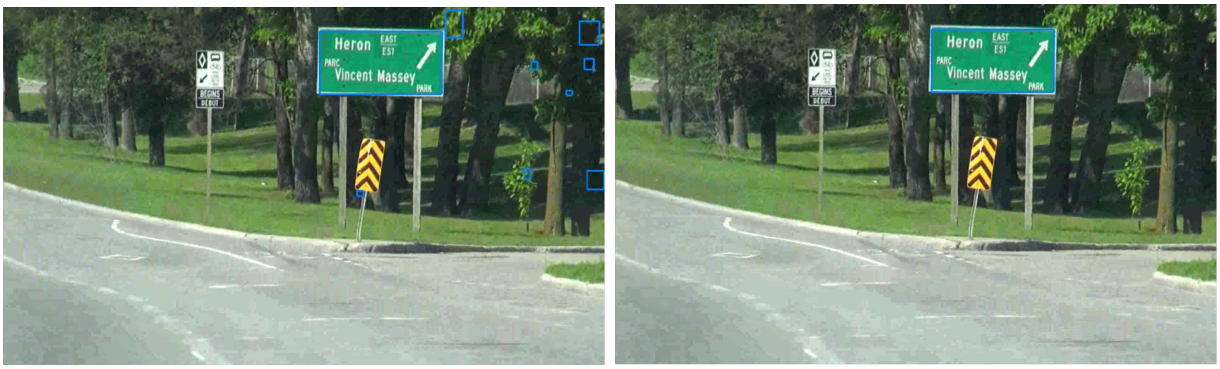


Figure 3.8: Left: a sample frame before the dimension filter step, Right: the same frame after the dimension filtering step.

Color Heuristics

Our OOIs exhibit color features that are unique enough to allow for two color features to be utilized. The first feature is the ratio of white pixels to non-white pixels in a given object. As done with the dimension heuristics, we manually extracted many OOIs from images and recorded the ratios. More numerical details are provided in Section 5.2.1. An illustration of this feature is shown in Figure 3.9.

The second feature is concerned with the presence and/or absence of specific colors in an object. Through observation we are able to deduce that the vast majority of TBs are green-coated, with the exception of the text (written in white). The few that are not green-coated are blue-coated. This will help us in removing a considerable number of unwanted objects from the MSERs obtained earlier. Figure 3.10 shows an example of this feature on a TB. To produce the image on the right, the frame is first *green-normalized*. This is achieved by setting each pixel to $G/(R+G+B)$, where G, R and B are the green, red and blue values of each pixel, respectively. By setting each pixel to this value, the red and blue features of a

Figure 3.9: Through observation, we were able to deduce that our OOIs exhibit a fixed range of white to non-white pixel ratios

given frame diminish while the green feature is maintained. By thresholding the resultant green-normalized image, the desired image is produced. For blue-coated boards, the same process is carried out with the green value in the numerator replaced by the blue value. Figures 3.11 and 3.12 provide an illustration of the effect the color filter has on a sample frame. As for the LPs, we observed that the vast majority of NA plates are white-coated, thus enabling us to discard any color-coated object. More numerical details are provided in Section 5.2.1.

Figure 3.10: Left: original frame, Mid: green-normalized frame, Right: thresholded frame

Figure 3.11: Left: sample frame before applying the color filter, Right: same frame after applying the color filter. The false positive present in the first frame is removed by the color filter due to the absence of green pixels.

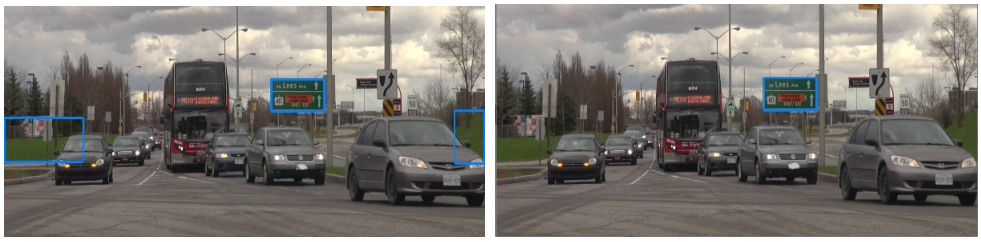


Figure 3.12: Left: sample frame before applying the color filter, Right: same frame after applying the color filter. The color filter is also able to remove false positives that contain green pixels if they do not satisfy a color threshold. Details on this threshold are provided in Section 5.2.1.

3.4 Texture Analysis (TA)

Despite the reduced set of objects, more false positives need to be filtered. At this step, we are looking to impose stricter filters in order to remove the more stubborn false positives. To achieve this, we deemed it necessary to incorporate a texture analysis step for to its reputation in providing greater robustness compared to its CCA counterpart. Figure 3.21 illustrates an example of the effect this step has on a sample frame. A flowchart for this step is shown in Figure 3.13.

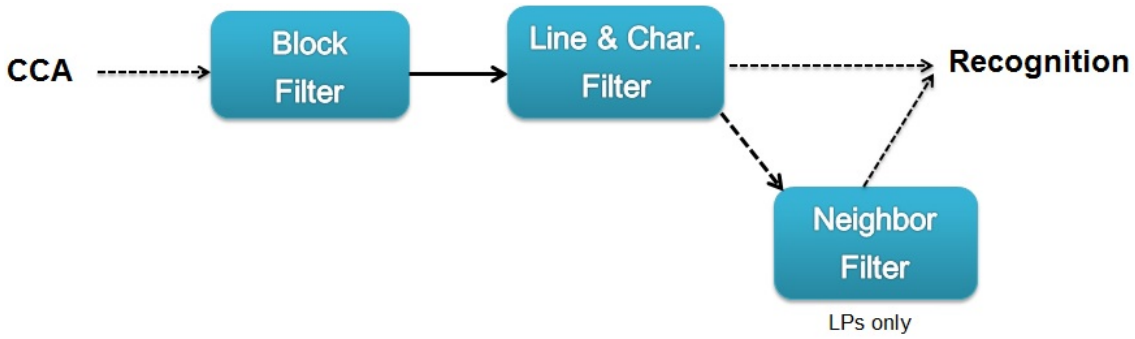


Figure 3.13: Flowchart of the TA step

3.4.1 Novel Texture Analysis Approach

Since the automotive context is our target, it is essential to use a TA approach that achieves desirable results within reasonable execution times. However, most previously used TA approaches (mentioned in Section 2.1.2) suffer from delays that are not suitable for the automotive context. Therefore, we are faced with the sole option of implementing our own texture filters. The following subsections provide the logical details of our texture filters that, as the numerical figures provided in Section 5.2.2 show, carry out the intended task with desirable accuracies at very low execution times.

3.4.2 Block-based Texture Analysis

The text displayed on our OOIs adds a layer of texture that can be exploited in order to distinguish the OOIs from other objects. The challenge, however, is how to exploit this texture. The first step in doing so is the application of an edge detector, a canny edge detector in this case. As Figure 3.14 shows, the edges present in a sample OOI tend to be adjacent to each other, i.e. some degree of compactness is exhibited. To be able to measure this compactness, we decided that a block-based approach would suit our goals best. By dividing each image into 4×4 cells, counting the edges found in each cell and comparing it to a threshold, we would be able to filter out a considerable number of unwanted objects. Figure 3.15 illustrates this process. A brief pseudocode alongside numerical details on how the threshold was obtained and on the effectiveness of this filter are provided in Section

5.2.2.

Figure 3.14: Top: A traffic board and its canny-edge equivalent, Bottom: An unwanted object and its canny-edge equivalent

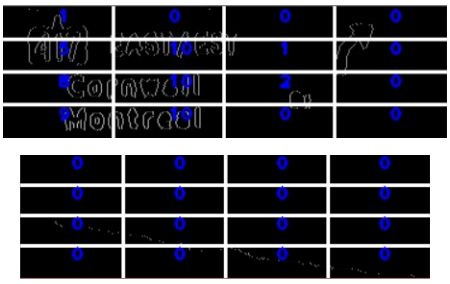


Figure 3.15: Each image is divided into 4x4 cells. The number of edges contained in each cell are counted. These counts will be used to determine whether the given image passes or not. Top: Traffic board, Bottom: Unwanted object

3.4.3 Character and Line-based Texture Analysis

Traffic Boards

We are able to observe that TBs contain a fixed range of lines that come in a fixed range of sizes, as shown in Figure 3.16. Therefore, by limiting the range of lines and their dimensions, more unwanted objects are discarded.

Furthermore, in order to further improve the potential results of the subsequent recognition step, we opted to add an extra substep that separates highway numbers from other texts within a traffic board. This is beneficial to our work in two ways:

- The system would be able to differentiate between highway numbers and other possible numbers on a traffic board, such as distance (e.g. 700m)
- The recognition step benefits from the addition of this substep in terms of the overall recognition rates, since the OCR of choice works best when given smaller and simpler input.

The separation is based on the observation that highway numbers are printed on a differently-colored background (white) than the rest of the characters (green, and sometimes yellow). This observation is demonstrated in Figure 3.17.

Figure 3.16: Left: lines found on a traffic board, Right: lines found on an unwanted object

Figure 3.17: Highway numbers (417 in this frame) are printed on white backgrounds, whereas the rest of the text is printed on green backgrounds

License Plates

Through observation, we found that the following two texture feautres are suitable for detecting license plates:

- The edges found in a false positive come in many different shapes and sizes, unlike the characters in a license plate that appear in fixed shapes sizes, as shown in 3.19. Moreover, the range of the number of characters found in a license plate is very limitied, i.e. a license plate usually has a minimum of 6 characters and a maximum of 8. Therefore, by setting dimension restrictions on the edges found in a license plate and limiting the number of edges, numerous unwanted objects are discarded.
- Unlike with TBs, LPs have only one text line that is of interest. In some occasions, however, the canny edge detector produces several, overlapping text lines as a result of noisy edges within the image. By eliminating the overlapping text lines and keeping the larger text line, we are able to pinpoint the exact location of the text line within the LP. Additionally, by limiting the size of the text line to the average width and height of a standard NALP, more unwanted objects as well as incomplete LPs are eliminated. Examples are shown in Figures 3.18.

Figure 3.18: Left: LP with overlapping text lines, Middle: LP without overlapping lines, Right: Incomplete license plate

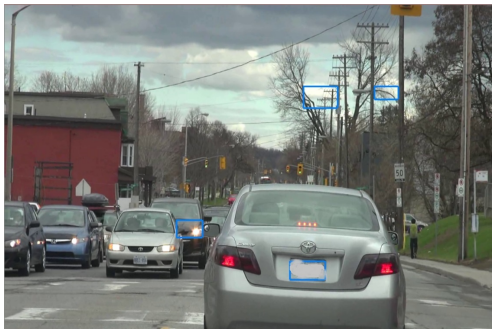
Figure 3.19: Left: edges found in a license plate, Right: edges found in a unwanted object

3.4.4 Neighbor Texture Analysis

License plates have an extra feature that can bolster our approach. As shown in Figure 3.20, most (if not all) LPs are surrounded by texture-light areas, i.e. an edge detector would produce little to no edges. Therefore, by applying the aforementioned block-based filter over the neighboring areas of an LP, we would be able to remove the most stubborn of false positives.

Figure 3.20: License plates are typically surrounded by edge-light areas

(a) Frame before the block-based texture filter

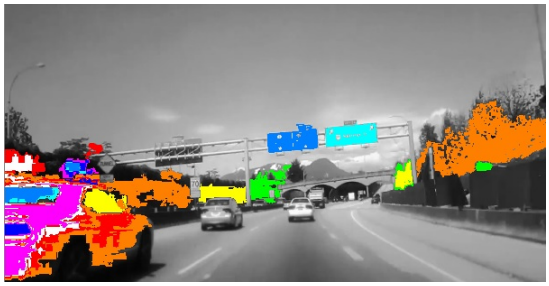


(b) Frame after the block-based texture filter

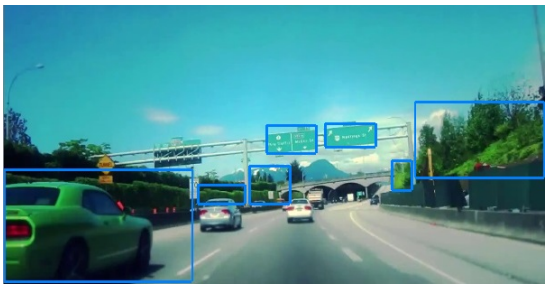
(c) Frame after the character and Line-based texture filter

(d) Frame after the neighbor-based texture filter

Figure 3.21: An example of the effect the Texture Analysis step has on a sample frame

