CHALMERS
UNIVERSITY OF TECHNOLOGY

Evaluation of traffic light detection algorithms for automated video analysis

Master's thesis in Software Engineering

MUHANAD NABEEL

DAVID USTARBOWSKI

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2016

Master's thesis 2016

Evaluation of traffic light detection algorithms for automated video analysis

MUHANAD NABEEL

DAVID USTARBOWSKI

Department of Computer Science and Engineering
Division of Software Engineering
Chalmers University of Technology
Gothenburg, Sweden 2016

Evaluation of traffic light detection algorithms
for automated video analysis
Muhanad Nabeel & David Ustarbowski

Supervisor: Christian Berger, Department of Computer Science and Engineering
Supervisor: Jordanka Kovaceva, SAFER – Vehicle and Traffic Safety Centre at Chalmers
Examiner: Miroslaw Staron, Department of Computer Science and Engineering

Master's Thesis 2016
Department of Computer Science and Engineering
Division of Software Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Evaluation of traffic light detection algorithmsfor automated video analysis
MUHANAD NABEEL
DAVID USTARBOWSKI
Department of Computer Science and Engineering
Chalmers University of Technology

Abstract

Vehicle and Traffic Safety is a growingly important research topic among the automotive industry and academia. Being able to analyse driving behaviour and collecting data is key for gaining understanding about potential risks affecting traffic safety. Traffic lights are important in terms of traffic safety, therefore it is of importance to have a solution to detect them without having to spend time to find their occurrence manually in a video analysis. The goal of this this paper was to evaluate a traffic light detection algorithm for automated video analysis. The study was conducted as a case study with a quantitative research method, and present an evaluation of the implemented algorithm. The implemented algorithm is benchmarked and evaluated on a dataset exceeding one million frames coming from videos of naturalistic driving in different conditions. The result of this study covers an evaluation of the algorithm based on the benchmark. This study concluded that using Haar feature-based cascade classifiers for traffic light detection is a suitable method if some trade-offs can be made. This paper also presents recommendations for developers facing similar problems in terms of automated detection of objects connected to the real world. The process of designing and creating a solution for an automated video analysis is emphasized in a top-down approach, giving an insight for developers facing similar challenges.

Keywords: Traffic light detection, traffic light recognition, object detection, object detection algorithms, Viola–Jones object detection framework, Haar-like features, cascade classifiers, computer vision, euroFOT, big data.

Acknowledgements

We would like to thank our supervisors Christian Berger and Jordanka Kovaceva for their dedication and support during the thesis project. We would also like to thank the case company SAFER, the Vehicle and Traffic Safety Centre at Chalmers University, for providing us access to the video data of the euroFOT project and allowing us to use their facilities.

Muhanad Nabeel & David Ustarbowski, Gothenburg, August 2016

Contents

1 Introduction

With the vast interest of traffic safety research, the request for solutions concerning the area has been increasing in the past few years [1]. The traffic safety topic is attracting more researchers to collaborate with the automotive industry to find solutions for a safer traffic environment. Those solutions can vary depending on the sensors equipped in the vehicles. One of those sensors is a video camera equipped in the vehicles, which enables the vehicles to detect objects in different traffic environments in order to avoid traffic accidents. One of those solutions is creating a robust traffic light detection and recognition, which is an essential task for autonomous driving in traffic environments and to behave safely in various traffic situations.

The task of detecting traffic lights can be difficult due to the complexity of the roads and the vast amount of information that have to be extracted to comprehend by a visual based solution. To reduce the complexity of the task of recognizing traffic lights, several studies focus on specific scenarios where the traffic lights are suspended [1, 2]. The results of those studies are promising; they are evaluated on scenarios that are easy to comprehend in terms of computer vision. These scenarios however do not often occur in Swedish traffic environment where traffic lights are mostly not suspended.

While there are already a couple of commercial vision-based systems available, the research in this area is still ongoing [3, 4]. Various previous studies of traffic light detection focus mostly on recordings with color to detect and recognize the features of a traffic light. The authors of existing studies often argue about different approaches for traffic light detection, such as image processing methods or machine learning processes. Both of these methods have their own trade-offs and present different results. The existing solutions around this topic could not be narrowed down to one, due to the nature of the environment. One solution presented for the French traffic environment gave 94% in precision and 63% in recall [2], the same solution was applied to a Chinese traffic environment and achieved 96% in precision and 39% recall [2]. This shows that the environment has a big impact on the algorithms in terms of measuring the performance. The knowledge from the previous studies was taken into consideration when conducting this study.

The goal of this research was to evaluate an object detection algorithm by approaching the problem with a solution that contributes to software engineering. To tackle this problem we aim to benchmark the chosen algorithm by using a large amount of video data and measure the performance in terms of precision and recall in order to identify the limitations. In order to evaluate the robustness of those solutions using traffic light detection and recognition algorithms, we aim for two approaches. The first approach is the machine learning to detect the traffic lights, combined with a image processing approach to recognize traffic lights. The second approach is to evaluate the algorithm with an automated solution which will serve as a benchmarking environment for the first approach, with a focus on methods within software engineering. The euroFOT [5] project has collected more than 30 terabytes of video recordings from naturalistic driving; those recordings will help to evaluate the algorithm used in this research.

This paper is conducted as a case study in collaboration with SAFER - Vehicle and Traffic Safety Centre at Chalmers [6]. The case-company provided us with access to the euroFOT database [5]. Their interest of this study concerned an evaluation of solutions for detecting, recognizing and counting traffic lights from video recordings. Furthermore, creating an automated video content analysis would serve as a tool for the researchers at SAFER to extract information regarding traffic lights.

The data collected by the euroFOT project [5] was the main data source for this study. These videos are in grayscale which means that there was a need for appropriate solutions regarding the limitation that comes with grayscale videos. One of the main problems that arise when working with grayscale videos is the lack of colors that otherwise could be used in order to detect traffic lights and to determine their status (such as extracting a red circle, which represents stop). This makes the problem we are facing more complex and adds uniqueness to the research compared to previous researches. Furthermore, this paper will aim to give recommendations and guidelines to Software Engineers facing similar problems in terms of machine learning and object detection through computer vision.

1.1 Problem Domain & Motivation

This case study aims to investigate the different parameters that could have a potential effect on an object detection algorithm in terms of precision and recall. The chosen algorithm for detecting traffic lights in this case, is still under study and a complete solution is not yet achieved in the domain of computer vision. The reason is due to the different complex traffic environments in the real world.

In this particular research, the aim is to evaluate an object detection algorithm capable of detecting traffic lights from a large video set. The video set consists of grayscale video recordings of naturalistic driving. These facts need to be taking into consideration while implementing the solution for the video content analysis, due to the limited color information in grayscale videos. The motivation for this study is concerned around two factors, the first factor can be referred to the previous studies made within the topic. There is clearly a gap regarding the evaluation of the Viola-Jones object detection algorithm when it comes to videos recorded in different environments with different scenarios and weather conditions. The second factor, or motivation, is connected to the available euroFOT [5] data consisting of a large amount of naturalistic driving videos, making it possible to test and evaluate the object detection algorithm on a large amount of video frames.

1.2 Research Goal & Research Questions

The goal of this research is to study and identify the parameters, which have an impact on the chosen object detection algorithm in terms of Precision and Recall through an automated evaluation. This study also seeks to identify the functional and non-functional parameters of an automation, which includes the machine learning (ML) approach. The research will also provide recommendations and guidelines for the software engineering methods in the domain of machine learning and computer vision. This study aims to answer the following research questions:

RQ1: Which parameters would influence the traffic light detection algorithm.
RQ2: Which functional and non-functional parameters would influence an automated evaluation in a benchmarking environment for video analysis?

Answering the research questions will provide knowledge on which parameters could influence the object detection algorithm for detecting traffic lights. Furthermore, answering the research questions aims to give a perspective on the potential solutions for a software engineering approach in terms of an automated video content analysis.

1.3 Contributions

The results from this study give an insight on the evaluated object detection algorithm for traffic light detection. The information gathered can be of a value for software engineers who deal with the same problem in the area, in terms of creating solutions in form of automation for video content analysis. The solutions made were for grayscale video recordings derived from the euroFOT project [5]. A solution for grayscale videos is not preferred due to the limited information of colors, which is useful in image processing to efficiently distinguish between unique features of an object. The grayscale data have a big benefit in terms of gathering large amount of data and be stored without adding huge costs for the customer. Therefore a solution for this kind of data could cut on financial expenses for a customer. Another contribution is using an open source algorithm, which gives accurate results, instead of using other proprietary off-the-shelf software. These two contributions could benefit companies dealing with big data and strive for efficient and low cost solutions.

1.4 Scope

The scope of this research is within the domain of computer vision, software engineering and machine learning. This study is conducted in order to evaluate an object detection algorithm for traffic lights. The results were planned to be divided into two parts, the first part measuring the Precision and Recall in different weather conditions presented in the video recordings. The second part aims on evaluating various parameters that affect the detection rate. The algorithm will be evaluated in several complex traffic environments, with different scenarios from a large set of video recordings provided by the case company. The study is crucial for identifying a potential ML-algorithm for detecting traffic lights while handling a large amount of grayscale videos recorded in different weather conditions. The contribution of this study is valid on grayscale video recordings including traffic lights of different designs. Furthermore, the study also aims to provide recommendations for software engineers facing the challenges of object detection in various environments.

2 Background & Related Work

This section introduces relevant background information on automated video content analysis, object detection algorithms and theories related to traffic light detection.

2.1 Video Content Analysis

Video content analysis is a growing area of research within computer vision [7]. The challenges in this area are connected to the large amount of data that has be handled in order to make correct assumptions. Furthermore, advanced techniques are needed for categorizing, learning, structuring and analysing the video data [7, 8]. According to [7], "The challenge is to discover and interpret tiny fractions of useful information in a whirlwind of meaningless noise."

Different video solutions, such as surveillance systems etc. collects a large amount of data in terms of video recordings, therefore there is a need of an automation process in order to save time by not having to go through all the data manually [7, 8].

2.2 euroFOT

European Field Operational Test (euroFOT) was a project intended for reducing environmental impact and increase safety through evaluation of existing technologies for driver assistance systems [5]. The project achieved to find links between the driver behaviour and the existing technologies, fuel efficiency and traffic safety [5].

Driver behaviour is to a large extent the reason for traffic accidents in the European Union, which was one of the reasons for the euroFOT project to research into driver assistance technologies for understanding those behaviours [5]. The project launched over one thousand cars and trucks, which included those driver assistance systems [5]. The vehicle's movements and locations were tracked and recorded for the project to study driver behaviours. For the recordings, the cameras were located in multiple views. One view is shown in figure 2.1.

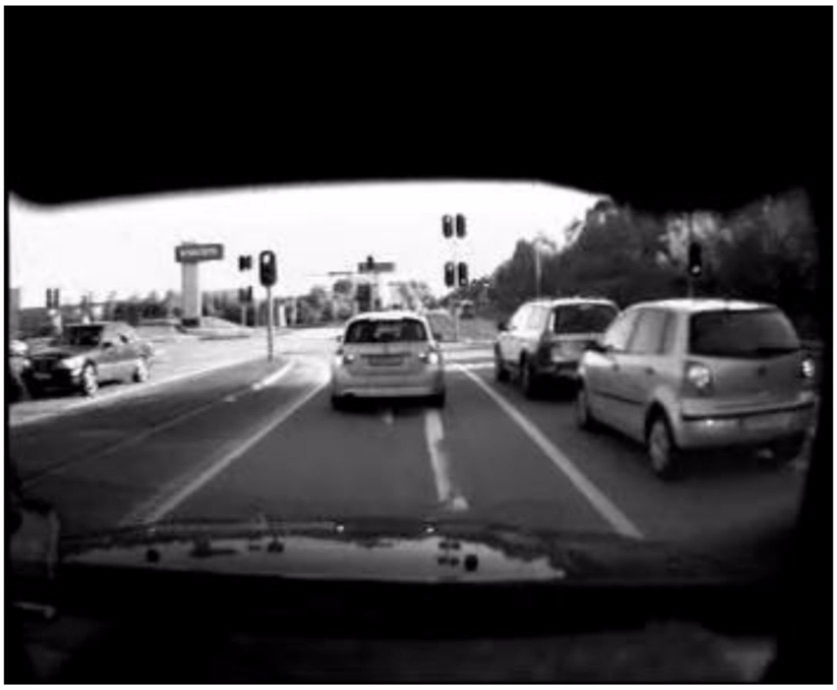


Figure 2.1: Driving video from the euroFOT project

2.3 Object Detection

Object detection is a significant topic of research related to computer vision [9, 10]. Techniques for detecting and recognizing objects have been used broadly for different systems and solutions. Such systems exist in areas of robotics, video surveillance, traffic control and medical imaging to name a few [11, 12]. The ability of detecting objects are crucial in various applications of computer vision as many of these solutions require objects to be determined in terms of presence and location [11]. Furthermore, object detection is being used for visual tracking applications, which means that the desired object is detected and tracked in one or many frames in the video, even in real time (robotic surgery) [12].

