laSalle ENG
Ramon Llull University

Escola Tècnica Superior d'Enginyeria
Electrònica i Informàtica La Salle

Treball Final de Màster

Màster Universitari en Enginyeria de Telecomunicació

Conceptual approach for the implementation of a Camera data processing algorithm on a Xilinx FPGA in VHDL

Alumne
Ian Riera Smolinska

Professor Ponent
Alejandro González

ACTA DE L'EXAMEN
DEL TREBALL FI DE CARRERA

Reunit el Tribunal qualificador en el dia de la data, l'alumne

D. Ian Pau Riera Smolinska

va exposar el seu Treball de Fi de Carrera, el qual va tractar sobre el tema següent:

Conceptual approach for the implementation of a Camera data processing algorithm on a Xilinx FPGA in VHDL

Acabada l'exposició i contestades per part de l'alumne les objeccions formulades pels Srs. membres del tribunal, aquest valorà l'esmentat Treball amb la qualificació de

Barcelona,

VOCAL DEL TRIBUNAL VOCAL DEL TRIBUNAL

PRESIDENT DEL TRIBUNAL

Abstract

Automotive world is in constant technological evolution and one of the fields under development are the camera monitoring systems for mirror replacement. A simple substitution does not generate a real value as the processing capabilities of this systems can provide advanced driver-assistance. Object detection can be implemented in order to enhance the information provided by the system as well as provide passive security measures for the vehicles. However, the vehicles are already overloaded with electronic components and fitting the processing system for this kind of problem might be difficult. Therefore, an approach to connect the vehicle with a cluster of processor units on the cloud would allow powerful analytic resources without adding more electronics equipment nor increasing the power consumption to the car.

CPU's and GPU's have been tested both for hosting image processing algorithm. A potential tool to replace those platforms are FPGA, as they offer parallel computing that could accelerate the image processing and allow real-time application. However, FPGA present a low portability either from one platform to the other or between different boards.

In terms of object detection algorithms, convolutional neural networks are the state-of-the-art technology. However, as mentioned before, FPGA portability among boards is a weak point and, for a hardware field that is constantly evolving and improving presenting better and more powerful boards, having to redesign every time the algorithm is not optimal and might be the reason why there is still not a neural network framework for this kind of boards. The lack of an optimal framework design and the high resources required for implementing neural networks, especially in real-time applications, makes the implementation of those on FPGA a complicated task.

.

Resum

El món de l'automocó està en constant evolució tecnològica i un dels camps en desenvolupament són els sistemes de monitorització de càmeres per al reemplaçament de miralls retrovisors. Una mera substitució no genera un valor real, ja que les capacitats de processament d'aquests sistemes poden proporcionar assistència avançada per a la conducció. Es pot implementar la detecció d'objectes ja sigui per millorar la informació proporcionada pel sistema com per proporcionar mesures de seguretat passives per als vehicles. Tanmateix, els vehicles ja estan sobrecarregats amb components electrònics i la instal·lació del sistema de processament per a aquest tipus de problema pot ser difícil. Per tant, un enfocament per connectar el vehicle amb un clúster d'unitats de processadors al núvol permetria disposar recursos analítics potents sense afegir més equips electrònics ni augmentar el consum d'energia en el vehicle.

Les CPU i les GPU han estat utilitzades per a la execució d'algorismes de processament d'imatges. Una eina potencial per reemplaçar aquestes plataformes són les FPGA, ja que ofereixen execució en paral·lel que permetria accelerar el processament d'imatges i permetria l'aplicació en temps real. No obstant això, les FPGA presenten una baixa portabilitat des d'una plataforma a l'altra o entre diferents plaques.

Pel que fa als algorismes de detecció d'objectes, les xarxes neuronales convolucionals són la tecnologia més avançada. Tanmateix, com s'ha esmentat abans, la portabilitat de les FPGA és un punt feble i, per a un camp de maquinari que evoluciona constantment i que presenta cada cop plaques millors i més potents, haver de redissenyar cada vegada que l'algoritme no és òptim i podria ser la raó per la qual encara no existeix un framework de disseny per a xarxes neuronals per a aquest tipus de dispositius. La manca d'un marc de disseny òptim i els alts recursos necessaris per implementar xarxes neuronals, especialment en aplicacions en temps real, fa que la implementació d'aquestes en FPGA sigui una tasca complicada.