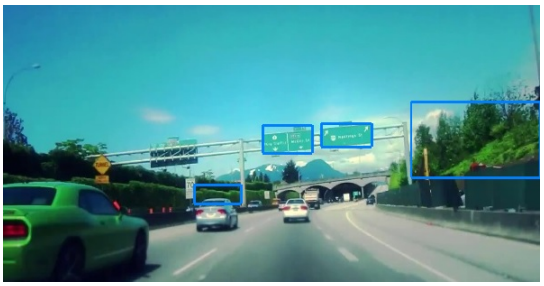


(a) MSER extraction step

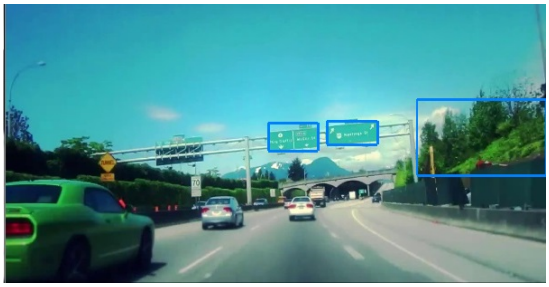


(b) Non-overlapping regions are displayed - 5 FPs

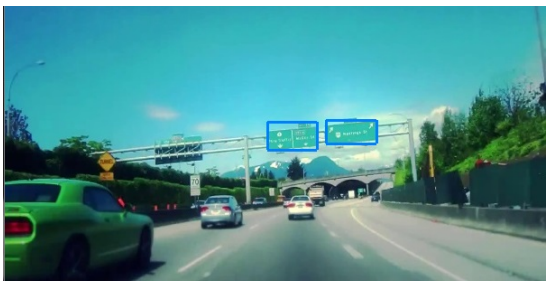
(c) Frame after applying the dimension filter - 3 FPs



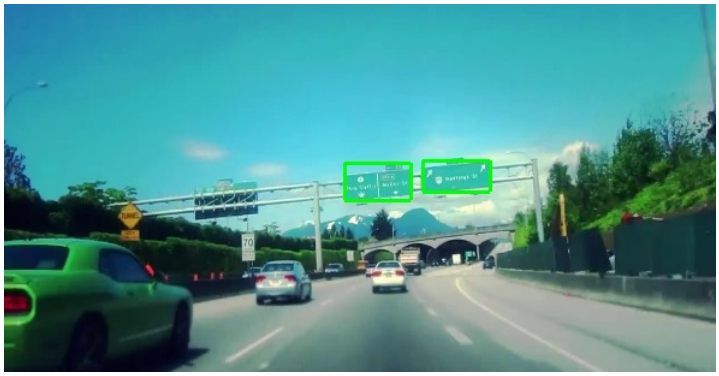
(d) Frame after applying the color filter - 2 FPs



(e) Frame after the block-based texture filter - 1 FP

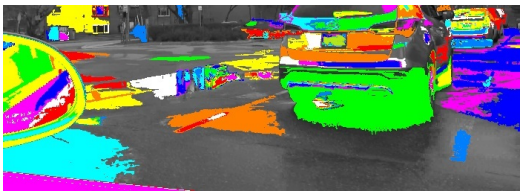


(f) Frame after the character and line-based texture filter - 0 FPs

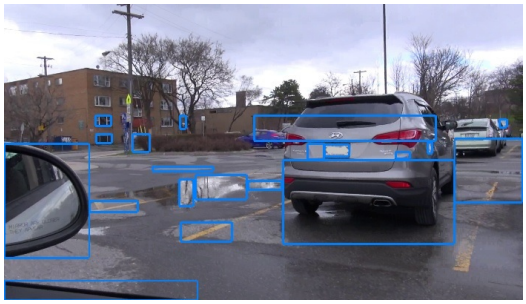


(g) Frame after all filters have been applied: all FPs removed

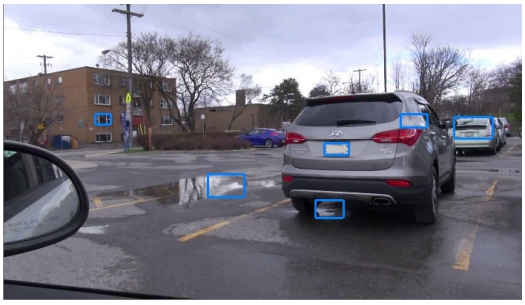
Figure 3.22: A sample frame going through each CCA and TA step for detecting a traffic board (FPs = False Positives)



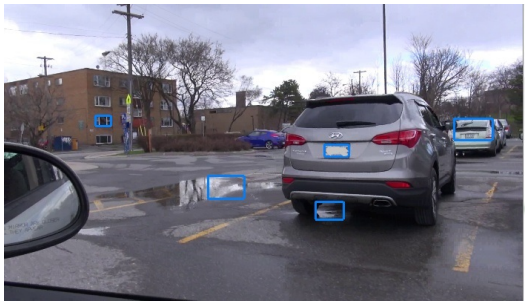
(a) MSER extraction step



(b) Non-overlapping regions are displayed - 19 FPs



(c) Frame after applying the dimension filter - 5 FPs



(d) [Frame after applying the color filter - 4 FPs

(e) Frame after the block-based texture filter - 2 FPs

(f) Frame after the character and line-based texture filter - 1 FP

(g) Frame after the neighbor-based texture filter - 0 FPs

(h) Frame after all filters have been applied: all FPs removed

Figure 3.23: A sample frame going through each CCA and TA step for detecting a license plate (FPs = False Positives)

Chapter 4

Recognition

Reading the text of the remaining ROIs is the final step of our framework. An Optical Character Recognizer (OCR) is required to achieve this, as the following sections provide details on the selection of the OCR and its integration into our framework. A flowchart for this step is shown in Figure 4.1.

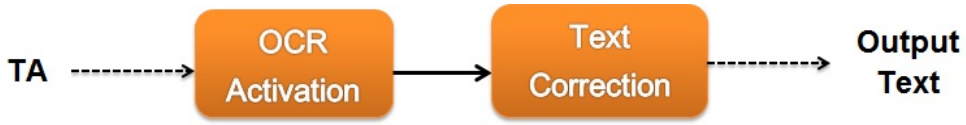


Figure 4.1: Flowchart of the recognition step

4.1 Selection of Optical Character Recognizer (OCR)

The choice of the OCR is based on Section 2. Some of the few previous text recognition-based works used an OCR called "Tesseract". Tesseract was initially developed by Hewelett-Packard (HP) in 1980s. It was used heavily in the 1990s as it was one of the most accurate OCR tools. After that, however, its popularity diminished drastically. After 2005, tesseract became an open-source tool sponsored by Google [89]. Since then, tesseract's popularity started witnessing a second rise and has undergone various improvements that make it the most highly regarded open source OCR to date. Therefore, we decided to use Tesseract as the OCR for our work.

The text recognition process of tesseract starts by performing CCA that aims to extract the outlines of characters and store them as blobs. Afterwards, the blobs are organized into text lines and analyzed based on their character spacing, whether it is fixed or proportional. A two-pass process then takes place. The first pass attempts to recognize each word individually and if a satisfaction threshold is surpassed, the word is fed into an adaptive classifier. In the second pass, the entire page is scanned a second time in an attempt to recognize those words that have not been properly recognized in the first pass. Despite its

effectiveness in text recognition, [89] shows that the computational cost of this process is relatively high.

4.2 Applying the OCR

The lines extracted through the line and character filter, mentioned in Section 3.2, are fed to tesseract. Sample results are displayed in Figure 4.2.

Figure 4.2: Sample results of the recognition step

While the outputs shown in Figure 4.2 look acceptable, achieving such outputs require extra steps. This is because the results produced by the OCR are sometimes inaccurate. This is particularly evident when dealing with very small texts appearing on faraway objects, as shown in Figure 4.3. This negatively impacts the overall added delay as the OCR will be used needlessly in many frames, as well as drop the overall recognition rate. To resolve this issue, we add two extra substeps: text correction and OCR activation control.

Figure 4.3: Application of the OCR on a faraway object often yields an incomprehensible output

4.2.1 Text Correction

Figure 4.4 presents a possible output without using this substep.

Regarded as the post-processing step, this substep ensures that the output displayed, if any, is always meaningful, i.e. transform the output in Figure 4.4 to the one displayed in Figure 4.2. In order to achieve this goal, a 'dictionary' needs to be used. The contents of this dictionary, however, need to be filled in accordance with the geographical location of

Figure 4.4: Sample OCR results without using the Text Correction substep

the vehicle, i.e. if the vehicle is in, for instance, Ottawa then the dictionary should contain locations that the vehicle would realistically go to from within Ottawa, such as Montreal, the University of Ottawa, the Parliment, etc. Otherwise, the dictionary would be too large and therefore infeasible to use.

As such, a meaningless output will be discarded and a semi-meaningful output will be displayed with its corresponding text from the dictionary. Numerical details on what separates a meaningless output from a semi-meaningful output, and the effect this substep has on the recongition rate are presented in Section 5.3.3.

4.2.2 OCR Activation Control

Despite the improved recognition rates that would accompany the text correction substep, the OCR is still being applied in every frame in which an OOI is detected regardless of the quality of the text. As a result, valuable processing time is wasted. Since the OCR is the most time-consuming component of our system and that achieving minimal delay is an objective of this work, an extra substep that ensures the activation of the OCR only when needed is a necessity.

To achieve this task, a measure for the quality of the input text is required. We deemed the height of a given input line to be a sufficient measure. Figure 4.5 illustrates an example of the effect this substep has on a sample input. Numerical details on the required height and the effect this substep has on the overall added time delay are provided in Section 5.3.3.

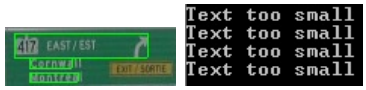


Figure 4.5: Sample OCR result with the OCR Activation substep

Chapter 5

Performance Evaluation

5.1 Data Used for Testing

5.1.1 Testing Criteria

As is the case with all works, *accuracy* (metrics are mentioned in Section 5.3) is a major criterion. However, since the target of our work is the automotive context, *delay* holds a significant importance as well. Therefore, our work will be judged by how accurate the results are and how much time is required to execute each step.

5.1.2 Test Data Categories

In order to have a better picture of our system's performance in the automotive context, we categorized our test data into three speed-based categories. Tables 5.1-5.4 provide details of our video test data. The majority of the videos were manually taken using a high resolution camera of 1920×1080 pixels, while the remaining videos were downloaded from the web and have a medium resolution of 1280×720. The impact of resolution on the effectiveness our system is discussed in Section 5.3.2.

Category	**Range (km/h)**	**# Videos**	**# Boards**	**# LPs**	**AVL**
Highway Speed	100+	27	82	33	24.0 sec
Mid Speed	50 - 100	9	10	23	15.0 sec
Slow Speed	0 - 50	27	11	135	20.8 sec
		63	**103**	**191**	

Table 5.1: The three test data categories (AVL = Averge Video Length)

Video #	**Resolution**	**# of Boards**	**# of LPs**
1	High	1	1
2	High	1	4
3	High	2	5
4	High	3	0
5	High	1	0
6	High	1	2
7	High	2	0
8	High	6	7
9	High	1	2
10	High	1	1
11	High	1	0
12	High	0	3
13	High	1	0
14	High	1	1
15	High	2	1
16	High	8	3
17	High	4	3
18	High	5	-
19	Mid	3	-
20	Mid	7	-
21	Mid	2	-
22	Mid	5	-
23	Mid	3	-
24	Mid	3	-
25	Mid	2	-
26	Mid	6	-
27	Mid	10	-
		82	**33**

Table 5.2: Highway speed (100+ km/h) category

Video #	Resolution	# of Boards	# of LPs
1	High	0	5
2	High	0	9
3	High	1	3
4	High	1	1
5	High	1	1
6	High	1	2
7	High	2	1
8	High	3	1
9	High	1	3
		10	**23**

Table 5.3: Mid speed (50 - 100 km/h) category

5.2 Parameter Experimentation

This Section highlights the process in which we determined the various thresholds, mentioned in the following subsections, of the steps mentioned in Sections 4 and 5. In each step, our goal is to minimize the number of false detections, i.e. false positives, while keeping the number of missed detections, i.e. false negative, at a minimum.

5.2.1 CC Analysis Parameters

Dimension Filters

In order to determine the dimension constraints, we gathered as many images of our OOIs as possible and recorded their dimensions. The images were cropped out of numerous still images and from video frames. Since we aim to minimize the number of false negatives, the maximum and minimum of each dimension should be used as the cutoff point. However, at an attempt to exercise even more caution, we added a small arbitrary offset to each maximum and subtracted a small arbitrary offset from each minimum. The extracted numbers from the cropped images and the final cutoff values are shown in Table 5.5

The texts printed on traffic boards are more readable than those printed on license plates. This is because traffic boards are larger in design and have smoother texts that have enough contrast to allow them to be read from long distances. As such, we enabled all distance modes (refer to Section 3.3.2) for traffic boards and disabled long-range mode for license plates.

Video #	**Resolution**	**# of Boards**	**# of LPs**
1	High	0	3
2	High	0	4
3	High	0	1
4	High	0	9
5	High	0	9
6	High	0	1
7	High	1	0
8	High	0	2
9	High	0	3
10	High	0	6
11	High	0	4
12	High	0	4
13	High	1	0
14	High	0	5
15	High	1	0
16	High	1	0
17	High	1	7
18	High	6	0
19	High	0	11
20	High	0	6
21	High	0	8
22	High	0	6
23	High	0	4
24	High	0	13
25	High	0	8
26	High	0	11
27	High	0	10
		11	**135**

Table 5.4: Slow Speed (0 - 50 km/h) Category

			Cutoff values	
	TBs	**LPs**	**TBs**	**LPs**
Min Width	70	28	50	25
Min Height	23	20	15	15
Min Area	1754	673	1250	650
Min Aspect Ratio	-	1.7	-	1.5
Max Width	202	158	250	180
Max Height	98	110	130	125
Max Area	9802	2765	11000	3000
Max Aspect Ratio	-	2.45	-	2.6

Table 5.5: Dimension information (in pixels) taken from 150 traffic boards and 250 license plates, and the final cutoff values

Color Filters

As done in the dimension filtering step, we gathered many images of our OOIs and recorded the color information that we needed. Table 5.6 displays the extracted numbers and the final cutoff values.