Detecting objects in images and videos is generally a challenging task. Real-world objects that are required to be detected in videos tend to have intermittent shapes making them visually differ between each other [13]. Other challenges are connected to motion and occlusion in the environment [13]. Another issue can be meeting processing requirements, in case real time has to be guaranteed [13].

Another problem that is faced in object detection is about evaluating the detector in order to gain understanding about well it is performing [14]. There are different approaches for estimating and evaluating how well a detector works, one uncomplicated but time-consuming way is manual evaluation by visual inspection. Other solutions for testing an object detector could include various performance estimations and measures, such as precision and recall [14].

2.4 Object Detection Algorithms

There are various known object detection algorithms where Feature-based methods are some of the most common. Features are calculated based on image properties and used for matching objects. The features are so called invariants which mean that they can handle image scaling and rotations [15]. An extensive amount of features can be extracted from images by using the appropriate algorithms.

2.4.1 Viola–Jones object detection framework

One well-known detection algorithm using feature-calculations is the **Viola–Jones object detection framework** presented in 2001 by Paul Viola and Michael Jones [16]. The algorithm is able to process images fast and give high detection rate. It was demonstrated and partially motivated by the task of face detection and is using a machine learning method, that also is the mechanism behind the feature selection that is performed during training [16]. The detection framework requires a process of training, where a set of positive and negative images are required [16]. Once the training is finished it returns a Cascade Classifier that can be used with a detector, referred to as Haar Cascade Detection [17].

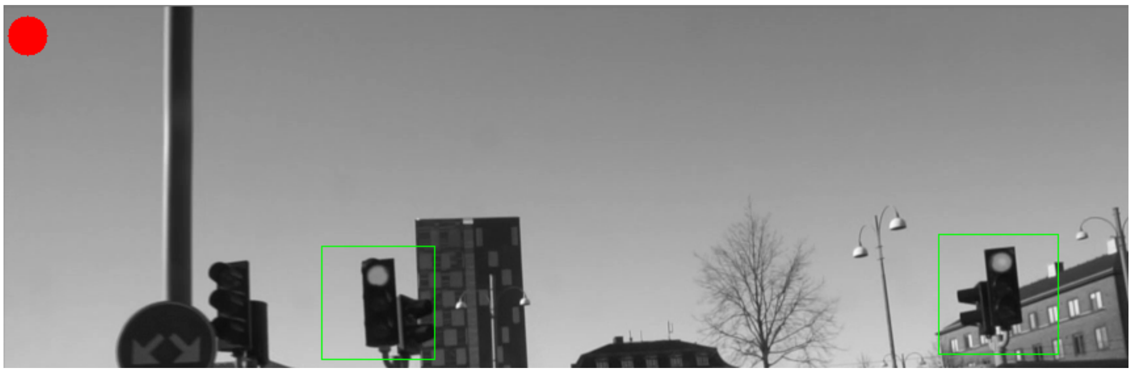


Figure 2.2: Rectangles around two traffic lights detected by the Viola–Jones object detection framework

The Viola-Jones algorithm works by finding shapes of so called Haar features. The features are basically rectangles of black and white regions. The algorithm selects the white rectangle and subtracts it with the black rectangle, this is done creating a **integral image** based on the inputted image, the integral image, also called Summed Area Table, is an effective and rapid way to calculate the sum of pixel values [16]. The purpose is to allow quick computations of any areas in a image with a few memory lookups. Once the sum of the rectangles with the white and black areas are within a threshold, the algorithm continues on a new stage with increased complexity and more features to compute [16]. The process of matching features is illustrated in figure 2.3.

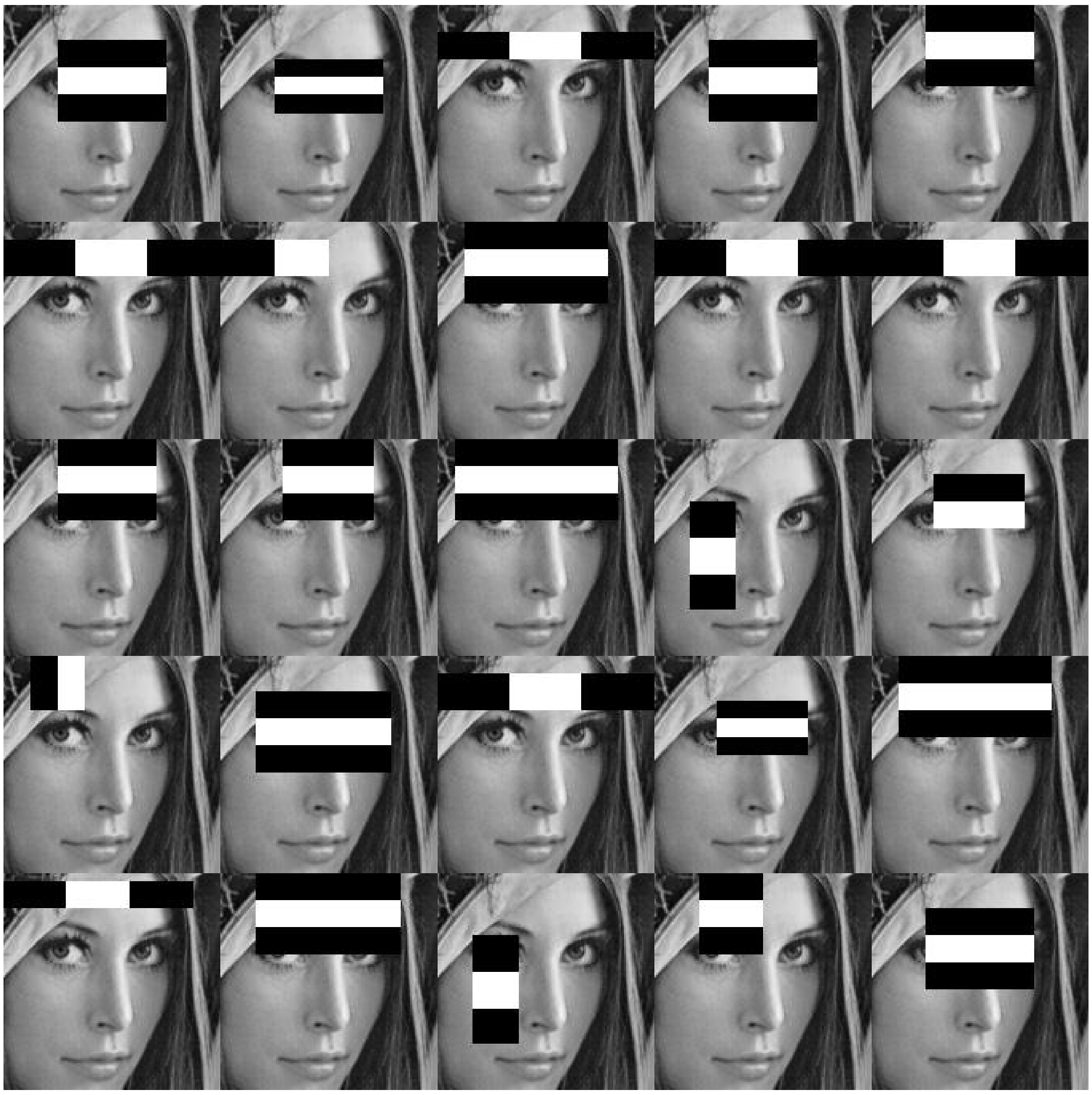


Figure 2.3: Feature matching on a face [18]

In order to detect any objects with the Viola–Jones object detection framework, a training has to be performed. The training takes positive and negative examples of images and runs a learning algorithm using AdaBoost, which accelerates the computational task [19]. OpenCV provides an application for cascade classifier training called *opencv_traincascade*, written in C++ in accordance to OpenCV 2.x API [20].
Important to point out is that the training can take days or even weeks depending on factors such as CPU clock rate, number of stages specified, boosting type and various other training parameters. Once the training is finished it will output a Haar cascade which is an XML file containing a lot of nodes with numbers that defines the Haar features.

Listing 2.1: An example of features defined inside an .xml classifier file

```
        <feature>
              <rects>
                <_>
                  11 14 8 21 -1.</_>
                <_>
                  13 14 4 21 2.</_></rects>
              <tilted>0</tilted></feature>
        <threshold>1.7821850487962365e-003</threshold>
        <left_val>-0.3374626934528351</left_val>
        <right_val>0.2435684055089951</right_val></_></_>
```

As can be seen in the listing 2.1, the example classifier contains of a node with sub-nodes. The numbers in <rects> are defining the shape of the Haar features where the first two numbers (11 14) are the coordinates of the rectangle that is to be summed using the integral image. The next two integers (8 21) are defining the length and height of the rectangle [21]. More information about how to use the algorithm, together with description of a training phase is to be found in the methodology section *3.5 Object Detection & Recognition*.

Another feature-based algorithm is **Scale-invariant feature transform (SIFT)** presented in 1999 by David Lowe. The object detection algorithm is able to detect local features in images by extracting "interest points" [15]. For detection, description is being used from a training image, as seen on figure 2.3, and then applied to another image containing the same object as in the training image [15].

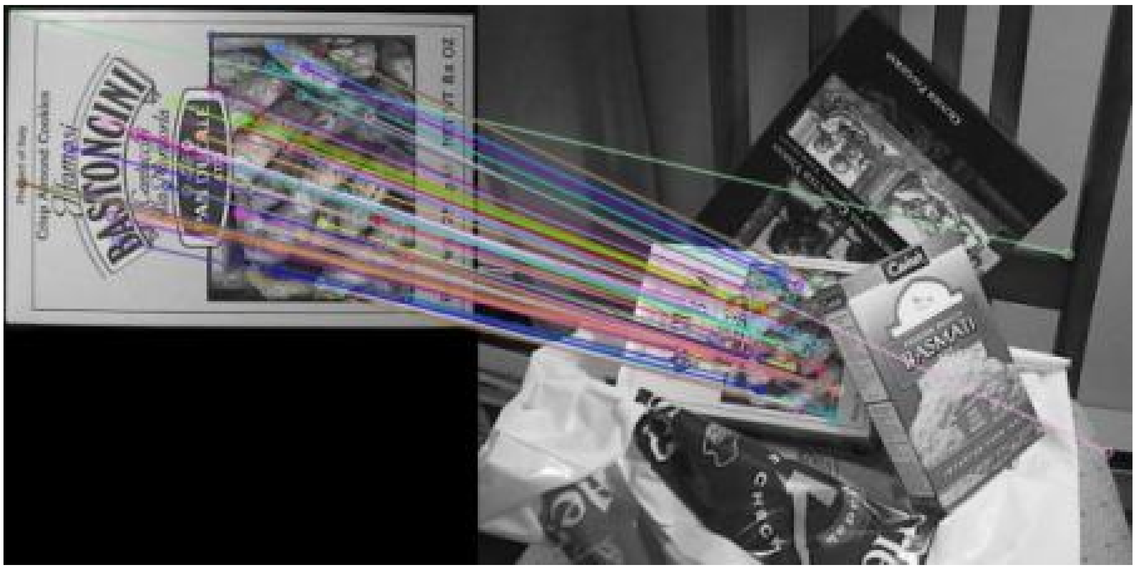


Figure 2.4: SIFT Feature Matching, the training image is to the left [18]

Speeded-Up Robust Features (SURF) is yet another algorithm used for object recognition, presented by Herbert Bay et al. in 2006. SURF is a local feature detector and descriptor with focus on fast computation without trade-off to performance. The SURF algorithm is partially similar to the SIFT algorithm although faster and more robust [22].

Circle Hough Transform [23] is a part of the original Hough Transformation algorithm [23], which operate by detecting edge points in an image. After detecting an edge point, the point is considered as a centre of a circle of radius R included in an accumulator array. When many of those circles intersect each other, they will eventually create a complete shape of a final circle as illustrated in figure 2.4. The algorithm is provided by OpenCV [18].