Resumen

El mundo automotriz está en constante evolución tecnológica y uno de los campos en desarrollo son los sistemas de monitoreo de cámaras para el reemplazo de los espejos retrovisores. Una simple sustitución no genera un valor real, ya que las capacidades de procesamiento de este sistema pueden proporcionar asistencia avanzada a la conducción. La detección de objetos se puede implementar o bien para mejorar la información proporcionada por el sistema, o bien para proporcionar medidas de seguridad pasiva para los vehículos. Sin embargo, los vehículos ya están sobrecargados con componentes electrónicos y el ajuste del sistema de procesamiento para este tipo de problema podría ser difícil. Por lo tanto, conectar el vehículo con un cluster de procesadores en la nube permitiría disponer de recursos analíticos potentes sin agregar más equipos electrónicos ni aumentar el consumo de energía en el vehículo.

Las CPU y las GPU han sido utilizadas como plataforma para ejecutar algoritmos de procesamiento de imágenes. Una herramienta potencial para reemplazar esas plataformas son las FPGA, ya que ofrecen computación en paralelo que podría acelerar el procesamiento de la imagen y permitir aplicaciones en tiempo real. Sin embargo, las FPGA presentan una portabilidad baja ya sea de una plataforma a otra o entre diferentes placas.

En términos de algoritmos de detección de objetos, las redes neuronales convolucionales son la tecnología más avanzada. Sin embargo, como se mencionó anteriormente, la portabilidad entre placas FPGA es un punto débil y, para un campo de hardware que está en constante evolución y mejora presentando placas mejores y más potentes, tener que rediseñar cada vez el algoritmo no es óptimo y podría ser la razón por la que todavía no existe un framework orientado a redes neuronales para este tipo de placas. La falta de un marco diseño óptimo y los grandes recursos necesarios para implementar redes neuronales, especialmente en aplicaciones en tiempo real, hace que la implementación de estos en FPGA sea una tarea complicada.

Acknowledgements

The initial concept of the project was provided by the supervisor Dr. Prof. Anestis Terzis, on the framework of the research taken out in the Institute of communication technology at the Hochschule Ulm to replace vehicle mirror systems with a camera monitoring system connected to a FPGA cluster in the cloud.

Previous to this project, a CMS with real-time display with a FPGA board was carried out. The previous board had been replaced for a new one. The current project has been supported by the doctorand Steffen Jannick Maier, whom provided knowledge and documentation through all the thesis duration.

The supervisor from the home university, LaSalle – URL, has been Alejandro González Alzate, who provided support during the competition of this project.

Table of contents

List of Figures

List of Tables:

1. Introduction

1.1. Project overview and goals

The project is carried out in the Institute of communication technology at the Hochschule Ulm, basing on a proposed project by Dr. Prof Anestis Terzis.

The purpose of this project is to design and concept the methodology for implementing a data processing algorithm for object detection and tracking in real-time for a Camera Monitor Systems (CMS). This project takes part on the framework of the research taken out in the Institute of communication technology department to study the feasibility of using FPGA cloud clusters for processing CMS in order to replace vehicles mirrors.

The idea of using FPGA is to exploit the parallel execution capabilities that this kind of boards provide. FPGA can be programmed multiple times by the user to test different designs.

The starting point of this project is working on an existing system completed previously on the same research framework, that consists of an FPGA system that displays on real-time the video images obtained by a peripherical camera. This existing project should have been upgraded with existing IP blocks[1] in order to include the object detection functionality. Due to an upgrade of the board, a compatibility problem appeared as several IP core blocks of the existing design and the camera device are not compatible with the new board. To overpass this setback, the thesis was reconducted to focusing on the conceptual design of an input-independent object detection algorithm, regardless on if the video images are provided in real-time by a camera device or if are prerecorded videos uploaded from a SD card.

The project main goals are:

1. Validate the existing base project.

2. Concept study of the algorithm for object detection

3. Design two different algorithms, for low-light and day-light situations

4. Concept study of the tools for implementing the algorithms in a FPGA

5. Determine the suitability of the system basing on the existing regulations requirements

[1] IP (Intelectual Property) blocks are predesigned and validated libraries of code that complete certain functionalities of common use in FPGA designs.

1.2. Requirements

The Camera Monitoring Systems has to fulfill some timing, resolution and field of view regulations in order to be suitable for real vehicle mirror replacement. Those regulations are:

- ISO 16505 (1): Describes the minimum technical requirements that should be fulfilled by a CMS to replace car mirrors in terms of safety aspects, ergonomic factors, performance criteria and the testing of such kind of system.