In order to measure the effectiveness of our cutoff values, we computed the percentage of removed false positives, percentage of added false negatives and recorded the added delay. Table 5.7 displays these results. Nearly negligible delay is added in removing an average of 25.4% and 32.7% of false positives by the Dimension and Color filters, respectively, while keeping the added false negatives to 0. The addition of these two filtering steps is, therefore, fully justified.

			Cutoff values	
	TBs	**LPs**	**TBs**	**LPs**
Min Green Pixels	42%	-	35%	-
Max Green Pixels	85%	-	90%	
Min White Pixels	10%	45%	7.5%	35%
Max White Pixels	65%	92%	75%	100%

Table 5.6: Color information (in pixels) taken from 150 traffic boards and 250 license plates, and the Final cutoff values.

	Dimension Filtering	**Color Filtering**
Removed FPs	25.4%	32.7%
Added FNs	0	0
Added Delay	1.260 ms	5.128 ms

Table 5.7: Effectiveness of the Dimension and color filtering steps. (FPs: False Positives, FNs: False Negatives)

5.2.2 Texture Analysis

Block-based Filter

This filter is based on the observation that the OOIs contain edge-dense areas in their canny edge map, unlike unwanted objects that either have no specific pattern in their edge density or are edge-light. Therefore, by counting the number of edge-dense areas in a given ROI and using the right thresholds, we are able to drastically reduce the set of unwanted objects and retain the majority of the OOIs. Algorithm 1 briefly explains this process, whereas Figures 5.1 and 5.2 visualizes it. For a given ROI to pass, the following conditions, with respect to thresholds t_{max}, t_{avg}, t_1, t_2 and t_{block}, need to be satisfied:

- The maximum cell of a given ROI must be close to the maximum cell of all ROIs, t_{max}. Additionally, this cell must be adjacent to cells that contain a relatively large number of edges. Specifically, it must be adjacent to cells containing no less than t_{avg} edges.
- Each ROI must contain at least t_{block} blocks of adjacent edge-filled cells, i.e. at least t_{block} sequence(s) of at least t_2 cells containing values of at least t_1. The value in each cell corresponds to the number of edges found in it.

All thresholds are determined experimentally. Table 5.8 shows some different threshold combinations that with their effectiveness based on the percentage of removed false positives and percentage of added false negatives.

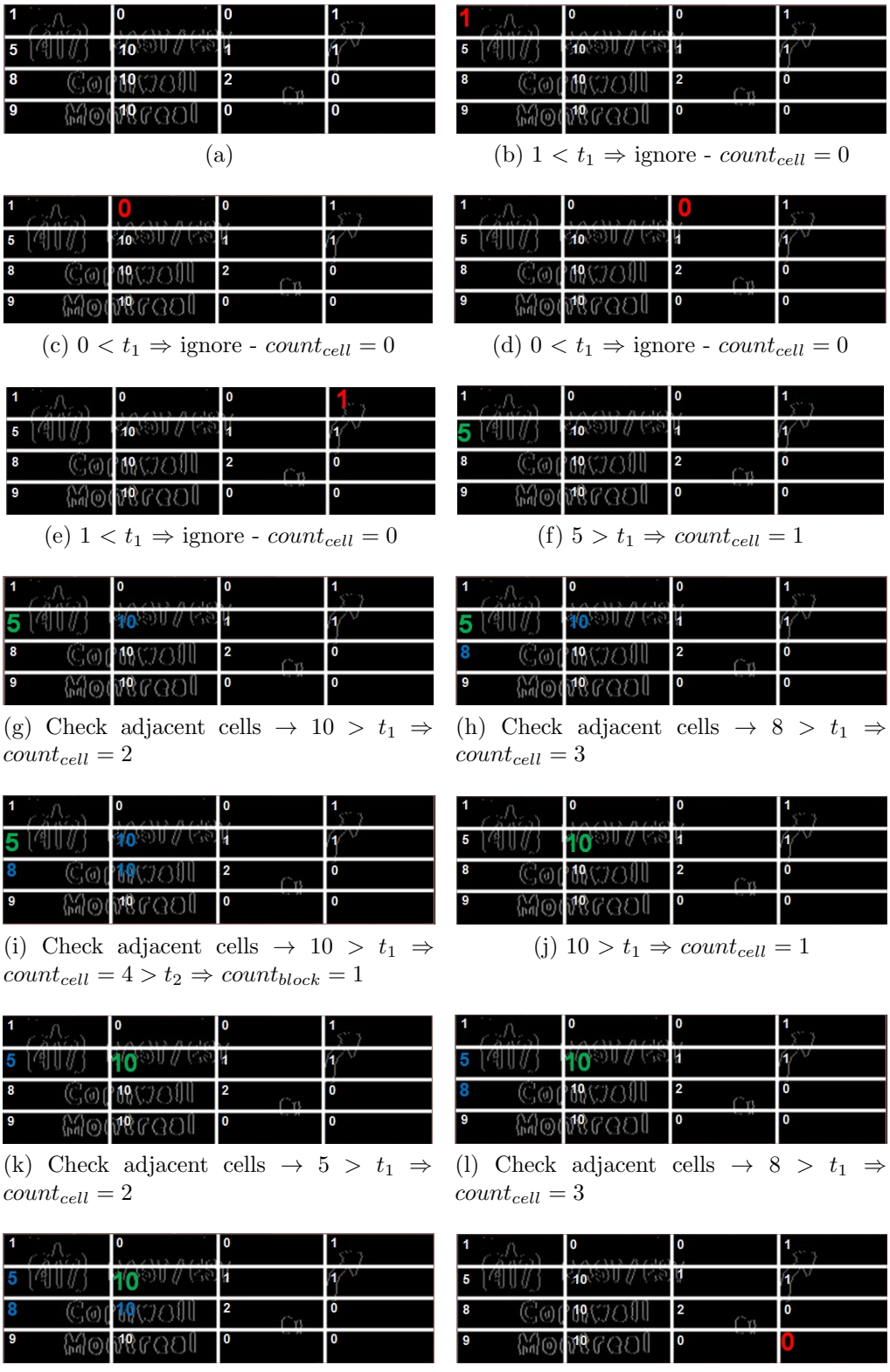
Algorithm 1 Block-based Filter

Each ROI is divided into 4×4 cells
Number of edges in each cell is counted and stored in an array
For each ROI, the number of blocks containing adjacent edge-filled cells are computed → `count_block`
Using thresholds t_{max}, t_{avg}, t_1, t_2 and t_{block}, decide to whether pass or reject given ROI

	Removed FPs	Added FNs
$t_{max} = 5$, $t_{avg} = 3$, $t_1 = t_2 = t_3 = 3$	75.2%	17.5%
$t_{max} = 5$, $t_{avg} = 3$, $t_1 = 3, t_2 = t_3 = 2$	63.2%	10.1%
$t_{max} = 5$, $t_{avg} = 3$, $t_1 = t_2 = t_3 = 2$	59.3%	6.4%
$t_{max} = 4$, $t_{avg} = 2$, $t_1 = t_2 = t_3 = 2$	53.7%	3.5%
$t_{max} = 4$, $t_{avg} = 2$, $t_1 = t_2 = 2, t_3 = 1$	49.8%	1.0%

Table 5.8: Some of the attempted combinations for the block-based filter and their effectiveness. (FPs: False Positives, FNs: False Negatives)

Figure 5.1: Maximum cell (10) $> t_{max}$ and is adjacent to cells containing $> t_{avg}$ cells $\Rightarrow$ First condition satisfied



(a)

(b) $1 < t_1 \Rightarrow$ ignore - $count_{cell} = 0$

(c) $0 < t_1 \Rightarrow$ ignore - $count_{cell} = 0$

(d) $0 < t_1 \Rightarrow$ ignore - $count_{cell} = 0$

(e) $1 < t_1 \Rightarrow$ ignore - $count_{cell} = 0$

(f) $5 > t_1 \Rightarrow count_{cell} = 1$

(g) Check adjacent cells $\rightarrow 10 > t_1 \Rightarrow count_{cell} = 2$

(h) Check adjacent cells $\rightarrow 8 > t_1 \Rightarrow count_{cell} = 3$

(i) Check adjacent cells $\rightarrow 10 > t_1 \Rightarrow count_{cell} = 4 > t_2 \Rightarrow count_{block} = 1$

(j) $10 > t_1 \Rightarrow count_{cell} = 1$

(k) Check adjacent cells $\rightarrow 5 > t_1 \Rightarrow count_{cell} = 2$

(l) Check adjacent cells $\rightarrow 8 > t_1 \Rightarrow count_{cell} = 3$

(m) Check adjacent cells $\rightarrow 10 > t_1 \Rightarrow count_{cell} = 4 > t_2 \Rightarrow count_{block} = 2$

(n) Iterate through remaining cells $\rightarrow count_{block} = 6 > t_{block} \Rightarrow$ ROI passes

Figure 5.2: Example: $t_1 = t_2 = 2, t_{block} = t_{avg} = 3, t_{max} = 5$

Line and Character-based Filter (LC filter)

This filter functions as follows:

- An edge is passed as a character if it satisfies the dimension constraints min_{width}, min_{height}, min_{ar}, max_{width}, max_{height} and max_{ar}.
- Two edges are considered to be on the same line if the difference between their y coordinates is at most $line_{diff}$.
- For a ROI to be identified as a traffic board, it needs to satisfy the thr_{line} requirement
- For a ROI to be identified as a license plate, it needs to satisfy the thr_{char}, lp_{widMin} and lp_{widMax} requirements.

Algorithm 2 Character and Line-based Filter Pseudocode

Require: Input ROI
  Apply Canny Edge detector on the given ROI
  Count number of character-looking edges, subject to dimension constraits in Table 5.9
  Count number of lines using $line_{diff}$
  if Traffic board **then**
    Use number of lines and thr_{line} to decide whether to accept or reject
    If accepted, separate highway number from text lines
  else
    Use number of characters and thr_{char} to decide whether to accept or reject
    Remove overlapping text lines
    Set text width and height restrictions lp_{widMin} and lp_{widMax}. If passes, ROI is accepted
  end if

Through observing the set of OOIs we have (from video frames and still images), we are able to see a clear difference in the number of lines between the OOIs and unwanted objects, thus rendering thr_{line} appropriate for TBs. However, thr_{char} is not effective for TBs since we are not able to spot a pattern that is strong enough to effectively separate the OOIs from the false positives. Likewise, thr_{line} is not an effective threshold for LPs. However, thr_{char} proved to be very effective. Determining the right value, however, was a challenging task since the initial set of objects already passed through many filtering steps.

The values used for dimension constraints min_{width}, min_{height}, min_{ar}, max_{width}, max_{height} and max_{ar} are determined experimetnally. Their values are shown in Table 5.9.

min_{width}	5	min_{ar}	0.3	$line_{diff}$	15
max_{width}	35	max_{ar}	1.3		
min_{height}	10	lp_{widMin}	20		
max_{height}	35	lp_{widMax}	31		

Table 5.9: Final values for the Dimension constraints, $line_{diff}$, lp_{widMin} and lp_{widMax}

Details on how the thresholds thr_{line} and thr_{char} are obtained and the overall effectiveness of the LC filter are shown in Section 5.3, where precision-recall curves are presented.

5.3 Results

This section illustrates the performance of our system in terms of the following metrics:

- **Detection module**: Precision and recall are used as primary metrics. Detection rate, miss rate and time delay are also used as metrics, but are observed under the influence of different *frame resolutions*.
- **Recognition module**: Recognition rate, false reading rate and time delay are used as metrics.

5.3.1 Detection Module

Tables 5.11-5.15 (Tables 5.11-5.12 for traffic boards, and Tables 5.13-5.15 for license plates) contain details on the performance of our system for each video of each speed scenario. The optimal values for thr_{char} (license plates) and thr_{line} (traffic boards) are used for these tables. Details on how these values are determined are provided in the upcoming subsection.