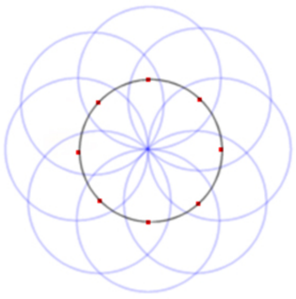


Figure 2.5: Hough Circle Transform

2.5 Traffic light detection - methods & approaches

Traffic lights are easily recognized by human vision, this is however a challenging task in terms of object detection where the results of a machine vision system is highly dependent on factors such as weather, camera type etcetera [1]. Furthermore, objects such as cars, billboards and pedestrians contribute to an increased risk of false detections/false positives [1]. A region of interest (ROI) can decrease the amount of false positives; this can be achieved by predicting the area in the frame where the traffic lights will appear [1]. Other factor that can affect the robustness of the detection is lightning conditions [24]. Detection performance tend to drop when it comes to night-time recordings, as detection systems are dependent on edge information and other methods such as thresholding and template matching the performance will get affected negatively, since the color information will decrease or in worst case vanish [1]. The same applies to recordings with occurrences of strong sunlight [1].

There are different designs of traffic lights, two main types are suspended traffic lights and supported traffic lights [2]. Suspended traffic lights are easier to recognize since the background often is static, which means that the background consists of few colors [2]. Supported traffic lights are mounted on a pole and is the standard in Sweden, as shown in figure 2.5.

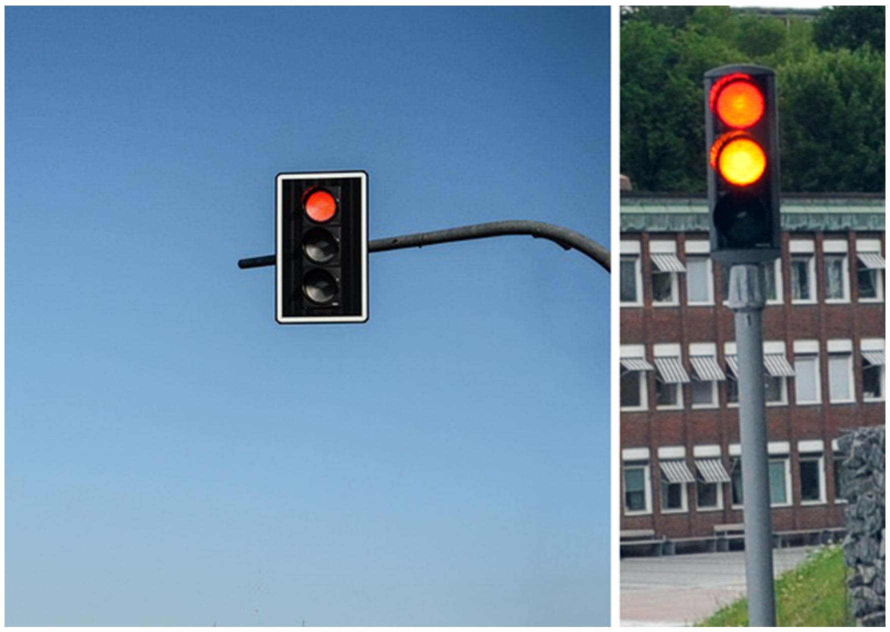
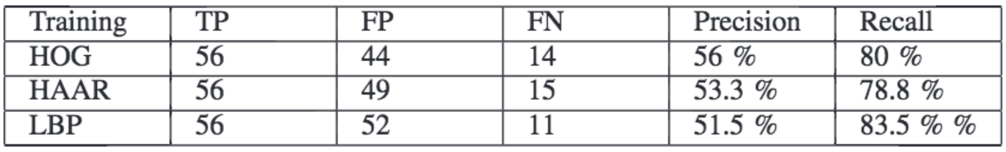


Figure 2.6: Suspended traffic light to the left, mounted Swedish traffic light to the right

There are various concepts to recognize traffic lights, for example by an learning process or by image processing. The learning process requires samples of traffic lights and non-traffic lights to teach the learning process algorithm the object it should detect. One learning process applied to traffic light detection could be the Haar algoritm [16], which often is evaluated in terms of performance and recall.

2.6 Related Work

Haar-like features has earlier been used in terms of detecting traffic lights, the authors of [25] evaluates three feature extraction methods including Haar Cascades, LBP and HOG and compare the results in terms of precision and recall. They also argue about the usage of various operations applied in order to reduce false positives and the execution time of the support vector machine (SVM) [25].



Training	TP	FP	FN	Precision	Recall
HOG	56	44	14	56 %	80 %
HAAR	56	49	15	53.3 %	78.8 %
LBP	56	52	11	51.5 %	83.5 % %

Figure 2.7: "Accuracy comparison in adverse weather conditions" [25]

According to the study, LBP showed the most promising results in terms of recognition of traffic lights with 83.5% recall [25] in adverse weather conditions. There is however a minor difference between the three methods presented, as can be seen in figure 2.4. The authors do not specify on how many frames their method was tested on more than: "At this point around 2000 images from the city and around are present, more images and video should be taken in the suburbs as well to train the algorithm also in a less populated area." [25].

Another study covering feature extraction is [26] which presents a solution to traffic light detection in daytime. Their detection method include a "potential traffic light detector", shape filter and Adaptive Multiclass Classifier [26]. The detector uses the YCbCr channel to generate binary images of green and red traffic lights [26]. Furthermore, the Adaptive Multi-class Classifier is used to define candidate regions through Haar features [26] and AdaBoost [19]. Their result is based on two sequences, the fist with a total amount of 13,723 frames and the second with 2,838 frames, giving up to 94% of detection rate for the proposed method [26].

A different approach to detecting traffic lights in real-time is presented by [2], the solution is entirely based on image processing with spotlight detection and recognition by adaptive templates. The traffic light recognition consists of three main steps, Spot Light Detection, Adaptive Template Matcher and Validation [2]. Furthermore, the authors developed various traffic light recognition systems in order to evaluate their own solution to other standard solutions [2], including trained cascade classifiers with AdaBoost and Haar features. The result of the author's own method reached up to 95% of precision: "the tests were performed using a video stream database consisting of more than 20 minutes of "useful" urban scenes sequence" [2]. The two of the total three video sequences had a total of 7,723 and 2,616 frames. The amount of frames in the last sequence was not specified [2].

The majority of the related work explored is relying on color images, furthermore the solutions are tested on a fairly small amount of frames. This is one of the distinguishing differences behind the research presented in paper and the related work. The reason behind creating a solution for grayscale video is because we cannot rely on color images as the euroFOT [5] data set is recorded in grayscale.

3
Methodology

This section introduces the approaches, methods and techniques applied in the phase of training a machine learning algorithm and designing an automated object detector in order to answer the research questions. The section is written in a top-down approach, beginning with listing requirements, describing the data collection and further taking up applied algorithms.

3.1 Functional and Non-functional Requirements

The functional and non-functional requirements were set with a requirements engineering approach, both internally with the case company that served as a product owner, and externally among us developers. The Requirement Elicitation Process [27] was conducted together with the case company SAFER - Vehicle and Traffic Safety Centre at Chalmers [6] in order to discuss and gather the expectations of the solutions that had to be provided.

The internal Requirement Elicitation was conducted through discussions and task analysis. All the internal and external requirements were placed in a System Requirements Specification document [28].

3.1.1 Internal Requirements

Functional requirement	***Description***
FR1	The detector should work with different weather conditions in video recordings, in specific sunny, cloudy, rainy and snowy.
FR2	The solution must to able to determine the state of the detected traffic light in a analysed video recording
FR3	The solution must count the total amount of traffic light stops in a analysed video recording
FR4	The solution must handle large amount of data in form of video recordings
FR5	The solution must not terminate during the process due to invalid input, in specific: handle various video inputs, at least .avi

Table 3.1: Internal functional requirements

Non-functional requirement	***Description***
NFR1	The solution must be able to run on the customer's machine.
NFR2	The solution should handle 12 frames/s without any extensive frame-dropping recording.
NFR3	The solution should provide high detection rate of the traffic lights in the video recordings.

Table 3.2: Internal non-functional requirements

3.1.2 External Requirements

The external Requirement Elicitation was conducted with aspects to Software Engineering and Machine Learning and were conducted through Domain Analysis and Observations [28].
The external functional and non-function requirements are based on the knowledge gained from previous studies. Furthermore, these requirements are conducted upon a software engineering approach and elicited with help of a Domain Analysis. Important to point out is that the solution intended to be provided should be designed as a black-box taking the needs of the product owner into consideration making the external requirements focus on both a Software Engineering and Machine Learning approach in terms of video content analysis.

Functional requirements	***Description***
FR1	The solution must be able to modify the input videos, in specific; change resolution, adjust brightness and blur.
FR2	The solution must accept different kinds of video input data, in specific .avi, .waw and .mp4
FR3	The solution must be able to handle different shapes and angles of the traffic recording.
FR4	The solutions must be able to save information about frames for validation purposes.

Table 3.3: External functional requirements

Non-functional requirement	***Description***
NFR1	The solution must conform to coding conventions in order to avoid the risks of expensive maintenance due to complexity.
NFR2	The solution must be modifiable in terms of code.
NFR3	The black-box solution must conform to set quality standards.

Table 3.4: External non-functional requirements

The internal and external non-functional requirements are directly related to the quality aspects of the solution and are discussed in the section 4.4.

3.2 Data Collection

3.2.1 Positives

The very first step into creating an object detector with a machine learning algorithm is the data collection of positive images, hereby referred to as positives. The positives are the images which represent traffic lights. The positives were photographed from the real world in form of high definition images of traffic lights. To relate to RQ1, the parameters would be the positives with visible traffic lights, positives of traffic light from different visibility angles and positives with different traffic light status (Red, Yellow, Green). These parameters are important to be taken into consideration since they may affect the results.

It was important to gather qualitative data, in terms of positives, for the algorithm training in order to extract all the unique features of a traffic light. This data is for machine-only processing to prepare the algorithm for its application to the euroFOT dataset [5]. All the positive images collected had the same resolution and were taken from different distances and angles, see figure 3.1. Other important characteristics were about to capture the different state of the traffic lights, red, yellow and green. The different shapes of the traffic lights were also included in the data collection, one shape is the rectangular traffic light and the second shape is the oval traffic light. These attributes are part of the parameters that RQ1 is intended to answer.

Once the data collection phase was done, the training of the algorithm could begin, section 3.5 reflects on the complete process of the algorithm training.

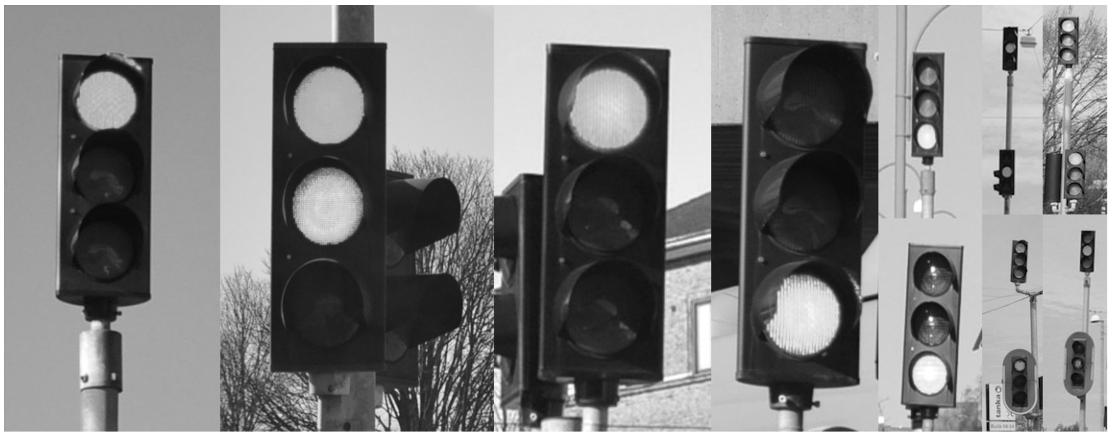


Figure 3.1: Swedish traffic lights - example of positives inputted into the training

3.2.2 Negatives

The negatives are images that not representing the object that is desired to be detected. It is a method for the algorithm to distinguish between the desired objects and anything else in the environment that shall not be detected. Those negatives could be collected from the same environment the object will appear in. The negatives collected for this research are of the same resolution as the positives mentioned in the previous paragraph.

3.3 Video Collection

The video collections used for this research was divided into two sets, the first video set was recorded by ourselves whereas the second video set was derived from the euroFOT project [5].

3.3.1 Own recordings

For the purpose of evaluating the training results from the machine learning algorithm, we recorded driving videos in central Gothenburg, Sweden, with several traffic light scenarios presented in those videos. The reason behind the recording of our own videos is due to restricted access to the complete video collection offered by the case company.

The first recording was performed in a crowded traffic environment surrounded by buildings and pedestrians as well as vehicles. Other recordings included highway driving with different vehicle speeds. The reasoning for this data collection was to test how well the detection works in different complex traffic environments.