- UN Regulation No. 46 (2): The ISO itself does not provide the legal framework for mirror replacing. Is the UN Regulation, applied in the European Union, Russia and Japan, the one that establishes the minimum requirements.

More specific requirements for this thesis are:

- Video object detection and tracking of automotive vehicles and pedestrians, for day and low light scenarios.

- Using a single FPGA as electronic control unit for the CMS system.

1.3. Structure of the work

The first chapter after this brief introduction, chapter 2, is a description of the research framework in which this project develops, including a brief introduction to the existing regulations and existing projects.

The document continues in chapter 3, which presents in its first part the research taken out on the different existing object detection algorithms and the suitability of those for covering the daylight scenario. A description of the different object detection algorithms used nowadays in image processing, a brief look into existing FPGA implementations and existing tools, and a discussion about its performance are presented. From the extracted conclusions, the best algorithm is selected and adapted to our problem scenario.

The second part of the chapter 3, focuses on the low-light algorithm design and testing. A demo algorithm programmed in MATLAB is presented with an example of the performance obtained.

Chapter 4 focuses on the overall task that should be completed to successfully implement a CMS with object detection. Includes the scenarios to take into account, the way the information should be presented.

Chapter 5 proposes the following steps required to continue not just from the thesis, also for the overall project.

Finally, chapter 7 summarizes the extracted conclusions during the competition of this thesis.

2. Research framework:

2.1. Camera Monitoring Systems

Camera Monitor Systems are a set of cameras and displays used for automotive mirror replacement, as seen in Figure 2. A camera captures the indirect field of view (FOV), forwards the signal to an electronic control unit (ECU) for image processing and the information is displayed in a screen for the driver. Those systems should provide a clear view of the back or lateral parts of the car, just as mirrors do, providing the driver with information on the indirect FOV. (3)

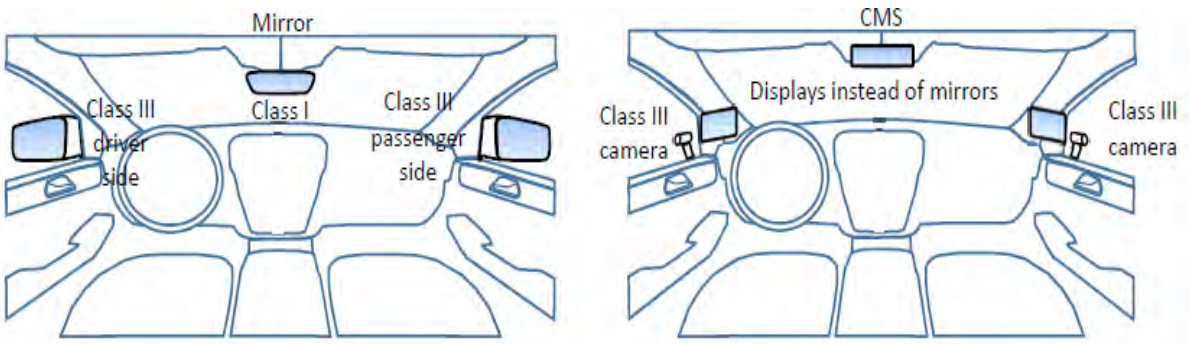


Figure 1: CMS replacement for a Class III mirror. Left image shows a conventional mirror placement in a car. In the right image the mirrors are replaced by cameras and displays, in the same position as the mirrors. Source: [3, p.5]

2.1.1. Advantages and disadvantages of the CMS

The main advantages of using those kinds of systems are:

- Improvement on aerodynamics, which leads to a reduction on fuel consumption of approximately 2%. This amount could be especially significant for mass cargo transportation, with an economical benefit for the companies due to a reduction of the petrol needed and environmentally, as heavy-duty vehicles are responsible for the 27% of the vehicle emissions in the European Union [2]. The improved aerodynamics would also reduce the noise pollution generated by the mirrors, especially in vans.

- Improvement on the indirect vision of the driver. Unlike the conventional mirrors, blind spots are covered by CMS.

[2] Statistics from the European Environment Agency (2018)
https://www.eea.europa.eu/themes/transport/heavy-duty-vehicles

- Adaptability to different light situations. Firstly, using CMS the driver would avoid glare at direct sunlight. Moreover, adding image processing between the camera and the display, heavy rain or low light scenarios could be treated in order to give a cleaner image to the driver. Finally, new ADAS (advanced driver-assistance systems) could be designed and added to the vehicle.