Video#	# Boards	# Detections	# Correct Detections	# FNs	# FPs
1	0	1	0	0	1
2	0	0	0	0	1
3	1	1	1	0	0
4	1	1	1	0	0
5	1	0	0	1	0
6	1	1	1	0	0
7	2	2	2	0	0
8	3	4	3	0	1
9	1	1	1	0	0
	10	**11**	**9**	**1**	**2**

Table 5.10: Performance on Traffic Boards - Mid speed scenario, 50 - 100 km/h

Video#	# Boards	# Detections	# Correct Detections	# FNs	# FPs
1	1	1	1	0	0
2	1	1	1	0	0
3	2	2	2	0	0
4	3	3	3	0	0
5	1	1	1	0	0
6	1	1	1	0	0
7	2	2	2	0	0
8	6	6	6	0	0
9	1	1	1	0	0
10	1	1	1	0	0
11	1	1	1	0	0
12	0	0	0	0	0
13	1	1	1	0	0
14	1	1	1	0	0
15	2	2	2	0	0
16	8	8	8	0	0
17	4	4	4	0	0
18	5	3	3	2	0
19	3	1	1	2	0
20	7	7	7	0	0
21	2	1	1	1	0
22	5	4	4	1	0
23	3	3	3	0	0
24	3	3	3	0	0
25	2	1	1	1	0
26	6	6	6	0	0
27	10	12	10	0	2
	82	**77**	**75**	**7**	**2**

Table 5.11: Performance on Traffic Boards - Highway speed scenario, 100+ km/h

Video#	# Boards	# Detections	# Correct Detections	# FNs	# FPs
1	0	0	0	0	0
2	0	1	0	0	1
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0
6	0	0	0	0	0
7	1	1	1	0	0
8	0	1	0	0	1
9	0	0	0	0	0
10	0	0	0	0	0
11	0	1	0	0	1
12	0	0	0	0	0
13	1	1	1	0	0
14	0	1	0	0	1
15	1	1	1	0	0
16	1	1	1	0	0
17	1	0	0	1	0
18	6	6	6	0	0
19	0	0	0	0	0
20	0	0	0	0	0
21	0	0	0	0	0
22	0	0	0	0	0
23	0	0	0	0	0
24	0	1	0	0	1
25	0	0	0	0	0
	11	**15**	**10**	**1**	**5**

Table 5.12: Performance on Traffic Boards - Slow speed scenario, 0 - 50 km/h

Video#	# LPs	# Detections	# Correct Detections	# FNs	# FPs
1	1	2	1	0	1
2	4	5	4	0	1
3	5	6	4	1	2
4	0	0	0	0	0
5	0	0	0	0	0
6	2	2	1	1	1
7	0	0	0	0	0
8	7	7	6	1	1
9	2	2	2	0	0
10	1	2	1	0	1
11	0	1	0	0	1
12	3	3	3	0	0
13	0	0	0	0	0
14	1	1	1	0	0
15	1	1	1	0	9
16	3	5	3	0	1
17	3	4	3	0	1
	33	**41**	**30**	**3**	**11**

Table 5.13: Performance on License Plates - Highway speed scenario, 100+ km/h

Video#	# LPs	# Detections	# Correct Detections	# FNs	# FPs
1	5	1	1	4	1
2	9	8	6	3	2
3	3	3	3	0	0
4	1	2	1	0	1
5	1	1	1	1	0
6	2	2	2	0	0
7	1	0	0	1	0
8	1	1	1	0	1
9	0	0	0	0	0
	23	**18**	**15**	**8**	**3**

Table 5.14: Performance on License Plates - Mid speed scenario, 50 - 100 km/h

Video#	# LPs	# Detections	# Correct Detections	# FNs	# FPs
1	3	3	3	0	0
2	4	5	4	0	1
3	1	1	1	0	0
4	9	9	7	2	2
5	9	10	8	1	2
6	1	1	1	0	0
7	0	1	0	0	1
8	2	2	1	1	1
9	3	3	3	0	0
10	6	6	4	2	2
11	4	4	3	1	1
12	4	4	3	1	1
13	0	0	0	0	0
14	5	7	4	1	3
15	0	0	0	0	0
16	0	0	0	0	0
17	7	9	7	0	1
18	0	0	0	0	0
19	11	10	10	1	0
20	6	7	5	1	2
21	8	9	8	0	1
22	6	6	5	1	0
23	4	5	4	0	1
24	13	14	13	0	1
25	8	8	8	0	0
26	11	10	10	1	0
27	10	9	9	1	0
	135	**143**	**121**	**14**	**22**

Table 5.15: Performance on License Plates - Slow speed scenario, 0 - 50 km/h

Precision and Recall

According to [78], precision and recall are defined as follows:

- **Precision**: the fraction of the retrieved instances that are relevant
- **Recall:** the fraction of the relevant instances that are retrieved

In other words, precision is a measure of the system's sensitivity to false positives, whereas recall is a measure of the system's sensitivity to false negatives. The final precision and recall values are given in Table 5.18. In these tests, a detection of an OOI is validated if the corresponding object is detected within the first 10 frames in which it

appears in. Attempting these tests on a per-frame basis is currently not feasible since the entire set consists of more than 30,000 frames without a ground truth.

Precision-recall curves have been plotted for TBs with respect to threshold thr_{line}, and for LPs with respect to the threshold thr_{char} (mentioned in Section 5.2.2). The curves are shown in Figure 5.3.
In order to determine the best value for thr_{char}, we need to compute the slope of each transition, i.e. when thr_{char} goes from n to $n+1$. A large slope corresponds to a big rise in precision and/or a small drop in recall, whereas a small slope corresponds to a small rise in precision and/or a big drop in recall. Therefore, the optimal transition is the one preceding the transition that yields the smallest slope. In other words, the optimal thr_{char} value is the initial value of the transition that yields the smallest slope, which is the the final value of the preceding transition. According to Table 5.16, the transition yielding the smallest slope is where thr_{char} goes from $6 \rightarrow 7$. The optimal value of thr_{char} is, therefore, **6**.

This result is further bolstered by observing the remaining curves. As shown in the highway, mid-speed and slow-speed scenario curves, the transition $6 \rightarrow 7$ of thr_{char} yields slopes of 0, 0.1 and 0.25 respectively.

thr_{char} **Transition**	**ΔPrecision**	**ΔRecall**	**Slope**
1 $\rightarrow$ *2*	$\|0.30 - 0.25\| = 0.05$	$\|0.95 - 0.96\| = 0.01$	$0.05 \,/\, 0.01 = 5$
2 $\rightarrow$ *3*	$\|0.44 - 0.30\| = 0.14$	$\|0.94 - 0.95\| = 0.01$	$0.14/\ 0.01 = 14$
3 $\rightarrow$ *4*	$\|0.61 - 0.44\| = 0.17$	$\|0.91 - 0.94\| = 0.03$	$0.17 \,/\, 0.03 = 5.7$
4 $\rightarrow$ *5*	$\|0.70 - 0.61\| = 0.09$	$\|0.90 - 0.91\| = 0.01$	$0.09 \,/\, 0.01 = 9$
5 $\rightarrow$ *6*	$\|0.82 - 0.7\| = 0.12$	$\|0.87 - 0.90\| = 0.03$	$0.12 \,/\, 0.03 = 4$
6 $\rightarrow$ *7*	$\|0.84 - 0.82\| = 0.02$	$\|0.76 - 0.87\| = 0.11$	$0.02 \,/\, 0.11 = 0.2$
7 $\rightarrow$ *8*	$\|0.87 - 0.84\| = 0.03$	$\|0.73 - 0.76\| = 0.03$	$0.03 \,/\, 0.03 = 1$
8 $\rightarrow$ *9*	$\|0.93 - 0.87\| = 0.06$	$\|0.68 - 0.73\| = 0.05$	$0.06 \,/\, 0.05 = 1.2$
9 $\rightarrow$ *10*	$\|0.96 - 0.93\| = 0.03$	$\|0.61 - 0.68\| = 0.07$	$0.05 \,/\, 0.01 = 5$

Table 5.16: Overall Slope values for License plates from $n = 1$ to $n = 10$

The same procedure applies to TBs but with threshold thr_{line} instead of thr_{char} (explained in Section 5.2.2). Table 5.17 and the curves in Figure 5.4 show why **5** is the optimal value for thr_{line}.

thr_{line} **Transition**	**ΔPrecision**	**ΔRecall**	**Slope**
1 → *2*	$\|0.65 - 0.53\| = 0.13$	$\|0.98 - 0.99\| = 0.01$	0.13 / 0.01 = 13
2 → *3*	$\|0.75 - 0.65\| = 0.10$	$\|0.96 - 0.98\| = 0.02$	0.01/ 0.02 = 5
3 → *4*	$\|0.83 - 0.65\| = 0.08$	$\|0.94 - 0.96\| = 0.02$	0.08 / 0.02 = 4
4 → *5*	$\|0.91 - 0.83\| = 0.07$	$\|0.92 - 0.94\| = 0.02$	0.07 / 0.02 = 3.5
5 → *6*	$\|0.93 - 0.91\| = 0.02$	$\|0.69 - 0.92\| = 0.23$	0.02 / 0.23 = 0.09
6 → *7*	$\|0.95 - 0.93\| = 0.02$	$\|0.47 - 0.69\| = 0.22$	0.02 / 0.22 = 0.09
7 → *8*	$\|0.96 - 0.95\| = 0.01$	$\|0.33 - 0.47\| = 0.14$	0.01 / 0.14 = 0.07
8 → *9*	$\|0.97 - 0.96\| = 0.01$	$\|0.20 - 0.33\| = 0.13$	0.01 / 0.13 = 0.08
9 → *10*	$\|0.98 - 0.97\| = 0.01$	$\|0.09 - 0.20\| = 0.11$	0.01 / 0.11 = 0.09

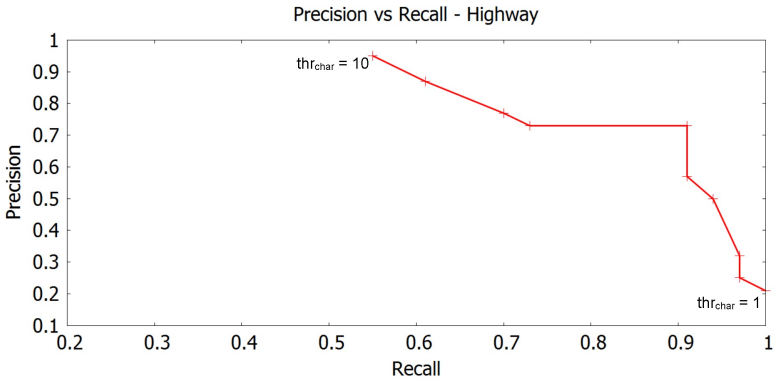
Table 5.17: Overall Slope values for Traffic Boards from $n = 1$ to $n = 10$

5.3.2 Frame Resolution

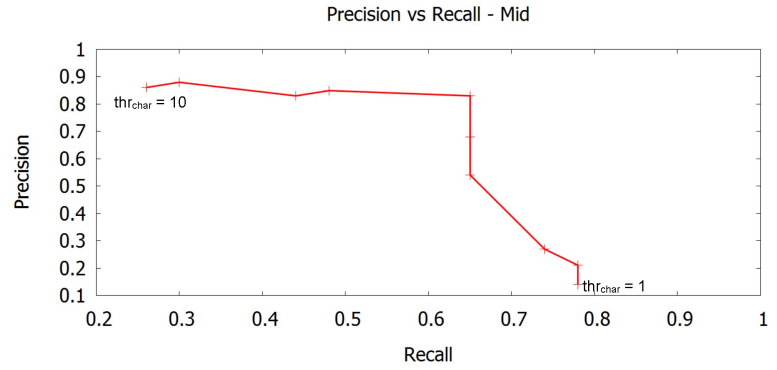
As with any detection/recognition-related issue, the effect the resolution of the processed frames has on the performance of the system needs to be investigated.

As mentioned at the beginning of this section, the test videos consist of mostly manually-taken footage using a high-end camera (1920 × 1080) and the remaining are downloaded from the web with medium quality (1280 × 720). However, in order to achieve the best performance in terms of detection rate, miss rate and the added time delay, different resolutions and aspect ratios need to be tested. To do so, we first need to determine the ideal aspect ratio for our application. Through simple experiments we determined the ideal aspect ratio to be 3:2 (width:height). We believe that this is not a coincidence since our OOIs exhibit a similar aspect ratio.

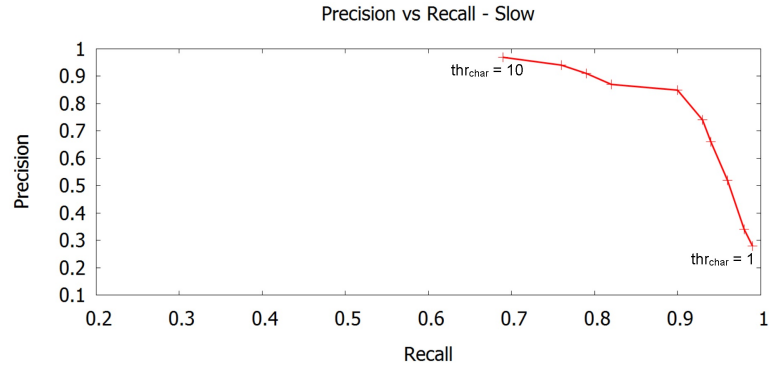
Therfore, in the subsections below all the experiments are based on different 3:2 resolutions. For each resolution, the detection rate, the miss rate and the time delay were calculated and measured. Both subsections contain tables and graphs that illustrate the results obtained.