3.3.2 euroFOT data

The video recordings from SAFER were derived from the euroFOT [5] project, as explained in the background section 2.2. Those video recordings are the data that were included in the benchmarking of the Haar algorithm. The videos consists of naturalistic driving in different weather conditions and introduces several complex traffic environments; for testing the limitations and the performance of the algorithm in terms of precision and recall, to answer RQ1.

3.4 OpenCV

OpenCV was used for applying the algorithms in this research. OpenCV is an open source library aimed for computer vision [18]. The library is written in C and C++ with bindings to other programming languages such as Python and Java. The OpenCV library contains more than 500 functions which contributes to computer vision, including camera calibration, stereo vision and robotics [18]. OpenCV does also features Machine-Learning modules, which makes it a suitable choice for this research.

3.5 Object Detection & Recognition

3.5.1 Training of the algorithm

The training of the Haar Cascade Classifier [20] was performed on the collected data mentioned in section 3.2. The first phase of the training was to identify the vectors of the positive images. In this phase the area where the objects is located in each positive image is retrieved by cropping the traffic light from the rest of the environment. The second phase into the training was to provide the negative images in a separate location from the positive images. The third phase was to provide different parameters for the Haar Classifier [20] to perform the training. The parameters are related to answer on RQ1 and are presented in table 3.5 and 3.6 below.

Positive Samples:

Arguments	***Parameters***
-num	1836
-w	32
-h	48
-bgthresh	80

Table 3.5: Parameters providing positive samples

Cascade Training:

Arguments	***Parameters***
-numPos	1836
-numNeg	3530
-numStages	27
-minHitRate	0.99500
-maxFalseAlarmRate	0.500
-weightTrimRate	0.99500
-bt	GAB - Gentle AdaBoost

Table 3.6: Cascade Training Parameters

After the training was finished the opencv_traincascade [20] provided Classifiers for each stage performed during the training, those in turn were provided to another tool from OpenCV [18] to generate a XML file with all the necessary data retrieved from the training. The training for this research took 3 weeks to complete on an Intel Core i7-4770K @ 3.50GHz with 8GB RAM.
The long training time was due to the large width and height parameters for the positive samples combined with a high number of cascade stages to be trained. The other reason is because the selection of up to 5 existing traffic lights per given positive sample, resulting in over 3500 positive objects in total.

3.5.2 Haar Feature-based Cascade Classifier for Object Detection

There are various well-known algorithms when it comes to object detection within computer vision, one of them is Object Detection using Haarlike Features proposed by Paul Viola and Michael Jones in 2001 [16]. Although the proposed algorithm originally is motivated for the task of face detection by the creators, it has been proved to suit well for detection of other objects than faces, such as vehicle registration plates, pedestrians and even traffic lights.

3.5.3 Haar features

There are many objects in the real world that share the same properties, the type of traffic light we want to detect will always consist of a pole with a black box containing three circles, at least one of the circles will always be illuminated. Haar-like features are calculated by finding these properties during the training stage [16]. Once training the algorithm, feature masks are placed on a sub-window in a specific location. There are different types of features that are consisting of dark and light regions, these are computed during the training by calculating the normalized sum of the pixels in the black areas, which then get subtracted from the normalised sum of the pixels in the white areas [29]. Once a training is finished it will output a so-called Cascade Classifier which is ready to be applied on the input data, in our case videos.

3.5.4 Detecting the traffic lights

Once the desired Cascade Classifier has been trained it can be used for detecting the desired object, this is performed with a so called detector. The detector used in this study is written in C++ with OpenCV [18] and is the core of the solution behind detecting the traffic lights. Once this detector is running, it will give outputs in case it can classify objects as traffic lights, if the output is correct it is classified as a true positive, otherwise as a false positive. Completely undetected traffic lights are classified as false negatives. Figure 3.2 shows a true positive as an example.

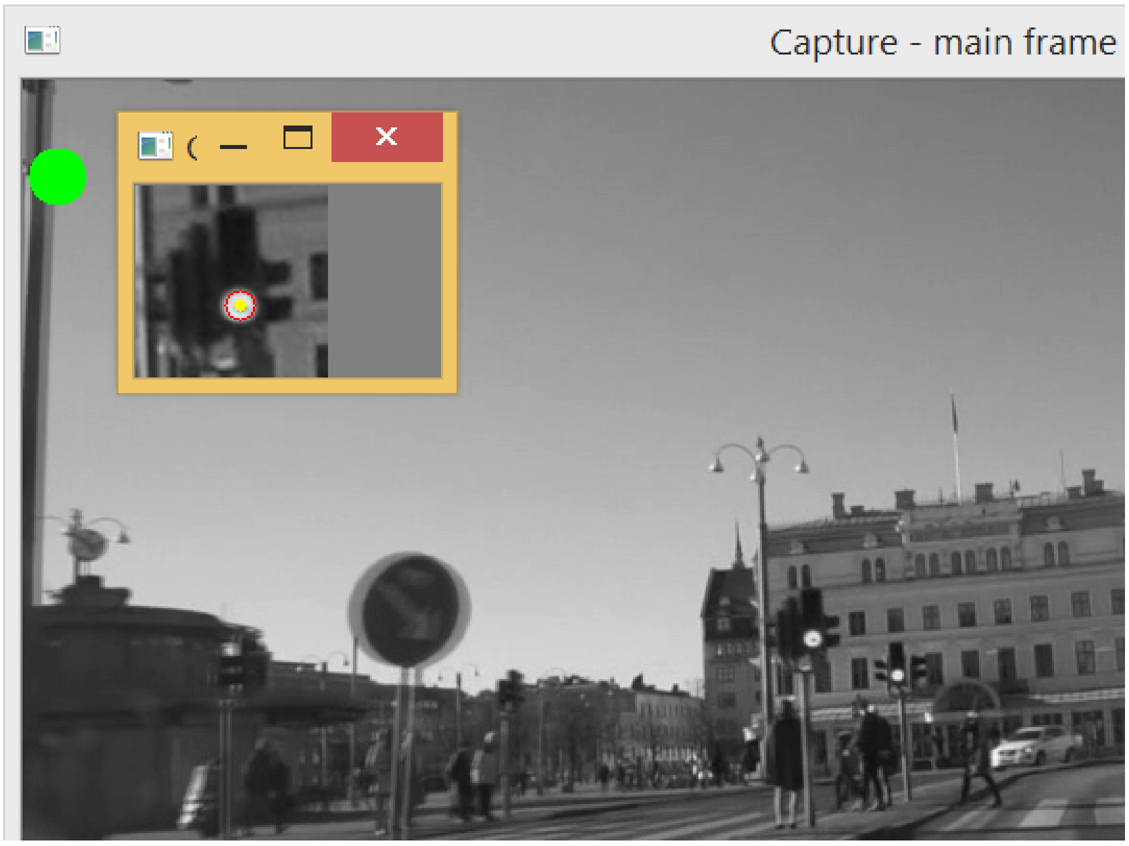


Figure 3.2: Detection giving a true positive in a frame with large displacement optical flow

3.6 Circle Detection for Object Recognition

After detection has been made through the detector, verification is done on the output; this verification is done to ensure that the output indeed is a traffic light and not something else by circle detection as shown in figure 3.4. Since all traffic lights have illuminated circles we find this being an adequate solution to reduce false positives. There can of course be outputs of false positives containing a circle; these are further decreased by the usage of a ColorMap [30]. The recolor of the grayscale output is performed in order to get the illumination from the circles in the traffic lights to appear in a specific color as shown in figure 3.4, this is done in order to further increase robustness of the verification. The flow of this process is illustrated in figure 3.3.

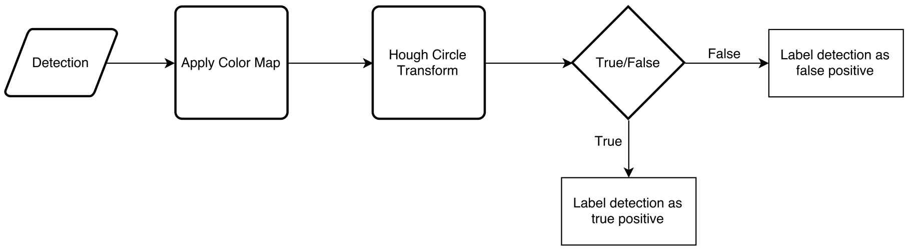

Detection
Apply Color Map
Hough Circle Transform
True/False
False
Label detection as false positive
True
Label detection as true positive

Figure 3.3: Flowchart of the circle detection

Hough Circle Transform [30] was the used algorithm for circle detection. The algorithm was set to give circle candidates even though they are not completely circular, this is due to the fact that the illumination the traffic light sometimes affect the shape of the circle making it look more like a blob than a perfect circle.

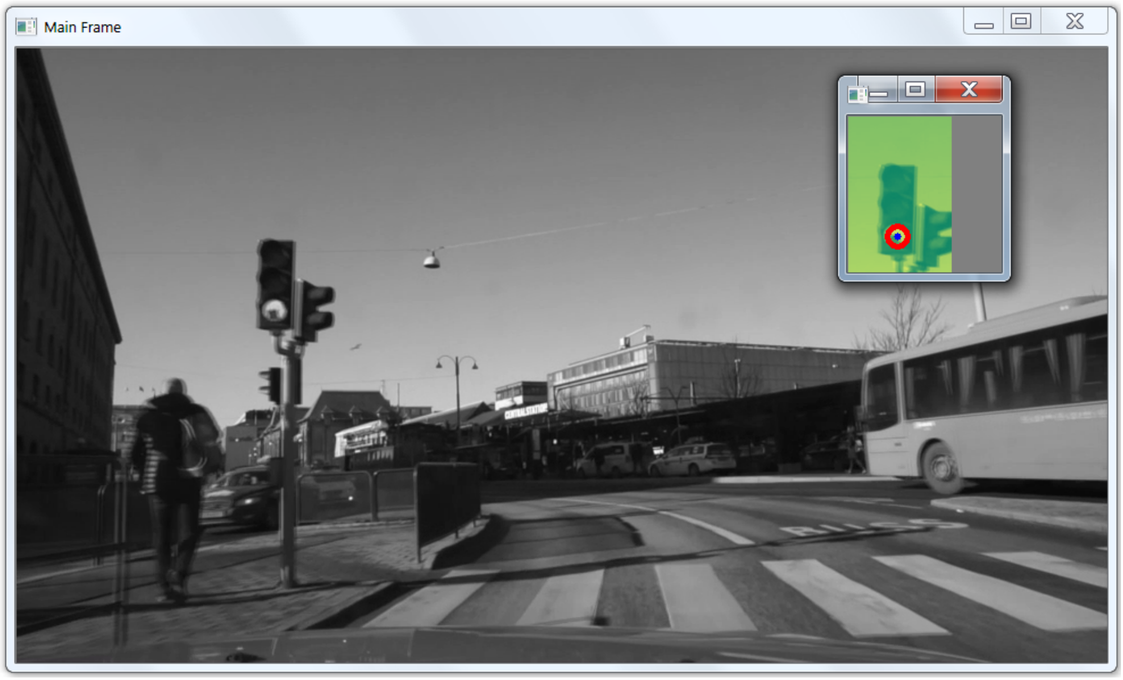


Figure 3.4: Circle detection with applied ColorMap

3.7 Saving information of previous frames

Once the first verification detected object is a traffic light has been made, another verification is performed in order to additionally increase the validity of the detection. The reason behind this solution is to avoid getting false detections in the sequence of true detections, which would affect the robustness of the state decision described in 3.8. The solution is verifying that every new detection coming in a sequence is within the same coordinates of the previous frame, these coordinates are changed iteratively once the traffic lights are moving in the frame with specific conditions disallowing the X-axis and Y-axis differ too much between frames.
Large differences are signs of false detections and are set to be avoided. This data is stored into vectors, which can be saved for further testing of the robustness behind this solution. An initial evaluation for this solution was made on our own video set with good results. Further testing of this solution would be on the complete video collection.

3.8 State Detection

State detection is the last step of the process and is performed once detection has been labeled as a true positive according to the steps described in the previous subsections.