- Information enhancement. Through image processing algorithms, information as the distance or type of object that is approaching could be added to the display and warn the driver in danger situations.

On the other hand, CMS have the following potential challenges:

- Display location. The space inside the vehicle might be an issue in nowadays cars to properly fit the new system. Usually drivers can move the head to get a better look at the mirrors and increase the FOV; on a display this is not possible, so to increase the FOV it would be needed to install bigger displays which is a space problem.

- Human visual system. Another limitation of the displays is that they show a 2D image, and stereoscopic depth will get lost. Therefore, a distance labelling system is needed to notify the driver. Another human eye limitation is the change of focus towards the front drive view and conventional mirrors (far range view), towards watching a display (close range view). Elder people or those with vision problems, might have problems to change the focus.

- Economic investment. CMS are way more expensive than simple mirrors, due to all the electronic components inside. But the positive feeling of security and progress could be seen for the costumer as a good reason to spend a bit more, especially in the luxurious segment of vehicles.

- Relocating the components inside the mirrors. Nowadays car mirrors house several components such as antennas or cameras. Therefore, those components should be relocated somewhere else in the car, what would might be not as optimum as actual designs.

- Re-education of the drivers. One of the mains challenges would be re-educating the driver to watch the display, because even if they would expect the displays to be somewhere close to where the mirrors were, this might not be the best location.

- CMS components lifespan. CMOS image sensors used have a lifespan around 5 years, while the automotive vehicles have a lifespan of 18 years. Consequently, this kind of system would require a periodic replacement. This would affect also in terms of design, as the components should be easily accessible and replaceable.

The subject that can unbalance the pro vs con of the CMS is power consumption. All the components in a CMS are active components which require power, way different than the passive mirror. Therefore, if the power consumption is too high, the power needed for loading the batteries would neglect the saving in terms of aerodynamics. That would make a requirement to use components and algorithms for the CMS that consume the least power possible.

2.1.2. Base architecture of the CMS

The main set of a CMS is a camera, an ECU and a display.

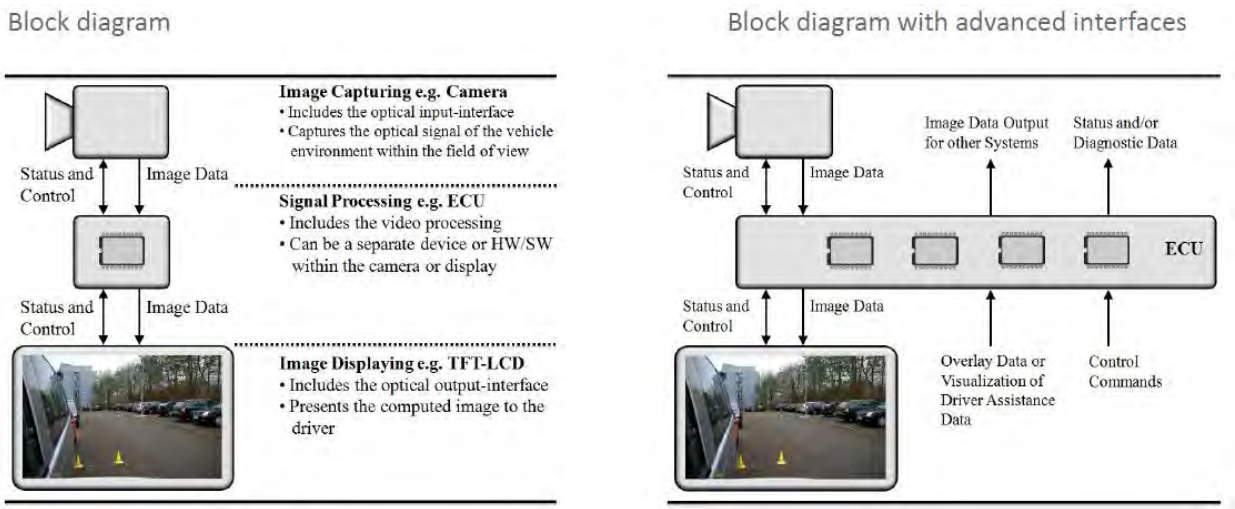


Figure 2: CMS architecture block diagram. Left image shows a simple CMS, where the image obtained is sent only to the display to be showed. In the right image, a more complex signal processing is completed and the information is not just sent to the displays but also to the ADAS components. Source: [3, p.24]

In a simple CMS (see Figure 2 left), the image is sent to the display after the proper video processing to correct the imperfections of the camera recording and proper adapting the image for displaying. Some of the light artifacts that are created in the image lens and should be processed and corrected are:

- Flares generated due to spurious reflections within the lens system, scattering or diffraction. Not avoidable. Especially critical in low-light scenarios.