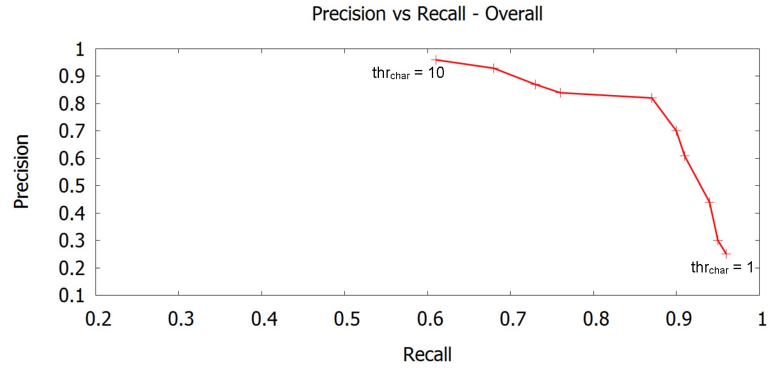
(a) precision-recall curve for the highway speed scenario



(b) precision-recall curve for the mid speed scenario

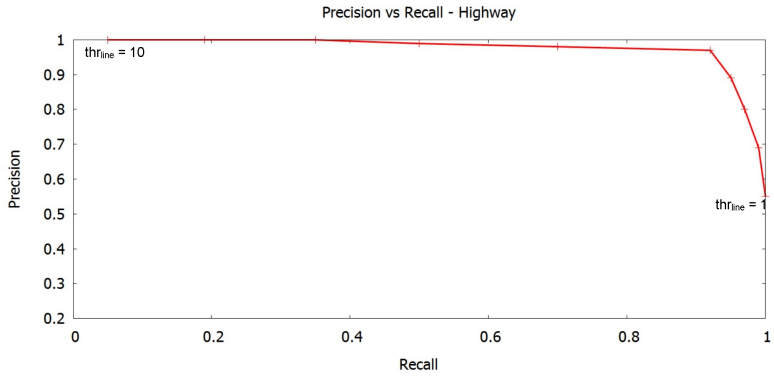


(c) precision-recall curve for the slow speed scenario

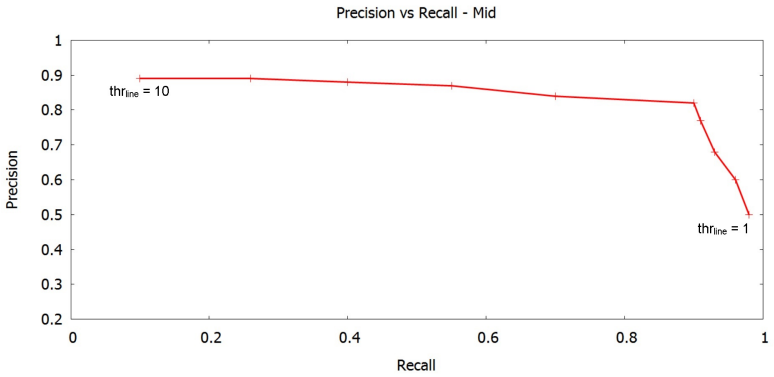


(d) Overall precision-recall curve

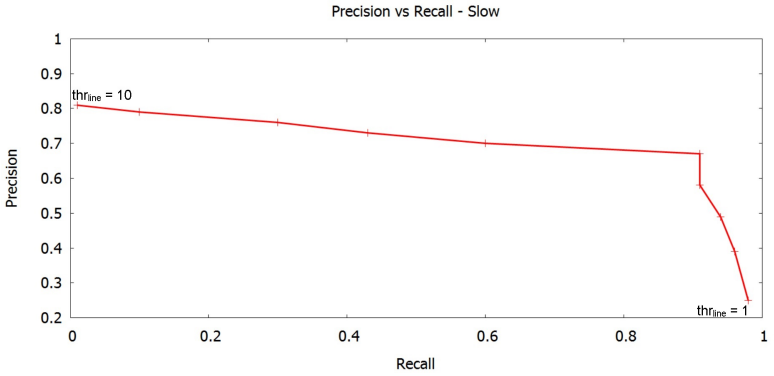
Figure 5.3: precision-recall curves for LPs with respect to thr_{char}



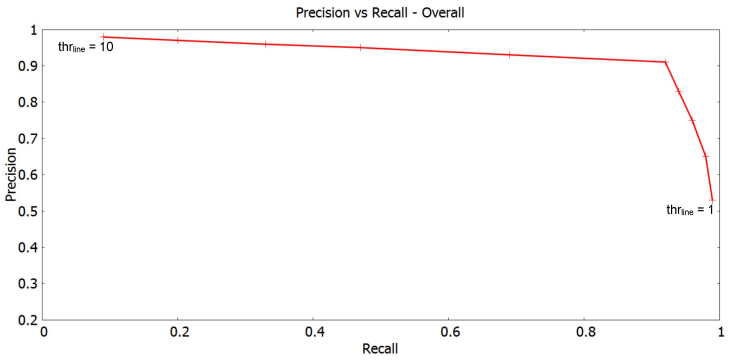
(a) precision-recall curve for the highway speed scenario



(b) precision-recall curve for the mid speed scenario



(c) precision-recall curve for the slow speed scenario



(d) Overall precision-recall curve

Figure 5.4: precision-recall curves for TBs with respect to thr_{line}

	Total	Total Detection	Total Correct Detection
TBs	102	103	94
LPs	191	202	166

Precision	Recall
91.26%	*92.16%*
82.17%	*86.91%*

Table 5.18: Final precision and recall values — $thr_{char} = 6$ and $thr_{line} = 5$

Detection Rate and Miss Rate

For the sake of simplicity, detection and miss rates are based on a detection-per-multiframe basis, i.e. a detection of a TB/LP is validated if the corresponding object is detected in at least one frame in which the object appears in. The thresholds used for these tests are the ones deduced from Section 5.3.1, i.e. $thr_{char} = 6$. $thr_{line} = 5$.

	Traffic Boards		License Plates	
Resolution	**Detection Rate**	**Miss Rate**	**Detection Rate**	**Miss Rate**
300 x 200	90.38%	9.62%	86.39%	13.61%
600 x 400	95.19%	4.81%	89.01%	10.99%
900 x 600	98.08%	1.92%	91.62%	8.38%
1200 x 800	91.35%	8.65%	88.48%	11.52%
1500 x 1000	81.73%	18.27%	83.77%	16.23%
1800 x 1200	77.88%	22.12%	80.11%	19.89%
2100 x 1400	72.12%	27.88%	76.44%	23.56%
2400 x 1600	63.46%	36.54%	69.59%	30.41%

Table 5.19: Detection and Miss rates on different 3:2 resolutions

The deterioration of the detection rates in very high resolutions, shown in Table 5.19 and Figure 5.5, can be attributed to the MSER extraction step. Higher resolutions result in a less overall contrast between the different objects found in a given frame, thus diminishing the effectviness of the MSER extraction step as it is based on spotting differently-contrasted objects within the frame. In addition, a frame loses information as it exceeds the original resolution (1920×1080 or 1280×800), which inevitably leads to a degradation in the system's effectiveness. It is also evident that the optimal resolution for this application is **900×600** (54,000 pixels).

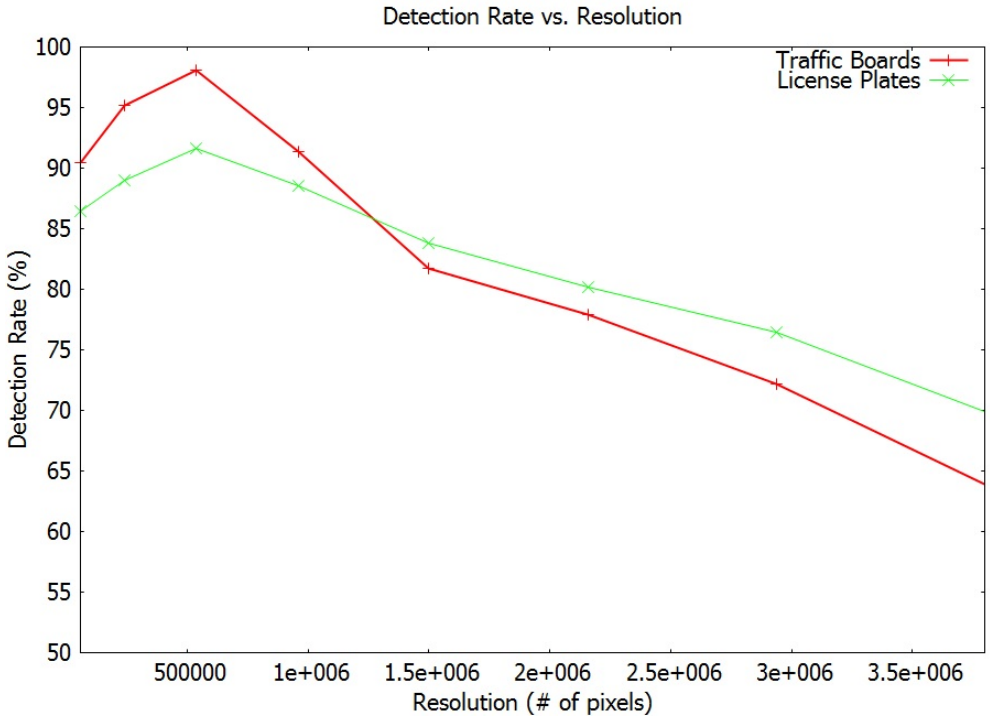


Figure 5.5: Detection and Miss rates on different 3:2 resolution

Time Delay

Resolution	MSER Ext.	Dim. filter	Color filter	Block filter	LC filter
300×200	8.564	0.0567	0.2682	0.5211	0.6502
600×400	34.58	0.4186	3.077	4.805	7.543
900×600	77.26	1.260	5.128	6.250	7.672
1200×800	141.1	2.280	6.761	5.844	6.230
1500×1000	215.6	3.193	6.953	6.460	3.355
1800×1200	307.9	3.679	6.387	3.360	1.139
2100×1500	423.74	3.697	5.823	2.315	0.171
2400×1600	556.72	3.294	5.313	1.604	0.0242

Table 5.20: Time delay of each step, in milliseconds, on different 3:2 resolutions

As shown in Figure 5.6, the total time delay added increases linearly (Figure 5.6) with the frame resolution. It is important to note, however, that the MSER extraction step is the biggest contributor to the total added delay, with an average contribution of 88.88%. The remaining filters, however, add little to no delay regardless of the frame resolution. The use of each filter is, therefore, justified.

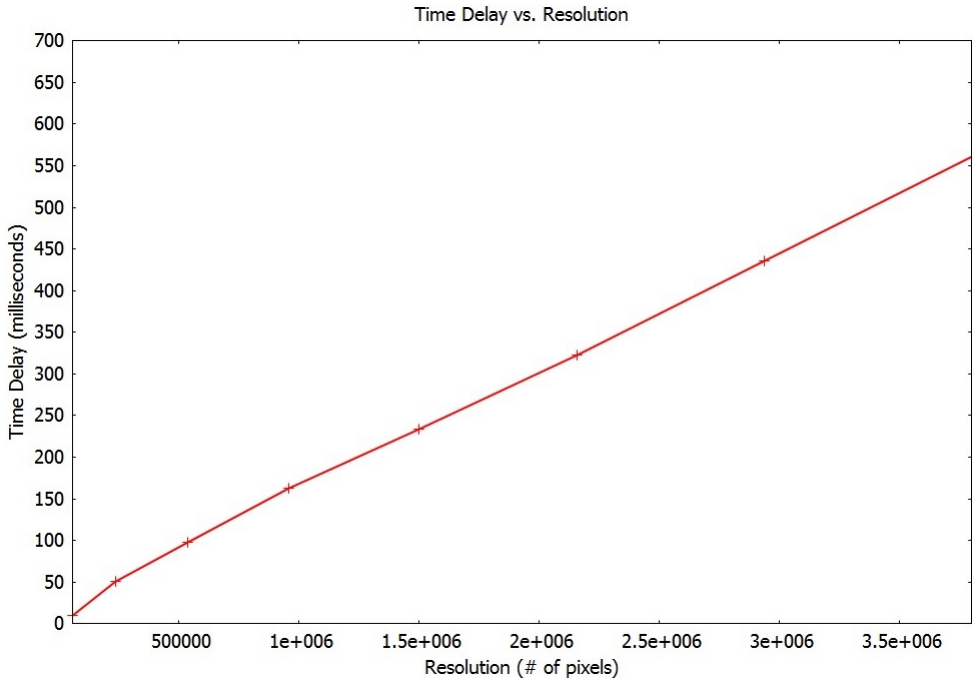


Figure 5.6: Total time delay, in milliseconds, in different 3:2 resolutions

5.3.3 Recognition Module

Two substeps are involved in the recognition stage, text correction and OCR activation control.

Text Correction

This substep ensures that the text displayed for the user is always meaningful. It does so by utilizing a pre-built dictionary that is cross-checked with every output produced by the OCR. If a semi-meaningful output is produced by the OCR, its closest match from the dictionary is displayed instead, otherwise no text is displayed. A 'similarity' thershold, thr_{sim}, is used to differentiate between a semi-meaningful output from a meaningless output. Algorithm 3 demonstrates this process. Figure 5.7 illustrates the process taken to determine the best thr_{sim} value.

The best way to gauge the performance of each thr_{sim} value is to compute the number of false readings. A false reading, in our work, is defined as a misreading, where the incorrect word from the dictionary is displayed, or a missed reading, where it is deemed that no word in the dictionary is similar enough to the desired text. Therefore, as is clearly presented in Figure 5.7, the value **0.65** is right choice as the absolute minimum is achieved at this point, with a false reading rate of 1.064%.

Algorithm 3 Text Correction

Build dictionary based on geographial locaiton
Apply OCR and store output → `OCR_output`
Tokenize `OCR_output` → `Tokens[]`
Compute similarity between every token in `Tokens[]` and every word in the dictionary → `sim_value`
Using `sim_value` and thr_{sim}, decide whether to accept this token or not.