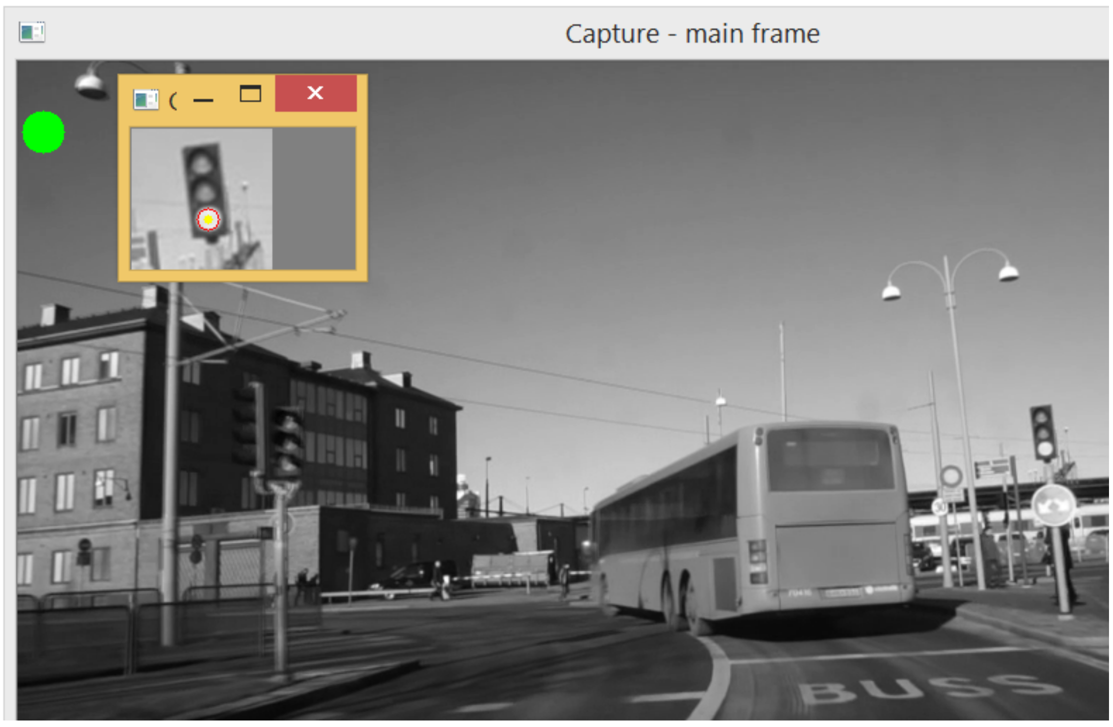


Figure 3.5: Example of the state detection in the application

The purpose of this approach is to identify which state the detected traffic light is in, the states refer to red light, yellow light and green light. The state determination is done on a labeled true detection by using the Hough Circle Transform algorithm [23]. The circle-coordinates in the region of interest served as ground for determining the state. If the location of the circle-coordinates matched the top area of the region the state was labeled as a red-light situation. If the location of the circle-coordinates was on the bottom instead, the state was labeled as a green-light situation. Yellow lights were considered as trivial for the state detection according to the specifications of this case study. The reason behind this was that the yellow light would appear in a few seconds before switching into green or red and those states would determine if the driver should stop or continue driving. An example of a green state is shown in figure 3.5 above.

3.9 Counting the traffic lights

One of the important requirements for the solution is to be able to count the amount of traffic light scenarios, this is a crucial part since the whole purpose of a video content analysis is to output relevant information and minimize the need to manually review the content. After a traffic light has been successfully detected and verified a function was implemented to count how many traffic light stops there were in each analysed video.

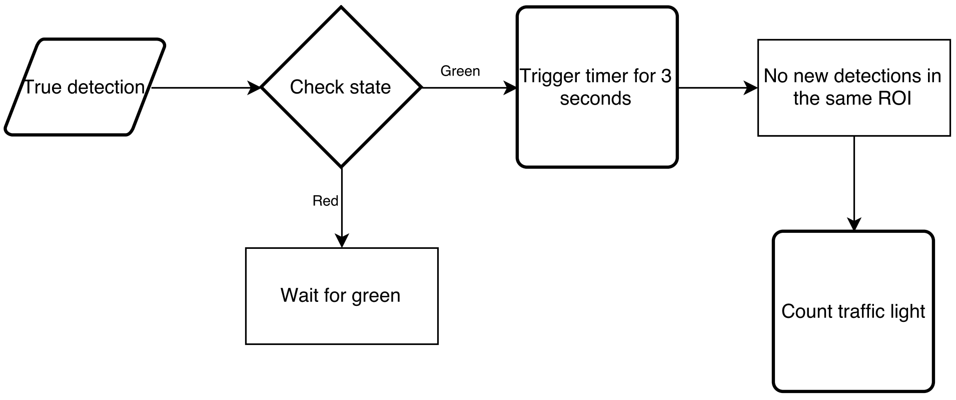


Figure 3.6: Flowchart describing how counting is performed

Every true detection will get a state assigned to it as described in section 3.8. In case the state is red no counting will be done, this is due to the vehicle is still standing still and waiting for a green state. Once the traffic light turns green a timer is triggered of 3 seconds, circle-coordinates are saved meanwhile and are checked if they exist in the same Region of Interest (ROI) of the detected traffic light, restart the timer. In case there are no more detections in the same ROI and the timer has passed 3 seconds the total traffic light counter increments and the traffic light scenario has been recorded as 1. The flow of this process is introduced in figure 3.6 above.

3.10 Automated Evaluation

Once all the methods were implemented from the previous sections, we combined them into one automation program. The automation served as an evaluation which included the Haar algorithm [16], circle detection, state detection and counting the amount of traffic lights as viewed in figure 3.7. As well as additional functional parameters to improve the detection of traffic light and circles. One of those parameters was added in the code, which would change the brightness of the frames from the input videos. Other functional parameters were directly linked to the circle detection algorithm, such as the ColorMap [30] and blur [31]. Those parameters reflect on our RQ2. The non-functional parameter was the memory consumption of the automation, which was solved through processing the input videos systematically and by applying a release function for each processed video recording.

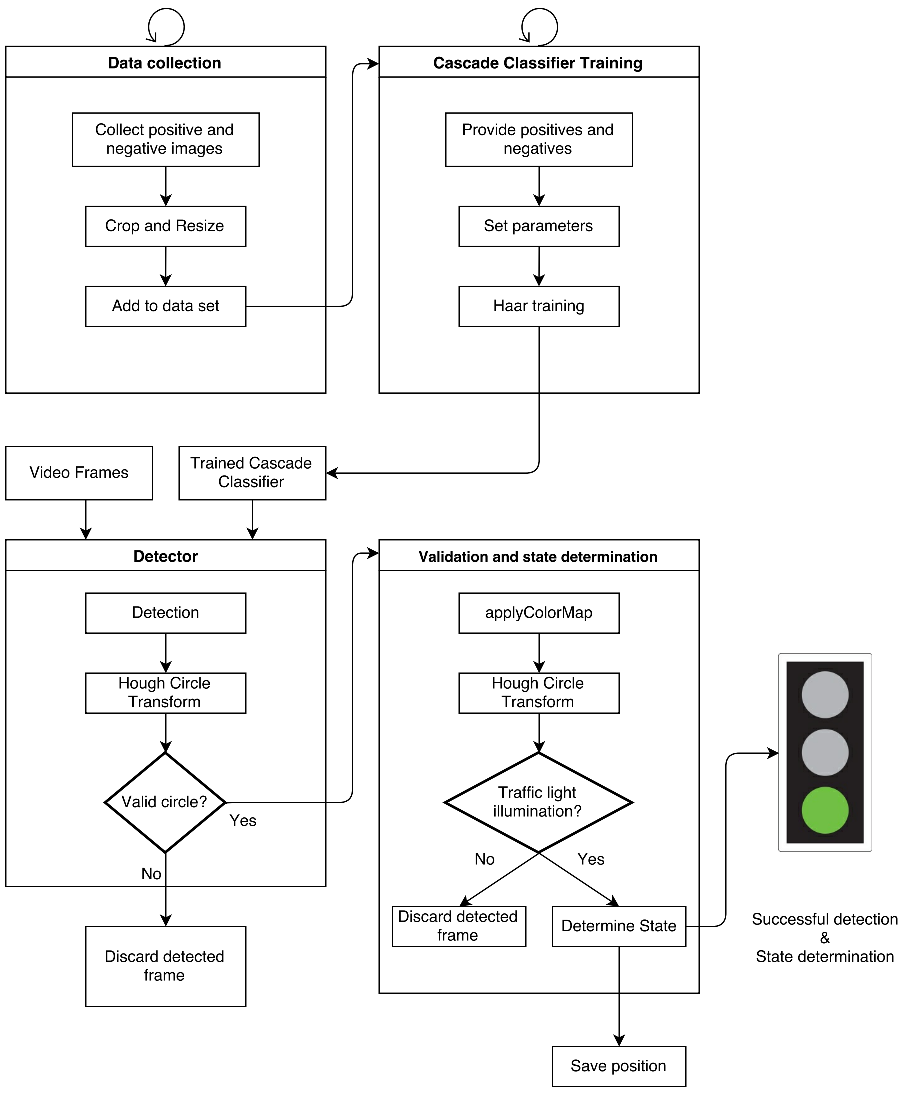


Figure 3.7: The complete solution of the automated traffic light detection

3.11 Detection in terms of traffic light scenarios

The detection rate is a measure of how many traffic lights have been detected in total with aspect to the amount of traffic light stops per analysed video; we refer to detection rate as recall. Important to point out is that every traffic light is counted as one "traffic light scenario" independent on how many lights there is for the specific driving lane. Important to point out is that our classification of false negatives differs from the other studies presented in the background chapter. The false negatives in our case refer to non-detected traffic lights at a traffic light scenario. We do not count each traffic light at the traffic light scenario, instead we count a traffic light scenario as one detection-scenario. In terms the algorithm detect a traffic light at a traffic light scenario it is counted as one successful detection for that scenario.

3.12 Selection of videos

The reason for selecting videos in beforehand and analysing the content was to ensure that the videos were appropriate with a clear view over the road and that they in fact had traffic lights in the video content. This was an approach to satisfy all the requirements set in order to fit the black-box which is discussed in section 3.13.

All the detected objects were saved into separate folders based on whether the detected object contained a circle or not. Important to point out is that all the detections were saved in order to ensure a reliable benchmarking. This procedure gives the option to count the amount of true positives and false positives detected during the benchmarking. The output were later analysed manually in order to ensure correctness of the algorithm.

Another important factor for the automation was to maintain a high quality to be able to handle big data. In this case the program should handle at least one million frames, which corresponds to over 22 hours of recorded videos. The reliability for the implemented program in this case should be high to ensure that the program executes all the selected videos without losing data or terminating during the execution.

3.13 Benchmarking & Data Analysis

The benchmarking was the most crucial process for evaluating the performance of the implemented algorithm; the results presented in this study are directly extracted from the outcomes of this process. Once the automation was finished and executed it returned an output of all the detections made, this can be referred to as a Black-box where input is given and processed, the output are including all the detections made no matter if they are false or true positives.

Input
Blackbox
Output

Figure 3.8: The black-box served as a benchmarking environment

After the automation-process finished an output was given containing the detections, this output was labeled with information about the input for enhanced verification, weather conditions etc. The output from the black-box were later manually analyzed for each of the inputted video.

This analysis included counting of how many traffic light scenarios each video contained and how many of these had a true detection, therefore it is important to point out that every single video inputted into the black-box was manually examined and every single output connected to that specific input was verified.

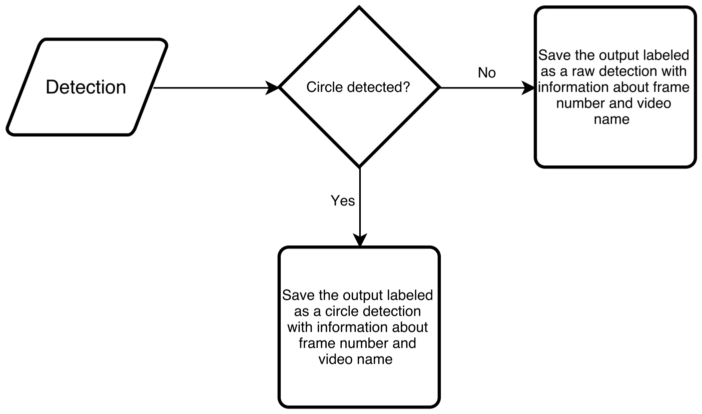

Detection
Circle detected?
No
Save the output labeled as a raw detection with information about frame number and video name
Yes
Save the output labeled as a circle detection with information about frame number and video name

Figure 3.9: Input and output of the automation process

The reason behind not having a full automation had to do with reliability. The implemented solution for the automation was simply not reliable enough to serve as a full automation for the implemented algorithms. With the manual inspection and calculations we ensure that result is valid for every video set. Therefore it is fair to say the implementation for the benchmarking was only a semi-automated evaluation, the next step would be to fully automate this process, this is argued in chapter 8.

4
Results

This section introduces the results of this study based on data from the benchmarking. The benchmarking was conducted on a total of 41 naturalistic driving videos where the inputted videos exceeded more than 22 hours of recordings, which were selected in beforehand as argued about in the selection of videos section. The visual-inspection of the total input together with the output gave us all the data required to present the results.