- Ghosts. That kind of artifacts are well localized within the image with characteristic shapes and can easily lead to misinterpretations of the scene.

- Veiling glare. The rays of a luminous object are not properly bended and steered and a ‘fog’ or ‘haze’ spreads around the image.

- Directed flares. It often appears like a six-point stars around the light points (sun, street lights…).

- Aperture ghosts. Defocused replicas of the aperture appear on a straight line in the image plane.

- Ghost images: An upside-down and same size as the original image reflection appears and pollutes the image, inducing the appearance of those ghosts.

However, the most effective way to reduce those flares is using high quality lens, lens hoods and applying a sufficient amount of tilt to the cover glass.

In more advanced systems (Figure 2 right), the information obtained can be sent to other systems in the car for driving assistance. Warnings, object detection, overlaying images, cooperation between cameras and radar sensors, are examples of the possibilities that an advanced CMS could provide. However, the amount of information that is shown over the direct picture on the display is limited by the standards.

2.2. CMS standards and regulations

A vehicle has to comply with vehicle regulations and requirements, from technical to tax ratings. In the same way, the installation of CMS systems needs to meet some standard and regulation requirements in order to replace mirrors in series vehicle production. Standard ISO 16505:2015 (1) and UN Regulation No.46 (2) are the ones that cover CMS field.

In order to better understand this requirement, a brief definition of what a Standard and a Regulation are:

- Standard: “A standard is a document that provides requirements, specifications, guidelines or characteristics that can be used consistently to ensure that materials, products, processes and services are fit for their purpose”.[3] In this case, it is an ISO standard, an international standardization settled by the International Organization for Standardization. The ISO standards are reviewed every 5 years to reflect the state-of-the-art development. Standards are not directly legally binding, but are used in court to determine current technological state of the art. If the manufacturer does not follow them, they will have to prepare a justification. When a standard is part of a regulation, then becomes mandatory.

[3] Source: [3, p.55]

- Regulation: Are the ones to bind the legal framework. In the automotive industry, there are over 100 regulations regarding different subjects like exhaust emissions, safety, etc. Those regulations are not always international, so it need to be regularized for every market. For CMS, the UN Regulation No. 46 was updated. A need to update the regulation was already stated back in 2009, but there was no technical background to base on. For that reason, Germany triggered the ISO 16505, and the UN R.46 was parallelly advancing until complete amended entering into force in August 2016.

The ISO 16505:2015 standard as well as UN R.46 form the basis for the CMS requirements. However, both contain cross references to additional standards and regulations that must be as well followed. From electromagnetic requirements to environmental influences, CMS have to take into consideration a wide field of requirements.

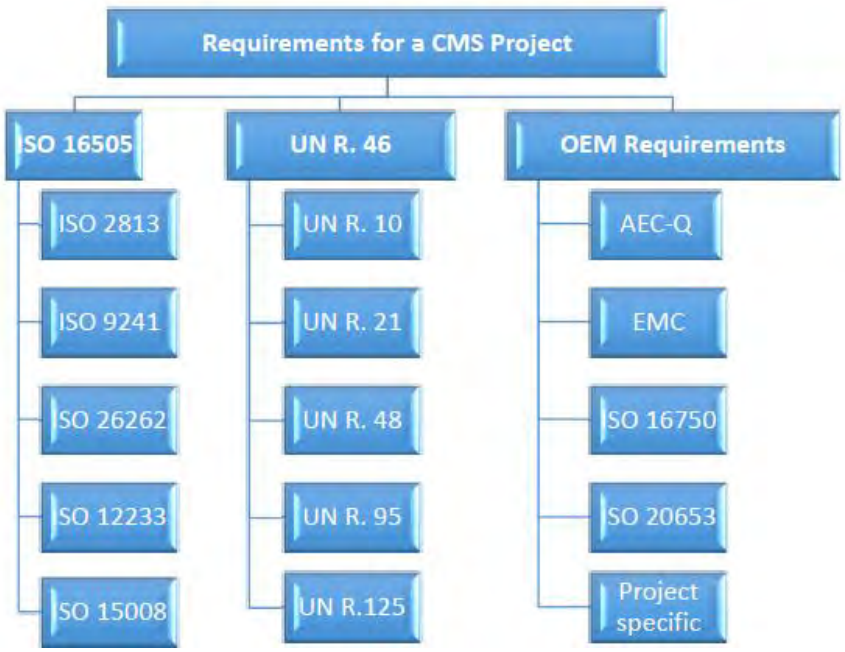


Figure 3: Standards and regulations affecting the CMS. Source: [3]