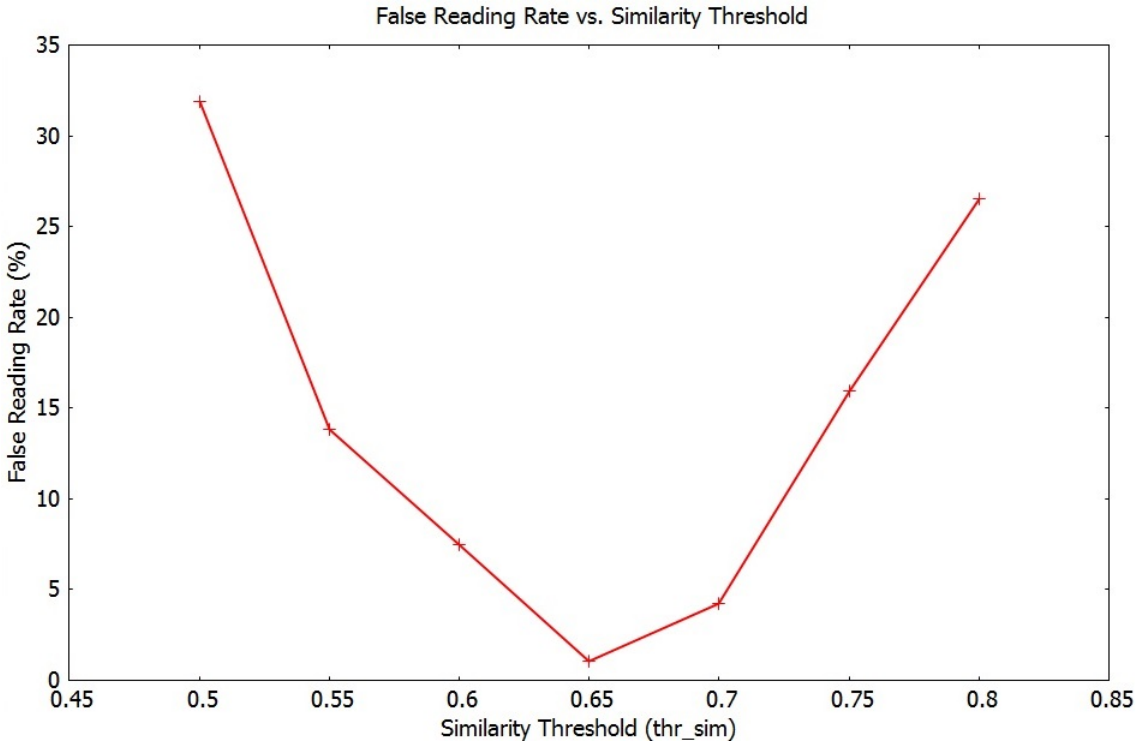


Figure 5.7: False Reading Rates with respect to thr_{sim}

OCR Activation Control

The OCR is only activated when the average height exceeds a height threshold, thr_{height}, thus preventing processing time from being wasted. The optimal value for thr_{height} would be one that best balances between the overall added delay and the recognition rate. Figure 5.8 illustrates the process taken to determine the required thr_{height} value.

As shown in the Figure, lower thr_{height} values correspond to higher recognition rates and larger OCR delays. However, the rate at which the recognition rate drops, average of a 0.2533 drop per a 0.5 increase in thr_{height}, is significantly smaller than then rate at which the OCR delay drops, average of a 31.04 drop per a 0.5 increase in thr_{height}. This allows us to target the thr_{height} that results in a very small delay without the fear of compromising the associated recognition rate. Accordingly, we deem the value of **17.5** to be suitable for

our application, with a recognition rate of 85.11% and an OCR delay of 121.6 ms.

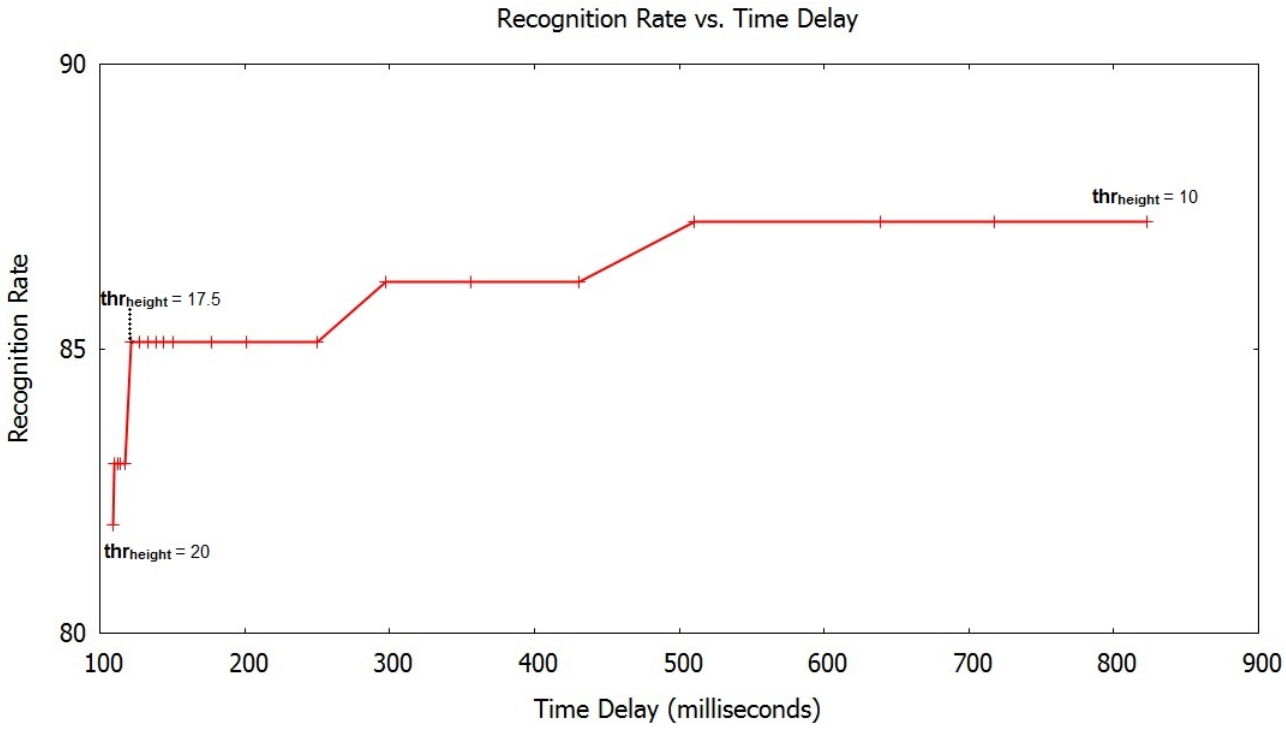


Figure 5.8: Recognition Rate vs. Time Delay with respect to thr_{height}

5.3.4 Interpretation of Results

The following can be deduced from the reported results:

- The precision and recall rates obtained for TBs are noticeably better than those obtained for LPs. This discrepancy in performance can be attributed to the textual features exhibited by LPs and TBs. Traffic boards are usually bigger in size, have smoother text and have color features (green background, white text) that very few other objects (if any) share. License plates, on the other hand, come in significantly smaller dimensions and therefore have smaller texts that are harder to read. Additionally, LPs do not posses a unique color feature that can be used to easily differentiate it from other objects.

- For TBs, the best results were obtained in the highway speed scenario, with precision and recall values of 0.97 and 0.92, respectively, compared to the 0.82 and 0.90 for the mid-speed scenario and 0.67 and 0.91 for the slow-speed scenario. This difference is, firstly, due to the fact that highways contain far less potential false positives than ordinary roads do. Secondly, and more importantly, the number of TBs present in our highway test video set is significantly greater (82) than the number of boards in the mid-speed and slow-speed set combined (21). This is shown in Table 5.1.

This results in more solid precision and recall values that are more indicative of the system's performance for the highway scenario than for the mid-speed and slow-speed scenarios. This, however, is not a concern for us as TBs are more important to the driver on a highway than on slower-moving streets and avenues.

- Similar to the previous point, the best LP results were achieved in slow-speed scenarios, with precision and recall values of 0.85 and 0.90 respectively, compared to the 0.83 and 0.65 for the mid-speed scenario and the 0.73 and 0.91 for the highway scenario. The reasons behind this difference are similar to the ones mentioned in the previous point. Our test video set contained 135 license plates for the slow-speed scenario while the two other scenarios add up to only 56 (Table 5.1).
- The text correction and OCR activation steps significantly help in properly integrating an open source OCR into our application. To achieve further superior results, however, a manually-implemented character and/or word recognizer needs to be incorporated. This is especially true when dealing with very small texts such as the ones found on license plates.

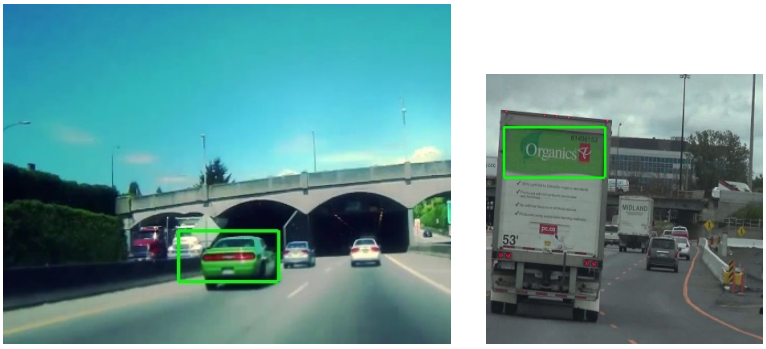
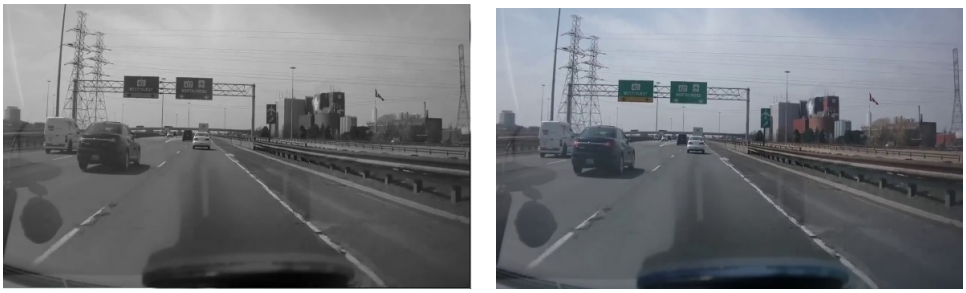


Figure 5.9: Sample false positives in detecting TBs. Such false positives have a mostly green background, to pass the color filter, and are edge-dense, to pass the texture filters.



(a) MSER image: 0 extracted regions (b) Two false negativse

Figure 5.10: Sample false negatives in detecting TBs. Such false negatives result from failing to extract the right MSERs. This occurs when the camera used exhibits poor color saturation (these images were taken from a video of subpar quality)

Figure 5.11: Sample false positives in detecting LPs. For such a false positive to occur, the false region must be edge-heavy as well as be surrounded by edge-light areas so that it passes all texture filters. Additionally, to pass the color filter, it must exhibit a certain ratio of white to non-white pixels

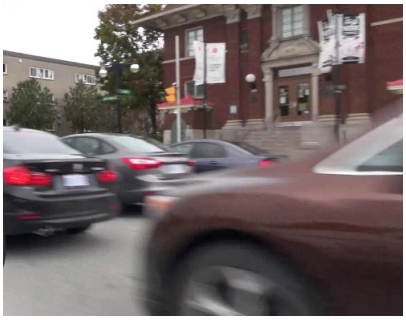


Figure 5.12: Sample false negatives in detecting LPs. Such false negatives result from having an LP appear blurry in all frames in which it appears

5.4 Comparison with Other Works

This section compares our work with two recent works that target similar objects.

5.4.1 License Plates

Zhou et al. in [102] rely on Principal Visual Word (PVW) and SIFT features. A training set of labeled images is used to generate PVWs for each character in the set of available license plates. Afterwards, SIFT features are extracted from the characters of a given candidate license plate and are compared with the list of generated PVWs.
Reported Dataset(s): The training dataset used included 220 Chinese license plate images and 364 U.S license plates collected from the web. Likewise, the testing dataset[1] consisted of 410 Chinese license plates, 160 of which were obtained from the web. The

[1] available:http://home.ustc.edu.cn/~zhwg/download/LicensePlate_dataset.rar

U.S. license plates were taken from the Caltech cars dataset[2] and contained 112 images with a resolution of 892×592. Since our work deals primarily with North American plates, we base the comparison on the U.S. plates from the Caltech dataset. Also, we are not able to access the Chinese dataset since the link provided by the authors seems to be broken. Samples of the Caltech dataset are shown in Figure 5.13.
Reported Metric(s): Accuracy and Time delay

Reported Results

	Accuracy	**Time Delay (ms)**
Zhou et al.	84.8%	7190
Proposed work	91.2%	112.5

Table 5.21: License plate detection comparison with [102]

As shown above in Table 5.21, our system achieves a considerably higher accuracy than Zhou et al's system. Additionally, while our system is tested on higher specs, the time delay our system adds is significantly (64 times) smaller than the delay added by Zhou et al's system. Our system, therefore, is better suited for the automotive context both in terms of accuracy and real-time performance. The few LPs that our system failed to detect were colored LPs. With the LP parameters mentioned in Section 5.2.1, the color filter discards any colored region since the vast majority of LPs nowadays have black text printed on white backgrounds. As support to this claim, none of the 191 LPs in our video test set is colored. An example of a colored LP in the U.S dataset is shown in Figure 5.14.