The results are presented and evaluated for each dataset, which consists of videos recorded in four different weather conditions. The results are categorized into weather conditions, which will contain a representation of the performance of the chosen algorithm in terms of precision and recall together with information about the total amount of frames for each category.

4.1 Research Question 1: Precision and recall

The performance of our trained classifier is measured in precision and recall, expressed in percentage. The precision gives an understanding of how well the algorithm is performing in terms of giving relevant detections.

$$Precision = \frac{tp}{tp + fp}$$

EQN 1: Precision

Relevant detections made in *aspect* to the total set of existing traffic lights in a video is measured as recall:

$$Recall = \frac{tp}{tp + fn}$$

EQN 2: Recall

4.1.1 Results from the benchmarking

This section lists the results derived from the benchmarking. The results are grouped into four categories depending on the weather conditions, including sunny, cloudy, rainy and snowy weather. The selected material consisted of naturalistic driving videos recorded in different environments, including various types of roads, areas and optical flows such as vehicles and pedestrians. It is important to mention that during the benchmarking, a increase of 50% brightness was included for the all the selected videos trough an operation in OpenCV [32]. All of the selected videos contained at least one traffic light scenario.
The total video time, total amount of frames, average precision and recall for all the videos is presented in the table 4.1 below:

Total video time	Total frames	Average precision	Average recall
22,7 hours	1,021,620	59,6%	85,4%

Table 4.1: Results calculated in average for all the weather conditions

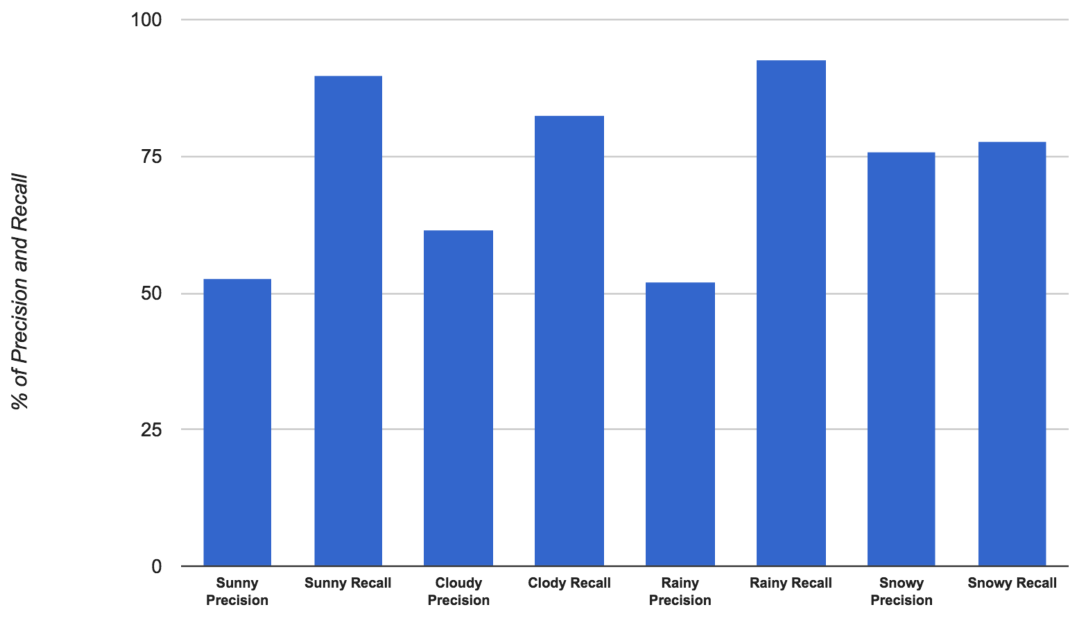


Figure 4.1: Precision and Recall for the different weather conditions

4.1.2 Sunny weather conditions

The benchmarked videos categorized under this condition are recorded in daylight with adequate lightning, which means the traffic lights are well visible throughout the video. One key factor affecting the detection rate in a negative way is the heavy sunlight that in some occasions reflects on the windscreen making it hard for the camera to capture anything on the road.

From the results of the videos with the weather condition sunny we have a total amount of *336 272 frames* and an average precision of 52,8% and average recall of 89,9%, see figure 4.1. The total time of naturalistic driving for this set was 7,4 hours. The best individual result achieved in this set gives 87% precision and 100% recall, this particular video contained *14964* frames with 10 traffic light scenarios.

The most poor individual result achieved in this set gave 55% precision and 62% recall, this particular video contained *14964 frames* with 8 traffic light scenarios. The reason behind this low detection is reflected upon in the discussion.

4.1.3 Cloudy weather condition

The benchmarked videos categorized under this condition are recorded in cloudy weather conditions. From the results of the videos with the weather condition cloudy we have a total amount of *376 140 frames* and an average precision of 61,6% and average recall of 82,5%, see figure 4.1. The total time of naturalistic driving for this set was 8,7 hours. The best individual result achieved in this set gave 93% precision and 100% recall, this particular video contained *16008 frames* with 8 traffic light scenarios.

The most poor individual result achieved in this set gives 27% precision and 50% recall, this particular video contained *18252 frames* with 6 traffic light scenarios.

4.1.4 Snowy weather condition

The benchmarked videos categorized under this condition are recorded in snowy weather conditions. One key factor affecting the detection rate in a negatively was snow hitting the windscreen in some occasions. Another factor that could contribute to lower detection and in particular decreased performance was reflections from the snow.

From the results of the videos with the weather condition snowy we have a total amount of *376 140 frames* and an average precision of 75,9% and average recall of 77,7%, see figure 4.1. The total time of naturalistic driving for this set was 3,1 hours. The best individual result achieved in this set gave 93,4% precision and 100% recall, this particular video contained *19932 frames* with 9 traffic light scenarios.

The most poor individual result achieved in this set gives 88,5% precision and 60% recall, this particular video contained *16584 frames* with 6 traffic light scenarios.

4.1.5 Rainy weather condition

The benchmarked videos categorized under this condition are recorded in rainy weather conditions. One key factor affecting the detection rate negatively was rain hitting the windscreen.

From the results of the videos with the weather condition sunny we have a total amount of *68472 frames* and an average precision of 52,1% and average recall of 92,8%, see figure 4.1. The total time of naturalistic driving for this set was 1,5 hours. The best individual result achieved in this set gives 28% precision and 100% recall, this particular video contained *25680 frames* with 8 traffic light scenarios.

The most poor individual result achieved in this set gives 75,4% precision and 85,7% recall, this particular video contained *42792 frames* with 14 traffic light scenarios.

4.2 Reflection on the measurements

The distinction between the precision and recall in the results is due to the approach of counting the false negatives which is mentioned in the methodology section, where recall tend to be higher. The explanation for this revolves around two factors. The first factor is that they are calculated in different ways, since our benchmarking is based upon traffic light scenarios we do not measure false negatives in the same manner as true positives and false negatives. Instead we are labeling every traffic light scenario and measuring whether there was a detection for this scenario. The second factor is that is about cost connected to the complexity that derives from having this large amount of frames. The potential outcome and risks of this measurement technique is argued in the threats to validity, chapter 6. Furthermore, to go in-depth and investigate the limitations of the algorithm, a third measurement could have been included and that would take the true negatives of the object detection into consideration. For that case, additional functions should be implemented to identify a true negative, and would be anything detected that does not have a traffic light. This would be verified through additional functions by processing the content of the detection. The measurement formula would be *specificity* were true negatives are divided by the total amount of negatives from the detections.

4.3 Parameters affecting detection

As reflected to in section 3.1 and to answer RQ1, one of the requirements was that the detection algorithm had to handle different states, shapes and angles of the traffic lights. In order to meet the requirement about the detector to find traffic lights at all angles accordingly, we had to find the right functional parameters affecting the problem.

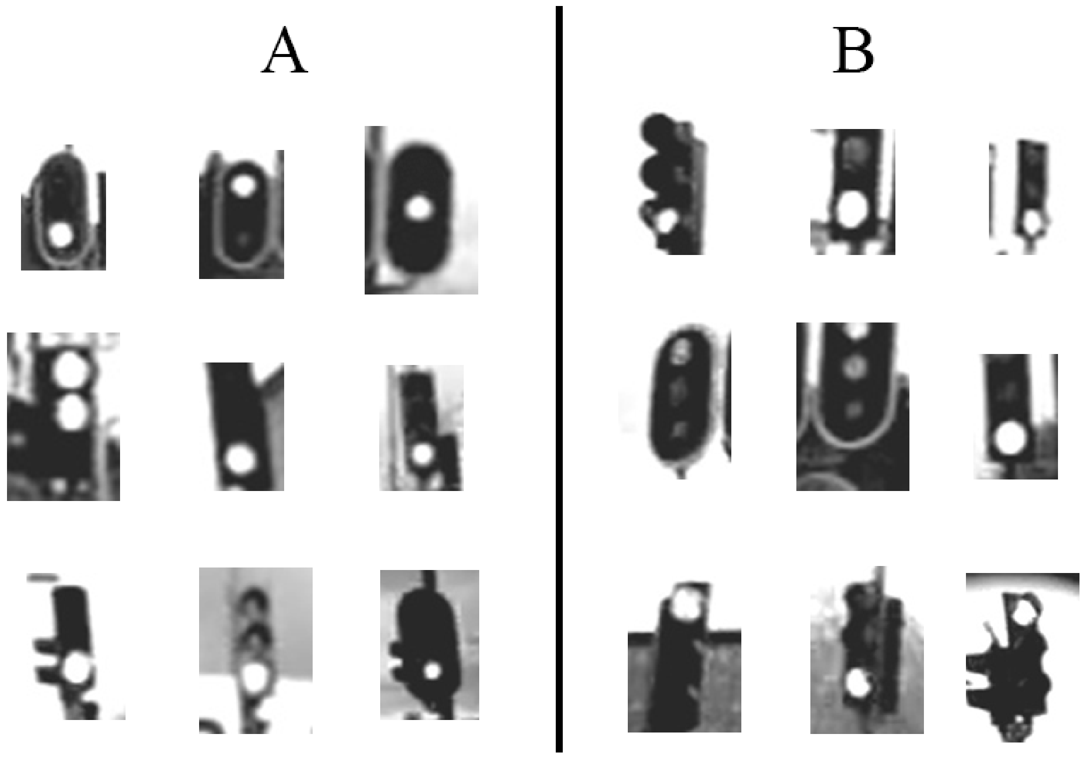


Figure 4.2: True positives detected, A. represent different states and B. different angles

As derived from the result of the benchmarking, one can determine that the internal requirement **FR2** was *strongly affected* by the functional parameter of providing positives with different states and angles to the machine learning algorithm. This can be verified due to the process of continuous training and visual testing. As can be seen on figure 4.2 the traffic light detector is able to find traffic lights in most states, shapes and angles both when the car is in motion and standing still.

4.4 Research Question 2: Automated Evaluation

This section answers the functional and non-functional parameters of an automated evaluation for ML-algorithms that had an influence on the automated video content analysis. The suggested parameters are derived from the visual inspection and experiences throughout the project. The automation developed for this research had to fulfill crucial requirements to handle large amount of video data, in this case over one million frames. The requirements were set around the ISO/IEC 25010:2011 [33].

The following characteristics were affecting the requirements:

Reliability; which serves under non-functional requirements, had to be reliable enough to input all the video recordings without termination and provide the desired outputs. To achieve the automation had to be tested with different amount of samples of video recordings, at first with a small set of videos containing a few hundred frames. The size of the video recordings were systematically increased, the last test contained up to 192,000 frames. The tests showed that the automation worked as intended without termination for larger amount of frames, this quality aspect serves under **FR4** and **FR5**.

Another requirement that the automation had to fulfill was about giving precise outputs from the detector, this quality refers to the **FR2** and **FR3**. This was tested through visual inspection [14]. The solution was also required to work reliably on the machine the customer provided; this quality refers to the **NFR1** from the requirements section 3.1. In this case the automation solution was able to run on the given machine.

Functionality; is a quality aspect for the automation to comply with any sort of video inputs from a complete data set in order to test the algorithm. One of the important functionalities was that the solution had to handle video inputs with different weather conditions and return appropriate outputs for the specific set of weather condition under test, see **FR1**. The automated was also required to handle the amount of frames/second without any extensive frame-dropping, as required by the product owner, see **NFR2**. This particular functionality was tested through OpenCV to get precise information about frames/sec under runtime and verify that no prior frame-dropping occurred. The functional parameters mentioned in the methodology section, and used in the automation during the evaluation, were blur and brightness. The increased brightness showed to give successful results in terms of detection rate. The blur was of type Gaussian blur [33].

Maintainability; is another quality aspect which was achieved through minimizing the complexity of the code, making the solution simpler to maintain. Another approach was to refactor the code as new features were developed to ensure that the code was maintainable and understandable,for other developers that might continue this study.