Table 1: United Nations Regulation affecting CMS. Source: [3]

United Nations Regulation	CMS relevance
Uniform provisions concerning the approval of vehicles with regard to	
No. 10: “Electromagnetic compatibility”	EMC compatibility of the complete CMS
No. 21: “Interior fittings”	CMS components behavior in case of impact with occupants or pedestrians
No. 48: “Installation of lighting and light-signaling devices”	Position of the direction indicators (e.g., currently part of the mirror housing)
No. 95: “Protection of the occupants in the event of a lateral collision”	Relevant for the characteristics of the in-vehicle CMS components, e.g., displays
No. 125: “Forward field of vision of the motor vehicle driver”	Position of the CMS components within driver’s forward field of view

The UN R. 46 establishes the requirement not just for CMS but also for conventional mirrors. One aspect that share in common both indirect vision devices, are the range of FOV that should be covered. These requirements need to be taken into account for establishing the region of interest (ROI) that will be displayed on the driver’s screen. This project focus on the standard car mirrors, specifically in the class I mirror. Figures 5 and 6 show the FOV that those mirrors or CMS should cover.

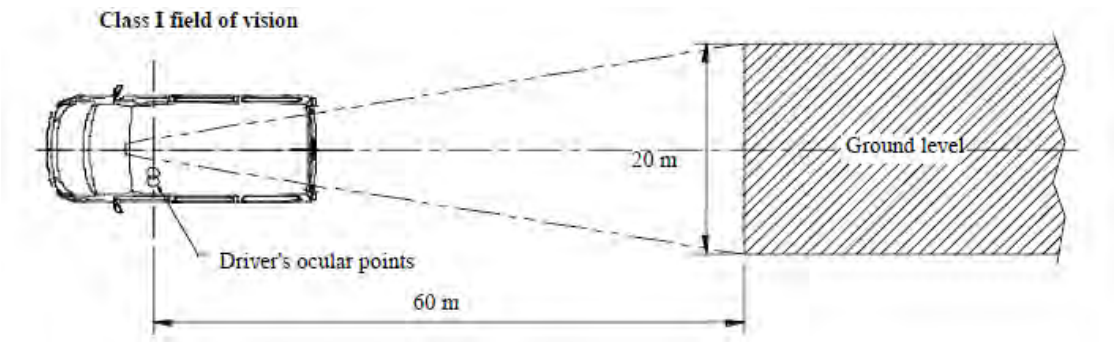


Figure 4: Class I indirect FOV. Those mirrors should cover an area of 20 meters width up to 60 meters straight away behind the driver’s ocular points. Source: [1]

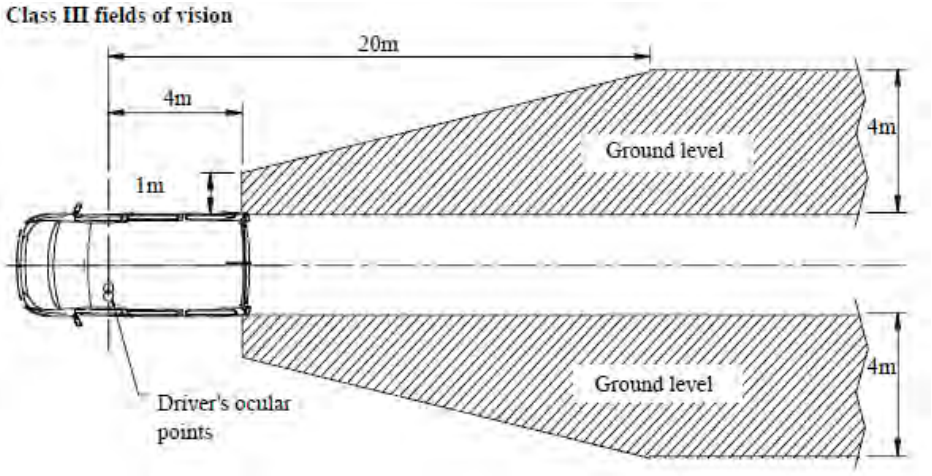


Figure 5: Class III indirect FOV. Those mirrors should cover the side indirect fov up to Source: [1]

The ISO and the regulation cover several requirements related to camera and display devices. As this project does not include designing and installing on a vehicle the physical devices, an example of these requirements is presented without specific details. Relevant requirements for this project are highlighted:

- Mechanical: In terms of passive safety in a crash scenario, the camera devices mounted should deflect in case of impact with an object, may break but without leaving sharp edges and camera lens should not break.