Figure 5.13: Sample images from the Caltech dataset used in [102]

[2] available:http://www.vision.caltech.edu/Image_Datasets/cars_markus/cars_markus.tar

Figure 5.14: A colored LP from the Caltech dataset used in [102]

5.4.2 Traffic Boards/Panels

To our knowledge, very few pervious works attempted to detect road traffic boards with good results. The most notable work we found belongs to Gonzalez et al. in their paper [34] that was published in early 2014. Gonzales et al. rely on color segmentation (blue and white) and a BOVW technique in conjuction and with a Support Vector Machine (SVM) and a Naive Bayes classifier in order to detect the traffic panels of interst. Afterwards, a character/symbol recognizer is applied followed by a word recognizer that is based on a unigram probabilstic langauge model.
Reported Dataset(s): The authors used Google street view to create their training dataset (5514 images) and testing dataset (10763 images). Each dataset consisted of mostly negative samples, i.e. no traffic panel, and the positive samples consisted of either blue or white panels located either at the side of the road (lateral view) or above the vehicle (upper view). The reported number of traffic panels in the training set is 383 blue (314 lateral, 79 upper) and 116 white (35 lateral and 81 upper). The reported number of traffic panels in the testing set is 129 blue (84 lateral in 680 frames, 45 upper in 164 frames) and 57 white (24 lateral in 87 frames, 35 upper in 123 frames). Since our work currently deals with colored TBs and that very few, if any, white TBs are found in North America, we base the comparison on the blue traffic panels and the negative samples from the test set. Samples of the test set are shown in Figures 5.15 and 5.16.
Reported Metric(s): Precision, Recall and f-score (mean of precision and recall).

	Blue Lateral		*Blue Upper*	
	Our work	**Gonzalez et al.**	**Our work**	**Gonzalez et al.**
Precision	0.8210	0.9253	0.8950	0.9536
Recall	0.9271	0.6042	0.9146	0.8963
f-score	0.8740	0.7674	0.9048	0.9300
Detection Rate	0.9643	0.9523	0.9777	0.9777

Table 5.22: Blue traffic panel detectoion comparison with [34]

We adopted the same experimentation procedure used by Gonzalez et al. in their work. That is, we compute the precision and recall values on a per-frame basis, i.e. each frame

is evaluated separately. The traffic panel detection results are computed on a multiframe basis where a successful detection of a panel is counted if the corresponding panel is detected in at least 2 consecutive frames.

Table 5.22 shows that our work achieves superior results in detecting blue traffic panels appearing in the lateral view, and nearly identical results in detecting blue panels appearing in the upper view. The superiority in the lateral view is associated with Gonzalez et al's relatively low recall score, i.e. high false negative number. Our system's immunity to false negatives in detecting traffic panels can be attiributed to the complementary nature of our two major steps, the CCA and TA. Specifically, our color thresholding step in CCA ensures that all blue objects will be considered by the subsequent block-based texture filter. The text-heavy oulook of a traffic panel deems it nearly impossible for the block filter to discard a traffic panel, thus resulting in a high recall score. The few frames in which a traffic panel was discarded are those frames that have the panel at a distance that is far enough for the text to be unreadable. This, however, is a desirable behavior since it will prevent the unreadable traffic panel to be passed on to the OCR module, thus minimizing the overall added delay.

Figure 5.15: Sample positive test images from the dataset created and used in [34]

Figure 5.16: Sample negative test images from the dataset created and used in [34]

Chapter 6

Conclusion

Some of the challenges found in the field of Text Detection and Recognition, such as bad weather conditions, blurry text appearances, presence of obstacles, etc., are out the control of researchers. Additionally, the automotive context adds more challenges such as shaky camera performances resulting from uneven road surfaces or even blurrier texts as a result of high driving speeds. Rather than attempting to resolve each issue individually, we decided to, instead, mitigate the potential damage that could be caused from them. This work did so by adopting a hybrid detection module that takes advantages of the complementary nature that Connected Component Analaysis (CCA) approaches exhibit with Texture Analysis (TA) approaches. The results of the detection module are then fed into a recognition module that uses a powerful Optical Character Recognizer (OCR) that, with the addition of extra substeps that ensure optimal performance, produces accurate results in real-time execution times.

The CCA step uses the popular and relatively cheap blob detection method known as Maximally Stable Extremal Regions (MSER) to produce the first set of Regions of Interest (ROIs). A heuristic approach then takes advantage of the unique geometric and color features exhibited by our two Objects of Interest (OOIs), North American traffic boards (TBs) and license plates (LPs), by setting dimension and color restrictions. The TA step imposes stricter rules by applying filters that exploit the textual differences found between the OOIs and the false positives. The final set of ROIs is then fed into the recognition module. Two subsetps were added to improve the performance of the system. The first substep, text correction, utilizes a geographical dictionary in order to add quality insurance to the output given by the OCR. The second step, OCR activation cotrnol, uses the average height of the detected texts to decide whether to activate or disactivate the OCR. As such, recognition rates are elevated, false readings are reduced and the overall added delay is minimized.

A set of manually taken videos from various regions of Ottawa in addition to several downloaded videos were used to evaluate the performance of our system. The evaluation of the detection mocule was based on precision, recall, detection and miss rates, and time delay. Each CCA and TA filter was individually evaluated based on the number of removed false positives, added false negatives and added delay. The recognition module was evaluated

with respect to recognition and false reading rates, and time delay. The results clearly show that the detection module is very effective with f-scores of 0.9171 and 0.8454 for TBs and LPs, respectively. Detection rates of 98.08% (miss rate of 1.92%) and 91.62% (miss rate of 8.38%) were achieved in a resolution of 900×600 for TBs and LPs, respectively. The recognition module achieved recognition and false reading rates of 85.11% and 1.064%, respectively. The detection module added no more than 98 ms (0.098 seconds) of delay, whereas the recognition module added no more than 122 ms (0.122 seconds). With recognition considered as the most challenging and time-consuming step in any object recognition problem, the results produced render our system suitable for a real-time automotive-based text detection and recognition application. Additionally, our work was compared with two recent and highly regarded works sharing a similar goal. The comparison proved that our work produces superior results in terms of precision, recall, detection accuracy and, most importantly, time delay. Moreover, our work expresses greater flexiblity as it is able to detect both TBs and LPs using the same framework, whereas the works compared with target only either object.

6.1 Future Work

This work can be improved in the following ways:

- It was mentioned in Section 4.3 that the reason behind using a heuristic approach rather than following the more common approach of training a classifier is the current lack of publicly available datasets. However, if a dataset is manually created, whether by taking each frame from a set of videos or by using Google Street View, a powerful classifier could be trained. A Heuristic approach in conjunction with a classifier would surely improve detection results by discarding more false positives.

- Recognition of text still remains a challenging task, especially for small-sized texts. Open source OCRs are not a viable option when dealing with such texts. A manually designed and implemented character and/or word recognizer may be needed. As such, recognition rates, false reading rates and time delay could witness improvements. The design and implementation of a character/word Recognizer, however, is a highly demanding task and is out of the scope of this work.

- Expand the set of detected objects to accomodate for more text-rich objects, such as billboards, restaurant and shop names, etc.

- Expand the recognition scope to include symbols in addition to text. This can be achieved by training a classifier.

- The prospect of transmitting the information gathered by the proposed system to nearby vehicles should be explored. One could look into examples that are either fault-tolerant, energy-aware and, most importantly, context-aware, such as [8], [29]

and[19]. An overview of wireless network and mobile computing algorithms is provided in [7], whereas [21] provides a survey of wireless adhoc network routing protocols. Additionally, a tracking option could be added to enable the tracking of various objects of interest, using technologies such as [81].

- If the previous point (transmission of information) is to be implemented, the trust and security aspects need to be carefully taken into consideration. Issues such as the identity of the transmitter, identity of the receiver, integrity of the transmittied information, etc., would arise. For more insight on the topic, one could refer to [16] and [79].

- The scalability of the system needs to be tested, i.e. the impact that an increase in the number vehicles, traffic boards, license plates, etc., would have on the performance of the system. For this purpose, one could use the commonly used sequential simulator "Network Simulator (NS-2)" [1]. However, students in our PARADISE lab designed a scalable parallel simulation testbed "SWiMNet" [10] [11], which could prove to be more helpful in testing the scalabiltiy of the system.

- The idea of integrating GPS into the system for navigation and tracking purposes should be explored [26].

[1] http://www.isi.edu/nsnam/ns/

References

[1] O. Abumansoor and A. Boukerche. A secure cooperative approach for nonline-of-sight location verification in vanet. *IEEE Transactions on Vehicular Technology*, 61:275–285, January 2012.

[2] M. A. Aizerman, E. A. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, pages 821–837, January 1964.

[3] S. A. Angadi and M. M. Kodabagi. Text region extraction from low resolution natural scene images using texture features. *IEEE 2nd International on Advance Computing Conference, 2010 (IACC)*, pages 121–128, February 2010.

[4] A. Bamis, A. Boukerche, I. Chatzigiannakis, and S. Nikoletseas. A mobility aware protocol synthesis for efficient routing in ad hoc mobile networks. *Computer Networks*, 52:130–154, January 2008.

[5] R. B. Batista, A. Boukerche, and A. C. M. A. de Melo. A parallel strategy for biological sequence alignment in restricted memory space. *Journal of Parallel and Distributed Computing*, 68:548–561, April 2008.

[6] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, 1992.

[7] A. Boukerche. *Handbook of Algorithms for Wireless Networking and Mobile Computing*. Chapman and Hall/CRC, 2005.

[8] A. Boukerche, X. Cheng, and J. Linus. Energy-aware data-centric routing in microsensor networks. *Proceedings of the 6th ACM International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWIM)*, 13:42–49, September 2003.

[9] A. Boukerche and S. K. Das. Dynamic load balancing strategies for conservative parallel simulations. *11th Workshop on Parallel and Distributed Simulation, 1997*, pages 20–28, June 1997.

[10] A. Boukerche, S. K. Das, and A. Fabbri. Swimnet: a scalable parallel simulation testbed for wireless and mobile networks. *Wireless Networks*, 7:467–486, September 2001.

[11] A. Boukerche, S. K. Das, A. Fabbri, and O. Yildiz. Exploiting model independence for parallel pcs network simulation. *13th Workshop on Parallel and Distributed Simulation, 1999*, pages 166–173, May 1999.

[12] A. Boukerche and X. Fei. A coverage-preserving scheme for wireless sensor network with irregular sensing range. *Ad Hoc Networks*, 5:1303–1316, November 2007.

[13] A. Boukerche, S. Hong, and T. Jacob. A distributed algorithm for dynamic channel allocation. *Mobile Networks and Applications*, 7:115–126, April 2002.

[14] A. Boukerche, S. Hong, and T. Jacob. An efficient synchronization scheme of multimedia streams in wireless and mobile systems. *IEEE Transcations on Parallel and Distributed Systems*, 13:911–923, September 2002.

[15] A. Boukerche, K. R. L. Juca, J. B. Sobral, and M. S. M. A. Notare. An artificial immune based intrusion detection model for computer and telecommunication systems. *Parallel Computing*, 30:629–646, June 2004.

[16] A. Boukerche and X. Li. An agent-based trust and reputation management scheme for wireless sensor networks. *IEEE Global Telecommunications Conference, 2005 (GLOBECOM)*, 3:1857–1861, November 2005.

[17] A. Boukerche and M. S. M. A. Notare. Behavior-based intrusion detection in mobile phone systems. *Journal of Parallel and Distributed Computing*, 62:1476–1490, September 2002.

[18] A. Boukerche, H. A. B Oliveira, E. F. Nakamura, and A. A. F. Loureiro. Secure localization algorithms for wireless sensor networks. *IEEE Communications Magazine*, 46:96–101, April 2008.

[19] A. Boukerche, R. W. N. Pazzi, and R. B. Araujo. A fault-tolerant wireless sensor network routing protocols for the supervision of context-aware physical environments. *Journal of Parallel and Distributed Computing*, 66:586599, April 2006.

[20] A. Boukerche and C. Tropper. A distributed graph algorithm for the detection of local cycles and knots. *IEEE Transactions on Parallel and Distributed Systems*, 9:748–757, August 1998.

[21] A. Boukerche, B. Turgut, N. Aydin, M. Z. Ahmad, L. Boloni, and D. Turgut. Routing protocols in ad hoc networks: A survey. *Computer Networks*, 55:30323080, September 2011.

[22] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:1124–1137, September 2004.

[23] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, September 1995.

[24] D. J. Crandall, S. Antani, and R. Kasturi. Extraction of special effects caption text events from digital video. *International Journal on Document Analysis and Recognition (IJDAR)*, 5:138–157, September 2003.

[25] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005 (CVPR)*, 1:886–893, June 2005.

[26] H. A. B. F. de Oliveira, A. Boukerche, E. F. Nakamura, and A. A. F. Loureiro. An efficient directed localization recursion protocol for wireless sensor networks. *IEEE Transactions on Computers*, 58:677–691, May 2009.

[27] M. B. Dillencourt, H. Samet, and M. Tamminen. A general approach to connected-component labeling for arbitrary image representations. *Journal of the ACM*, 39:253–280, April 1992.