5 Discussion

Creating solutions for automated video content analysis requires broad domain knowledge, furthermore it is important to choose the right techniques, algorithms and methods for the intended task. The goal of this research was to provide an evaluation on object detection algorithms for traffic lights. The *first research question* was intended to answer which parameters would influence the traffic light detection algorithm. The result showed that the chosen and described parameters in this study have a great impact on the detection rate; this motivates the importance of having such parameters available since they often are very trivial. The result suggest using parameters that are specific to the environment around the object that has to be detected, it is therefore crucial to have knowledge about the the conditions in the video set that is aimed to be analysed.

Second research question aimed to answer which functional and non-functional parameters would influence an automated evaluation in a benchmarking environment for an video content analysis. To begin with, the result showed that different weather conditions have a significant impact on the precision and recall for the presented algorithm. As argued about in the result section, the quality of the recordings was affected in case of rain, snow or heavy sunlight, resulting in increased false detections and reduced detection rate. Furthermore, there showed to be various important functional and non-functional parameters influencing the automated analysis in a benchmarking environment. Those parameters were applied to the video recordings through the automation, for the algorithm to recognize the traffic light. The automation as a software consisted of other parameters regarding quality aspects and functionalities necessary for an automation, as mentioned in the result section 4.4.

The individual results per analysed video showed a visible deviation in terms of precision and recall. This is due to the low resolution of the videos, which in many cases made it very difficult to detect the traffic lights. Another factor affecting the measurements was about the driver stopping in front of the traffic light, making them partially covered. The different angles the traffic light appeared in was also a contributing factor to reduced detection, our parameter regarding the shapes and angles showed to have a positive impact on the detection, reducing this problem. Also, the speed of the vehicle the made it difficult detect the traffic lights because of the large displacement optical flow. Lastly, a factor that affected the detection was the lack of brightness in the videos, some of the traffic light appeared brighter than other traffic lights which made it difficult for the algorithms to distinguish them from the surroundings due to the video quality. This problem was partially solved by increasing the brightness and is one important functional parameter.

Reflecting on related work, this study mainly differs in three ways. To begin with, our evaluation is conducted on a video set that is much larger than presented in previous research [25, 26, 2]. In addition, the previous research cover solutions evaluated on a few video sequences, often recorded in one weather condition only. Furthermore, the solutions are evaluated on videos with driving in urban areas. Unlike the previous research, our solution is benchmarked on video sequences recorded in all kind of environments, both when it comes to weather, roads, and areas. This was possible thanks to the access to the euroFOT [5] data which included a large set of recordings of naturalistic driving. Secondly, in difference to the existing research, we include the machine learning process describing how the training of the chosen object detection algorithm was performed together with the given parameters. Lastly, this research covers the aspects of software engineering within computer vision, and in particular when it comes to automated video content analysis. We provide recommendations and guidelines based on our own experience gained by creating a complete solution for automated traffic light solution.

6
Threats to Validity

This section brings up the identified threats to validity connected to this research.

6.1 Construct validity

The algorithms used for this research were applied for all the videos included in the automation. The same parameters in terms of blur and brightness were applied on the videos, independent of the weather conditions. No changes were applied to those configurations during the benchmarking.

6.2 Internal Validity

For ensuring the validity of the results in precision and recall of the Haar detection algorithm, we execute the automation on one computer. This way we avoid a potential differences in the precision and recall of the Haar detection algorithm. Another threat is that we analysed the validity of traffic lights manually, which could oppose threats to missing traffic lights in the results. Other threat to validity is the counting of false negatives in our research compared to other papers evaluating their object detection. In our case we measure the false negatives as the amount of traffic light scenarios skipped in each video recording, other papers count the amount of traffic lights missed in total of their occurrences in a scenario with traffic lights.

6.3 External Validity

Threats to external validity could be the bias choice of the detection algorithms. Although this might be a true threat to the validity, we have analysed multiple papers that refer to the chosen algorithm and comparing other algorithms against it. The choice was based on the results presented in those papers.

7
Software Engineering & Computer Vision - Recommendations and Guidelines

The machine learning approach is often used for identifying a target with limited information. The target could be any object or shape in a computer vision perspective. Within the software domain, the targeted information can vary depending on various circumstances and could be a time consuming task to analyse. Machine learning introduces flexibility of retrieving useful information software developers seek [34, 1, 35, 36]. This section is reflecting upon software engineering within the field of computer vision and machine learning, together with guidelines and recommendations for developers facing similar challenges when comes to object detection and video content analysis.

7.1 Software Quality

The first approach into the development of a solution for this domain would be the adaptation to software quality using standards. It is important to ensure structural and functional quality and it is of importance to have a model in place, such as ISO/IEC 25010:2011 [33]. See figure 7.1 below.

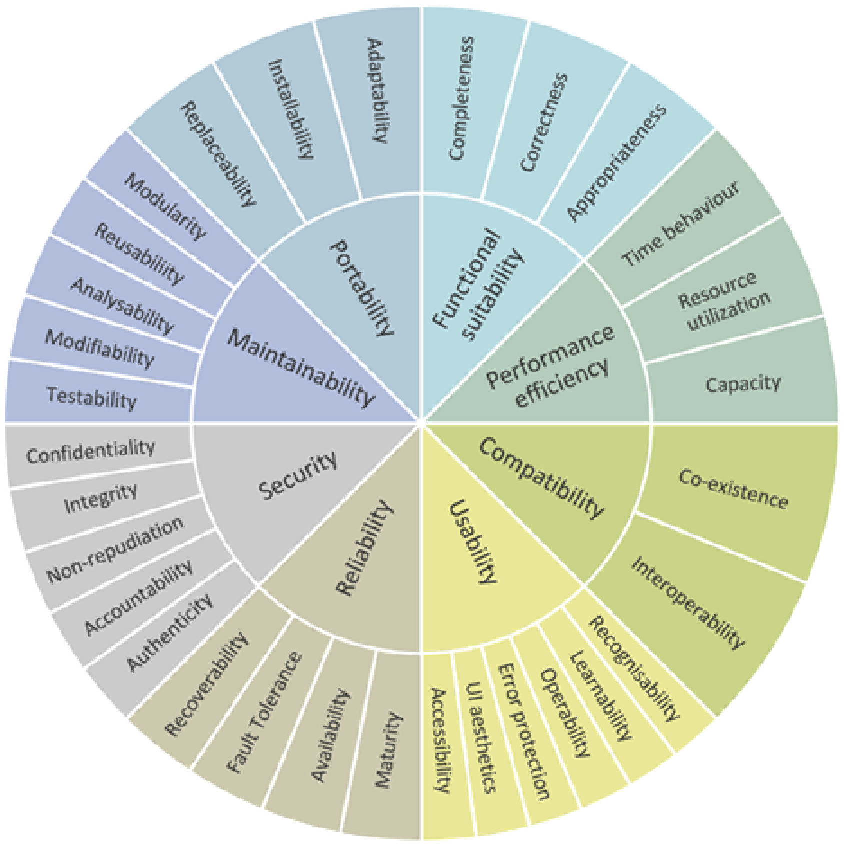


Figure 7.1: ISO/IEC 25010:2011 Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models [33]

Therefore, a recommendation would be to set requirements before the solution is developed, starting with structuring a quality framework by choosing relevant quality requirements for the intended solution. Consult the customer or the project owner and gain understanding about the important requirements for the solution. Once the developers elicit these requirements, one can implement the system accordingly. The initial phase of the system serves as a black-box which receives different inputs and provides outputs, which are then observed by the developers for validation. These inputs could be functional parameters which are evaluated later through the outputs of the black-box. With the help of the evaluation, the developers can gain more knowledge about the domain and apply improvements to the quality of the overall system.

7.2 Recommendations for Developers

Creating a robust and accurate object detection is a challenging and time consuming process, especially when the environment consists of numerous real-world objects that in many occasions share similar characteristics in terms of visual appearance. Object detection algorithms commonly use feature extraction and learning algorithms in order to detect the desired objects. Training such algorithms often requires a large set of data, another time consuming aspect is the actual training which in many cases can take days or even weeks to finish. The recommendations below are written for object detection using Cascade Classifiers but could in many cases be applied to other object detection algorithms.

7.2.1 Collect the needed data early

As emphasized, training satisfactory classifiers is a time consuming process, a recommendation would be to start the collection of the data needed for training as early as possible. As it is often ambiguous to what extent the characteristics of the positive data is needed. It is essential to continuously draw conclusions from the training and adapt upon the outcomes, to either add more data or change the approach of the data collection.

7.2.2 Continue collecting data through the whole project

Performance and accuracy are key factors for any object detector, as this is obtained through training it is crucial to collect enough data in order to achieve satisfying performance. Since it often is ambiguous how much data is needed, a recommendation would be to collect a larger set early and continue collecting until reaching the desired result as illustrated in figure 7.2.

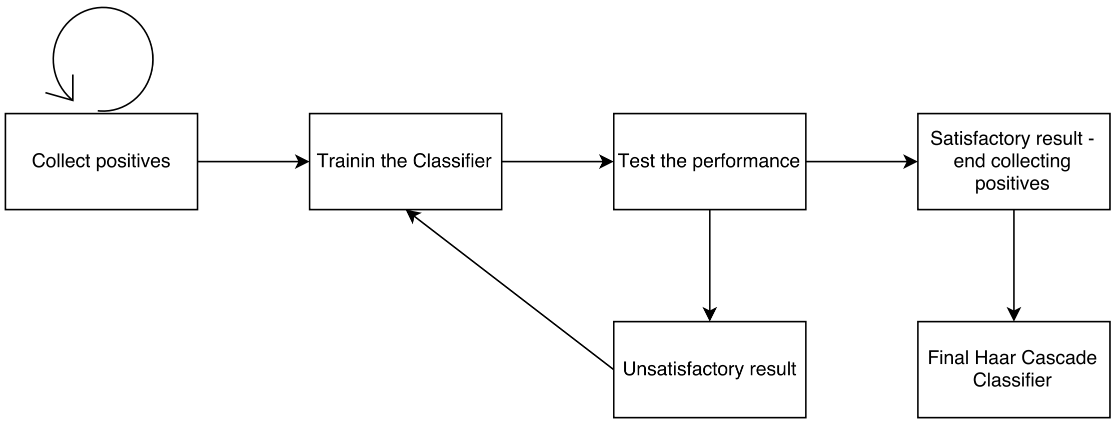


Figure 7.2: The recommended process of collecting data for training

7.2.3 Create a test environment

In case the aim of the project is to implement a solution that should be used to detect the desired object in various datasets, for instance videos, the recommendation would be to create a testing environment in order to test the solutions and evaluate the trained classifier. It is although important to point out that the testing environment should consist of data that will be similar to your customer data, this is reflected on in the section 7.3 below.

It is of importance to have knowledge about what functional and non-functional parameters are required to support the development, as the outcomes of the testing environment will later become an end product or solution. Therefore the starting phase should include videos with minimal content that does not take too long to develop. The parts that are needed for a systematic evaluation of a machine learning based approach are concerned around this content, and in specific: data set for testing, a machine-learning algorithm and a working detector. Once these parts of the test environment has been created, the next step would be to perform testing, input data that is easy to detect, and verification if it is fulfilling the functional requirements such as giving any detection. Then modify the different functional parameters in the test environment such as adjusting several parameters and verify again. Systematically draw conclusions about how the detection is getting affected by changing those parameters. This should be an repeated approach through the development process until reaching the desired performance.

7.3 Motivations for Using own Dataset of Videos

At the beginning of testing the trained cascade for detecting objects in video recordings, one could need to collect those videos as own dataset. The purpose here is to test the performance of the trained cascade. Therefore the collected videos should include different content, which reflects on the functional and non-functional parameters, that should be set at the beginning of the project. These parameters could be anything from different lighting environments to different complex environments where the desired objects might appear. The developers need to inspect the customer data and document the content of the data to include the same parameters for their own dataset. This procedure is important for the previously mentioned test environment, due to the reason of tuning and polishing the algorithm before attempting to apply it on the customer data. Another important part when collecting the dataset is to modify (if necessary) and include similar content, which might appear in the customer data. This is necessary to avoid the creation of a solution which would work for own dataset and fail for customer data. Another reason for using a own dataset might be because of the limited availability to the customer data at the beginning of the project.

For this study, we had to collect the desired dataset of videos due to the availability to the euroFOT data, which was not always accessible due to restricted access to SAFER [6]. At the first stages of the implementation and testing, a set of data had to be available at all times to visually review the performance of the algorithm. Collecting own data in terms of positives was also necessary to achieve better training of the cascade classifiers. This approach introduced flexibility to test and analyze the output of the algorithm and evaluate the functional and non-functional parameters. The recorded videos were adjusted to fit the same video characteristics as the customer videos from the euroFOT project [5] by changing them to grayscale.