- Monitor Arrangement: The position of the display monitor respect the driver's ocular reference point of view is determined by a set of required angles to be met. Left side field of vision should be displayed left of the ocular reference point and vice versa.

- Monitor Isotropy: The monitor shall conform to optical requirements over a relevant range of viewing directions.

- Luminance, contrast, gray scale and color rendering. Night conditions require a specific luminance contrast.

- Artifacts. Smear, blooming and flickering are bounded.

Figure 6: Class IV indirect FOV. Source: [1]

- Sharpness and Depth of Field: The CMS shall enable the driver to observe the object space and perceive the content shown within the range of interest with enough resolution to discern the details. Geometric distortion is bounded.

- Magnification Factor: The minimum and the average magnification factors of the CMS, in both horizontal and vertical directions are established.

- MTF (Resolution): Defines the minimum distinguishable details observable in an image as is represented by the MTF10. For reasons of simplicity the requirement is defined assuming an aspect ratio of 1:1:

- Magnification Aspect Ratio: In the required field of view, the difference between the average magnification factor for horizontal and vertical direction of a CMS shall satisfy the established requirements.

- System availability. The set-up time from cold start, fast reconnection and turn-off times are also established. As the system is critical for driving, a warning system in case of malfunction should notify the driver. FOV should we always displayed on screen while ignition is on.

- **Point Light Sources:** Two light source with an intensity of 1750 cd and a distance of 1.3 m should be distinguishable as two different light sources at a distance of 250 m. This requirement is relevant for the low-light situation algorithm, which relies in point light sources detection.

- **Frame rate**: The minimum frame rate of the system shall be at least 30 Hz. At low light conditions or while maneuvering at low speed, the minimum frame rate of the system shall be at least 15 Hz. The conventional mirrors are a real-time tool, therefore CMS should behave as close as possible to real-time.

- **Image Formation Time**: Maximum 55 ms at 22ºC±5ºC.

- **System Latency:** The latency shall be lower than 200 ms 22 °C ± 5 °C.

The time requirements are the most relevant ones for this project, especially the frame rate. The execution times of the introduced image processing algorithms must be low enough to keep the frame rate inbounds of the specified limits, to keep it a real-time application.

- **Overlays**: Only temporary, transparent and related to rearward driving-related information overlays are allowed. The maximum area a single overlay can cover is 2,5 % of the FOV, 15 % the complete obstructions in Class I and 10 % in the others. These obstructions include bodywork, heating elements, etc. This requirement is critical in the object detection scenario, as the amount of information highlighted and overlaid on the image should be restricted.

The ISO 16505:2015 also proposes testing methods for the CMS, in order to test the resolution, rendering and sharpness requirements. Test charts, like the chessboard used for contrast, are placed in front of the camera device.

2.3. FPGA research framework in HS Ulm

The research carried out at HS Ulm focus on using FPGA as the ECU for the CMS. Aside from the first logical step which is using one single FPGA, the long-term goal is to concept a CMS on which most of the image processing would run on the cloud, being the central processors a cluster of FPGAs (Figure 7). The vehicle would send the information to the cloud, where the algorithms would run and return the processed information for the ADAS and virtual reality displaying. With a powerful enough computational power, AI could be run to predict vehicle behaviors and trajectories from the images obtained.