[28] K-B. Duan and S. Sathiya Keerthi. Which is the best multiclass svm method? an empirical study. *Proceedings of the 6th International Workshop on Multiple Classifier Systems*, pages 278–285, 2005.

[29] M. Elhadef, A. Boukerche, and H. Elkadiki. A distributed fault identification protocol for wireless and mobile ad hoc networks. *Journal of Parallel and Distributed Computing*, 68:321335, March 2008.

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

[31] J. Fabrizio, M. Cord, and B. Marcotegui. Text extraction from street level images. *ISPRS Workshop on City Models, Roads and Traffic (CMRT)*, 38:199–204, September 2009.

[32] B. Gatos, I. Pratikakis, and S. J. Perantonis. Text detection in indoor/outdoor scene images. August 2005.

[33] B. Gatos, I. Pratikakis, and S. J. Perantonis. Adaptive degraded document image binarization. *Pattern Recognition*, 39:317–327, March 2006.

[34] A. Gonzalez, L.M. Bergasa, and J.J. Yebes. Text detection and recognition on traffic panels from street-level imagery using visual appearance. *IEEE Transactions on Intelligent Transportation Systems*, 15:228–238, February 2014.

[35] J. Greenhalgh and M. Mirmehdi. Real-time detection and recognition of road traffic signs. *IEEE Transactions on Intelligent Transportation Systems*, 13:1498–1506, August 2012.

[36] R. M. Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67:786–804, May 1979.

[37] R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *Systems, Man and Cybernetics, IEEE Transactions on*, pages 610–621, November 1973.

[38] P. Howarth and S. Ruger. Evaluation of texture features for content-based image retrieval. *Proceedings of the International Conference on Image and Video Retrieval*, 2004.

[39] https://www.ualberta.ca/ ccwj/teaching/.

[40] D. Karatzas, S.R. Mestre, J. Mas, F. Nourbakhsh, and P.P. Roy. Icdar 2011 robust reading competition - challenge 1: Reading text in born-digital images (web and email). *International Conference on Document Analysis and Recognition, 2011 (ICDAR)*, pages 1485–1490, September 2011.

[41] K. I. Kim, K. Jung, and J. H. Kim. Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:1631–1639, December 2003.

[42] Il K. Koo and D. H. Kim. Scene text detection via connected component clustering and nontext filtering. *IEEE Transactions on Image Processing*, 22:2296–2305, June 2013.

[43] S. Kumar and M. Hebert. Discriminative random fields: a discriminative framework for contextual interaction in classification. *9th IEEE International Conference on Computer Vision, 2003*, 2:1150–1157, October 2003.

[44] Breimanm L. Random forests. *Machine Learning*, pages 5–32, October 2001.

[45] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *18th International Conference on Machine Learning (ICML)*, pages 282–289, June 2001.

[46] H. C. Lee and D.R. Cok. Detecting boundaries in a vector field. *IEEE Transactions on Signal Processing*, 39:1181–1194, May 1991.

[47] G. Ling and D. W. Jacobs. Shape classification using the inner-distance. *IEEE Transaction Pattern Analysis Machine Intelligence*, 29:286–299, February 2007.

[48] C. Liu, C. Wang, and R. Dai. Text detection in images based on unsupervised classification of edge-based features. *International Conference on Document Analysis ad Recognition (ICDAR)*, pages 610–614, August 2005.

[49] Y. Liu, S. Goto, and T. Ikenaga. A contour-based robust algorithm for text detection in color images. *Transactions on Information and Systems*, 89:1221–1230, March 2006.

[50] Z. Liu and S. Sarkar. Robust outdoor text detection using text intensity and shape features. *19th International Conference on Pattern Recognition (ICPR)*, pages 1–4, March 2008.

[51] D.G. Lowe. Object recognition from local scale-invariant features. *The Proceedings of the 7th IEEE International Conference on Computer Vision, 1999.*, 2:1150–1157, September 1999.

[52] S. Lu and K. E. Barner. Weighted dct coefficient based text detection. *International Conference on Acoustics, Speech and Signal Processing, 2008 (ICASSP)*, pages 1341–1344, March 2008.

[53] S.M. Lucas. Icdar 2005 text locating competition results. *8th International Conference on Document Analysis and Recognition, 2005*, 1:80–84, August 2005.

[54] Ronald Lumia, Linda G. Shapiro, and Oscar A. Zuniga. A new connected components algorithm for virtual memory computers. *Computer Vision, Graphics, and Image Processing*, 22:287–300, May 1983.

[55] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *International Journal of Computer Vision*, 43:7–27, June 2001.

[56] A. Mammeri, A. Boukerche, J. Feng, and R. Wang. North-american speed limit sign detection and recognition for smart cars. *38th IEEE Conference on Local Computer Networks Workshops, 2013 (LCN Workshops)*, pages 154–161, October 2013.

[57] A. Mammeri, A. Boukerche, and M. Zhao. Keypoint-based binocular distance measurement for pedestrian detection system. *Proceedings of the 4th ACM International Symposium on Development and Analysis of Intelligent Vehicular Networks and Applications, 2014 (DIVAnet)*, pages 9–15, September 2014.

[58] A. Mammeri, E-H. Khiari, and A. Boukerche. Road-sign text recognition architecture for intelligent transportation systems. *80th IEEE Vehicular Technology Conference, 2014 (VTC Fall)*, pages 1–5, September 2014.

[59] A. Mammeri, D. Zhou, A. Boukerche, and M. Almulla. An efficient animal detection system for smart cars using cascaded classifiers. *IEEE Conference on Communications, 2014 (ICC)*, pages 1854–1859, June 2014.

[60] A. Mammeri, T. Zou, and A. Boukerche. Lane detection and tracking system based on the mser algorithm, hough transform and kalman f-factorilter. *Proceedings of the 17th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, 2014 (MSWiM)*, pages 259–266, September 2014.

[61] A. Mammeri, T. Zuo, and A. Boukerche. Extending the detection range of vision-based driver assistance systems application to pedestrian protection system. *IEEE Global Communications Conference, 2014 (GLOBECOM)*, pages 1358–1363, December 2014.

[62] V.N. Manjunath Aradhya, M. S. Pavithra, and C. Naveena. A robust multilingual text detection approach based on transforms and wavelet entropy. *Procedia Technology*, 4:232–237, February 2012.

[63] V. Y. Mariano and R. Kasturi. Locating uniform-colored text in video frames. *15th International Conference on Pattern Recognition(ICPR)*, 4:539–542, October 2000.

[64] O. Marques. *Practical image and video processing using MATLAB*. Wiley-IEEE Press, 2011.

[65] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki. The det curve in assessment of detection task performance. pages 1895–1898, 1997.

[66] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *British Machine Vision Conference*, pages 384–393, 2002.

[67] R. Minetto, N. Thome, M. Cord, J. Fabrizio, and B. Marcotegui. Snoopertext: A multiresolution system for text detection in complex visual scenes. *International Conference on Image Processing (ICIP)*, pages 3861–3864, September 2010.

[68] R. Minetto, N. Thome, M. Cord, N. J. Leite, and J. Stolfi. T-hog: An effective gradient-based descriptor for single line text regions. *Pattern Recognition*, 46:1078–1090, March 2013.

[69] R. Minetto, N. Thome, M. Cord, J. L. Neucimar, and J. Stolfi. Snoopertext: A text detection system for automatic indexing of urban scenes. *Computer Vision and Image Understanding*, 122:92–104, May 2014.

[70] M. Mohieddin and S. Mozaffari. Hybrid approach for farsi/arabic text detection and localisation in video frames. *IET Image Processing*, 7:154–164, March 2013.

[71] A. Mojsilovic, D. Rackov, and M. Popovic. On the selection of an optimal wavelet basis for texture characterization. *International Conference on Image Processing, 1998. (ICIP)*, 3:678–682, October 1998.

[72] Wayne Niblack. *An Introduction to Digital Image Processing*. Strandberg Publishing Company, Birkeroed, Denmark, Denmark, 1985.

[73] E. Nobuo. Text detection from natural scene images: towards a system for visually impaired persons. *International Conference on Pattern Recognition*, pages 683–686, August 2004.

[74] T. Ojala, M. Pietikainen, and D. Harwood. Performance evaluation of texture measures with classification based on kullback discrimination of distributions. *Proceedings of the 12th IAPR International Conference on Pattern Recognition, 1994. Conference A: Computer Vision and Image Processing.*, 1:582–585, October 1994.

[75] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9:62–66, 1979 1979.

[76] Y-F. Pan, x. Hou, and C-L. Liu. A robust system to detect and localize texts in natural scene images. *The Eighth IAPR International Workshop on Document Analysis Systems, 2008 (DAS'08)*, pages 35–42, September 2008.

[77] Y-F. Pan, X. Hou, and C-L. Liu. A hybrid approach to detect and localize texts in natural scene images. *IEEE Transactions on Image Processing*, 20:800–813, March 2011.

[78] D. M. W. Powers. Evaluation: From precision, recall and f-factor to roc, informedness, markedness and correlation. Technical report, AILab, CSEM, Flinders University of South Australia.

[79] Y. Ren and A. Boukerche. Modeling and managing the trust for wireless and mobile ad hoc networks. *IEEE International Conference on Communications, 2008 (ICC)*, pages 2129–2133, May 2008.

[80] T. Retornaz and B. Marcotegui. Scene text localization based on the ultimate opening. *International Symposium on Mathematical Morphology (ISMM)*, 1:177–188, 2007.

[81] S. Samarah, M. Al-Hajri, and A. Boukerche. A predictive energy-efficient technique to support object-tracking sensor networks. *IEEE Transactions on Vehicular Technology*, 60:656–663, February 2011.

[82] H. Samet and M. Tamminen. Efficient component labelling of images of arbitrary dimension represented by linear binary trees. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 10:579–586, July 1988.

[83] S. P. Sandip and S. P. Harshal. Study and review of various image texture classification methods. *International Journal of Computer Applications*, 75:33–38, August 2013.

[84] J. Serra and J.C. Simon. Toggle mappings: From pixels to features. *Elsevier*, pages 61–72, 1989.

[85] A. Shahab, F. Shafait, and A. Dengel. Icdar 2011 robust reading competition challenge 2: Reading text in scene images. *Internatoinal Conference on Document Analysis and Recognition, 2011 (ICDAR)*, pages 1491–1496, September 2011.

[86] M. H. Shehzadm and L. Prevost. Texture based text detection in natural scene images: A help to blind and visually impaired persons, 2007.

[87] C. Shi, C. Wang, B. Xiao, Y. Zhang, and S. Gao. Scene text detection using graph model built upon maximally stable extremal regions. *Pattern Recognition Letters*, 34, January 2013.

[88] P. Shivakumara, W. Huang, T. Q. Phan, and C. L. Tan. Accurate video text detection through classification of low and high contrast images. *Pattern Recognition*, 43:2165–2185, April 2010.

[89] R. Smith. An overview of the tesseract ocr engine. *9th International Conference on Document Analysis and Recognition, 2007. (ICDAR)*, 2:629–633, September 2007.

[90] J. Sochman and J. Matas. Waldboost - learning for time constrained sequential detection. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005 (CVPR)*, 2:150–156, June 2005.

[91] L. P. Sosa, S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young. Icdar 2003 robust reading competitions. *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, pages 682–687, August 2003.

[92] GN Srinivasan and G. Shobha. Statistical texture analysis. *Proceedings of World Academy of Science: Engineering and Technology*, 36:1264–1269, December 2008.

[93] K-K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:39–51, January 1998.

[94] H. Tamura, S. Mori, and T. Yamawaki. Texture features corresponding to visual perception. *IEEE Transactions on Systems, Man and Cybernetics*, 8:460–473, June 1978.

[95] M. R. Turner. Texture discrimination by gabor functions. *Biological Cybernetics*, 55:71–82, November 1986.

[96] F. Villas, A. Boukerche, H. S. Ramos, H. A. B. F. de Oliveira, R. B. de Araujo, and A. A. F. Loureiro. Drina: A lightweight and reliable routing approach for in-network aggregation in wireless sensor networks. *IEEE Transactions on Computers*, 62:676–689, January 2012.

[97] K. Wang, B. Babenko, and S. Belongie. End-to-end scene text recognition. *2011 IEEE International Conference on Computer Vision (ICCV)*, pages 1457–1464, November 2011.

[98] E. K. Wong and M. Chen. A new robust algorithm for video text extraction. *Pattern Recognition*, 36:1397–1406, June 2003.

[99] C. Yi and Y. Tian. Text string detection from natural scenes by structure-based partition and grouping. *IEEE Transactions on Image Processing*, 20:2594–2605, March 2011.

[100] H. Zhang, C. Liu, C. Yang, X. Ding, and K. Wang. An improved scene text extraction method using conditional random field and optical character recognition. *2011 International Conference on Document Analysis and Recognition (ICDAR)*, pages 708–712, 2011.

[101] H. Zhang, K. Zhao, Y-Z. Song, and J. Guo. Text extraction from natural scene image: A survey. *Advances in cognitive and ubiquitous computing Selected papers from the Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS-2012)*, 122:310 – 323, 2013.

[102] W. Zhou, H. Li, Y. Lu, and Q. Tian. Principal visual word discovery for automatic license plate detection. *IEEE Transactions on Image Processing*, 21:4269–4279, September 2012.