7.4 Lessons Learned

This project could have been done with a Test-driven development (TDD) [37] approach to increase the quality of the implementation and the process of detecting the desired object. By first having a cascade classifier [29] one can create test cases which would fail at first. Secondly write a piece of code that should succeed by matching the coordinates of the desired objects with the output of the cascade classifier. Once the test case succeeds, refactoring should be the next step to make adjustments and improvements of the code.

The TDD approach would have been a suitable method for implementing a solution within the ML domain. The developers are required to always test every piece of code and observe the results at all time while developing in a computer vision project.

7.4.1 Minimizing complexity

The minimization of complexity is a fundamental part of software construction [38]. One of the lessons learned in this project was about “overengineering”, we realized that too much effort was put on writing code that turned out to be complicated without increasing the robustness and detection rate to the desired degree. This required us to spend more time on refactoring, which took away focus from the functional and non-functional requirements that were set. As mentioned in the Software Engineering Body of Knowledge [38] “Most people are limited in their ability to hold complex structures and information in their working memories, especially over long periods of time.“ [38], which could be reflected on the experience gained in this research. This is also important when reflecting to testing, as “The need to reduce complexity applies to essentially every aspect of software construction and is particularly critical to testing of software constructions” [38]. Therefore we find TDD [37] being a suitable software development process for this kind of projects.

7.5 Automation

For evaluating an object detection algorithm in a software engineering point of view, an automated solution should be implemented to increase efficiency of the evaluation. In terms of object detection from video recordings, the automation should be able to handle any kind of video inputs as a functional requirement for the automation. The content inside the video is the most crucial information for the detector, which should not in any case be ignored. Depending on the complexity of the video content, the developers should elicit the functional and non-functional parameters of such object detection algorithm. The functional parameters could be for the automation to detect the objects through different weather conditions appearing in the video recordings. Other functional parameters could be the different features of the object to be detected. By evaluating these parameters through the automation, the developers can determine the quality of the algorithm in terms of precision and recall. For the non-functional parameters, the automation should work properly at the desired frames/sec and not opposing any memory leaks. Memory consumption should be handled by the automation to not discard any necessary information from each frame due to increased computation.

An example could be if the objective is to detect pedestrians in a crowded environment, what would the functional and non-functional parameters to accurately detect these pedestrians be? For this case study, the object detection needed to distinguish between the vast information provided by the video recordings and shifting the focus of the algorithm to only detect the desired object. From the study one could conclude that the reliability aspect of the automation created had a major impact on the performance of the object detection. The automation should reliably detect the object at different weather condition as specified in the methodology section. Furthermore, the automation should handle different lightning conditions that could occur in different video sequences as this is affecting the detection rate. This could although be a tall order to accomplish, it is possible to find a middle ground where the reliability of the automation could give successful results.

Portability is a quality that should be considered when implementing an automation for an machine learning based approach [39]. The customer might have different hardware specification and other software environments which the automation might not work for.

Another quality aspect of this automation should be the maintainability. New features can be added during the project depending on the complexity of the target environment to detect the desired object. Therefore the code should be comprehensively written and allow those new features to be added without further complications.

8 Conclusion & Future Work

This research was conducted as a case study for the purpose of finding a suitable object detection algorithm for video content analysis, as requested by the case company SAFER. The object detection was build upon Haar-like features using Viola–Jones object detection framework [16]. The results from this study reflect on the precision and recall measured through a benchmarking with the help of an automated evaluation where the emphasised parameters affected the outcome. Such parameters covered weather conditions in the video sequences. Other parameters were functional and non-functional that influenced the software quality of the automation. The algorithm could handle different weather conditions with promising results. Compared to previous studies [25, 26, 2] which evaluated the same algorithm, we have achieved significant results in an extensive evaluation conducted on over one million video frames. The study was conducted with a software engineering approach taking the automated solution into consideration, with a reflection to relevant software quality characteristics. Furthermore, the study is leaving guidelines and recommendations for developers facing the challenges of object detection, both in terms of machine learning and development of an object detector with an aspect to functional and non-functional parameters.

Answering the research questions for this study, we can reflect on RQ1. The parameters used for the Haar training showed to have a significant impact on the precision and recall of the detection algorithm. The parameters that had biggest impact on the detection were different weather conditions. Regarding RQ2, the non-functional parameters showed to be important in terms of software quality to assure correctness of the automation. One of the most challenging and time-consuming part of this research was connected to the machine-learning. Using the right training-parameters showed to be a trivial task, since there are no clear instructions of what parameters are best suitable for good feature extraction of traffic lights, a trial and error method was used.

An extension of this study would include nighttime recordings together with an evaluation of the depending parameters that has to be taken into consideration for that specific case. Further extension of this study could include other object detection algorithms benchmarked against each other, with a result showing which one has the best performance in terms of Precision and Recall for the different weather conditions. Last but not least, the solution of the video content analysis could be extended to an application that could be used for detecting traffic lights in real time.

Bibliography

[1] Y. Zhang, J. Xue, G. Zhang, Y. Zhang, and N. Zheng, "A multi-feature fusion based traffic light recognition algorithm for intelligent vehicles," in *Control Conference (CCC), 2014 33rd Chinese*, pp. 4924–4929, July 2014.

[2] R. de Charette and F. Nashashibi, "Traffic light recognition using image processing compared to learning processes," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 333–338, Oct 2009.

[3] M. Diaz-Cabrera, P. Cerri, and P. Medici, "Robust real-time traffic light detection and distance estimation using a single camera," *Expert Syst. Appl.*, vol. 42, pp. 3911–3923, May 2015.

[4] F. Zaklouta and B. Stanciulescu, "Real-time traffic sign recognition in three stages," *Robot. Auton. Syst.*, vol. 62, pp. 16–24, Jan. 2014.

[5] "eurofot // the first large-scale european field operational test on active safety systems." `http://www.eurofot-ip.eu/`. (Accessed on 08/23/2016).

[6] "Chalmers: Safer." `http://www.chalmers.se/safer/`. (Accessed on 08/23/2016).

[7] M. Thida and H. Eng, *Contextual Analysis of Videos*. Synthesis Lectures on Image, Video, and Multimedia Processing, Morgan & Claypool Publishers, 2013.

[8] F. J. Seinstra, J. M. Geusebroek, D. Koelma, C. G. M. Snoek, M. Worring, and A. W. M. Smeulders, "High-performance distributed video content analysis with parallel-horus," *IEEE MultiMedia*, vol. 14, pp. 64–75, Oct 2007.

[9] M. Abualkibash, A. Mahmood, and S. Moslehpour, "A near real-time, parallel and distributed adaptive object detection and retraining framework based on adaboost algorithm," in *High Performance Extreme Computing Conference (HPEC), 2015 IEEE*, pp. 1–8, Sept 2015.

[10] L. He, H. Wang, and H. Zhang, "Object detection by parts using appearance, structural and shape features," in *2011 IEEE International Conference on Mechatronics and Automation*, pp. 489–494, Aug 2011.

[11] J. C. Kwak, T. R. Park, Y. S. Koo, and K. Y. Lee, "Implementation of object recognition and tracking algorithm on real-time basis," in *EUROCON, 2013 IEEE*, pp. 2000–2004, July 2013.

[12] M. Simic and R. Krerngkamjornkit, "Multi object detection and tracking from video file," in *Modern Tendencies in Engineering Sciences*, vol. 533 of *Applied Mechanics and Materials*, pp. 218–225, Trans Tech Publications, 5 2014.

[13] S. V. Kothiya and K. B. Mistree, "A review on real time object tracking in video sequences," in *Electrical, Electronics, Signals, Communication and Optimization (EESCO), 2015 International Conference on*, pp. 1–4, Jan 2015.

[14] R. Anirudh and P. Turaga, “Interactively test driving an object detector: Estimating performance on unlabeled data,” in *IEEE Winter Conference on Applications of Computer Vision*, pp. 175–182, March 2014.

[15] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Int. J. Comput. Vision*, vol. 60, pp. 91–110, Nov. 2004.

[16] P. Viola and M. J. Jones, “Robust real-time face detection,” *Int. J. Comput. Vision*, vol. 57, pp. 137–154, May 2004.

[17] “Cascade classifier — opencv 2.4.13.0 documentation.” `http://docs.opencv.org/2.4/doc/tutorials/objdetect/cascade_classifier/cascade_classifier.html`. (Accessed on 08/23/2016).

[18] “Opencv | opencv.” `http://opencv.org/`. (Accessed on 08/23/2016).

[19] K. Lin, R. Yan, H. Duan, J. Yao, and C. Zhou, “Objective classification using advanced adaboost algorithm,” in *Fuzzy Systems and Knowledge Discovery, 2008. FSKD ’08. Fifth International Conference on*, vol. 1, pp. 525–529, Oct 2008.

[20] “Cascade classifier training — opencv 2.4.13.0 documentation.” `http://docs.opencv.org/2.4/doc/user_guide/ug_traincascade.html`. (Accessed on 08/23/2016).

[21] “Opencv reference manual v2.1.” `http://picoforge.int-evry.fr/projects/svn/gpucv/opencv_doc/2.1/opencv.pdf`. (Accessed on 08/23/2016).

[22] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-up robust features (surf),” *Comput. Vis. Image Underst.*, vol. 110, pp. 346–359, June 2008.

[23] “Hough circle transform — opencv 2.4.13.0 documentation.” `http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_circle/hough_circle.html`. (Accessed on 08/23/2016).

[24] Z. Shi, Z. Zou, and C. Zhang, “Real-time traffic light detection with adaptive background suppression filter,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, pp. 690–700, March 2016.

[25] M. Salarian, A. Manavella, and R. Ansari, “A vision based system for traffic lights recognition,” in *SAI Intelligent Systems Conference (IntelliSys), 2015*, pp. 747–753, Nov 2015.

[26] “Effective traffic lights recognition method for real time driving assistance systemin the daytime.” `http://www.waset.org/publications/725`. (Accessed on 08/23/2016).

[27] S. Tiwari, S. S. Rathore, and A. Gupta, “Selecting requirement elicitation techniques for software projects,” in *Software Engineering (CONSEG), 2012 CSI Sixth International Conference on*, pp. 1–10, Sept 2012.

[28] D. Zowghi and C. Coulin, *Requirements Elicitation: A Survey of Techniques, Approaches, and Tools*, pp. 19–46. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.

[29] “Cascade classification — opencv 2.4.13.0 documentation.” `http://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html`. (Accessed on 08/23/2016).

[30] “Colormaps in opencv — opencv 2.4.13.0 documentation.” `http://docs.opencv.org/2.4/modules/contrib/doc/facerec/colormaps.html`. (Accessed on 08/23/2016).

[31] "Smoothing images — opencv 2.4.13.0 documentation." `http://docs.opencv.org/2.4/doc/tutorials/imgproc/gausian_median_blur_bilateral_filter/gausian_median_blur_bilateral_filter.html`. (Accessed on 08/23/2016).

[32] "Changing the contrast and brightness of an image — opencv 2.4.13.0 documentation." `http://docs.opencv.org/2.4/doc/tutorials/core/basic_linear_transform/basic_linear_transform.html`. (Accessed on 08/23/2016).

[33] "Iso/iec 25010:2011 - systems and software engineering – systems and software quality requirements and evaluation (square) – system and software quality models." `http://www.iso.org/iso/catalogue_detail.htm?csnumber=35733`. (Accessed on 08/23/2016).

[34] "Benchmarking machine learning techniques for software defect detection." `https://arxiv.org/pdf/1506.07563.pdf`. (Accessed on 08/23/2016).

[35] D. Zhang and J. J. P. Tsai, "Machine learning and software engineering," in *Tools with Artificial Intelligence, 2002. (ICTAI 2002). Proceedings. 14th IEEE International Conference on*, pp. 22–29, 2002.

[36] D. A. Clifton, J. Gibbons, J. Davies, and L. Tarassenko, "Machine learning and software engineering in health informatics," in *Proceedings of the First International Workshop on Realizing AI Synergies in Software Engineering*, RAISE '12, (Piscataway, NJ, USA), pp. 37–41, IEEE Press, 2012.

[37] L. Williams, E. M. Maximilien, and M. Vouk, "Test-driven development as a defect-reduction practice," in *Proceedings of the 14th International Symposium on Software Reliability Engineering*, ISSRE '03, (Washington, DC, USA), pp. 34–, IEEE Computer Society, 2003.

[38] "Swebok v3 • ieee computer society." `https://www.computer.org/web/swebok/v3`. (Accessed on 08/23/2016).

[39] "Portability in computer vision applications." `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.454.5984&rep=rep1&type=pdf`. (Accessed on 08/23/2016).