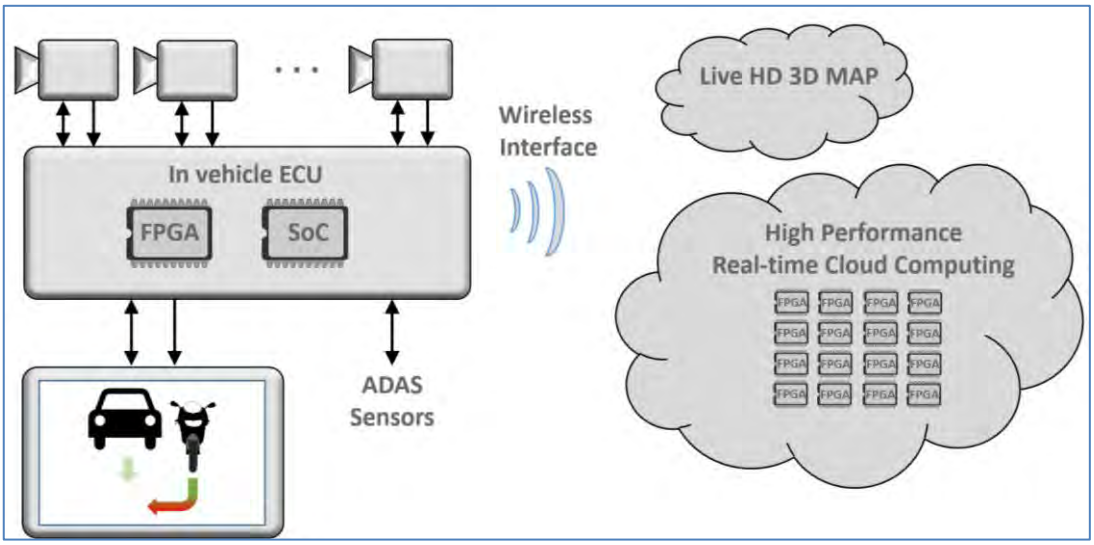


Figure 7: Hybrid image processing system proposed by Dr. Prof. A.Terzis. (4) The car would mount the camera devices and displays, but the processing will be carried out at the cloud. The information is sent via Wireless interface from the car to the FPGA cluster.

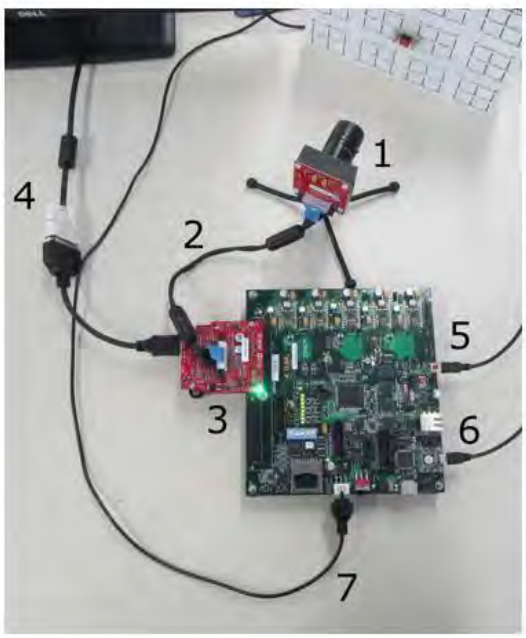
Some of the benefits that this hybrid system would provide are:

- Minimization of the power consumption in the car.
- Increased computational power.
- Expandable and upgradable functionality.
- Redundant system. New bottleneck would be the communications, but the irruption of the 5G communications and the IoT may provide the right infraestructure.

2.4. Starting point for the project

In the framework of the research taken out at HS Ulm, a first prototype CMS using a FPGA board had been implemented. This system is a generation 1 CMS, which means that its functionality was just to display a real-time image obtained through the camera. The following step, the goal of this thesis, was adding object detection algorithms to enhance the information showed on the display.

The system shown in Figure 8 used a ON Semiconductor VITA-200 Image Sensor with a 1920x1080 resolution and up to 92 fps. The camera connects with an Avnet FMC-Module that has HDMI input and output, LCED-interface for camera connection and FMC-Interface for ZYNQ-Board (FPGA) connection. As a display, a standard 24-inch PC Monitor with HDMI-Interface connection is used. The main component of the CMS acting as an ECU, is a ZYNQ-7000 SoC Evaluation Board (ZC702).



1. Camera
2. Interface: LCEDI
3. FMC IMAGEON HDMI Module
4. HDMI/DVI Output
5. JTAG USB Port
6. UART USB Port
7. DC current connector

Figure 8: Hardware setup of the mirror replacement system [HSU]

The code for this CMS system includes preprocessing of the image such as RGB to YCbCr transformation, Chroma resampling from 4:4:4 to 4:4:2 to reduce saturation and cutting out the ROI, reducing the size of the image to 1024x768 pixels. With this compression an image 4 times smaller is obtained, what fastens the data processing.

An overlay is also included to frame objects of interest as shown in Figure 9.

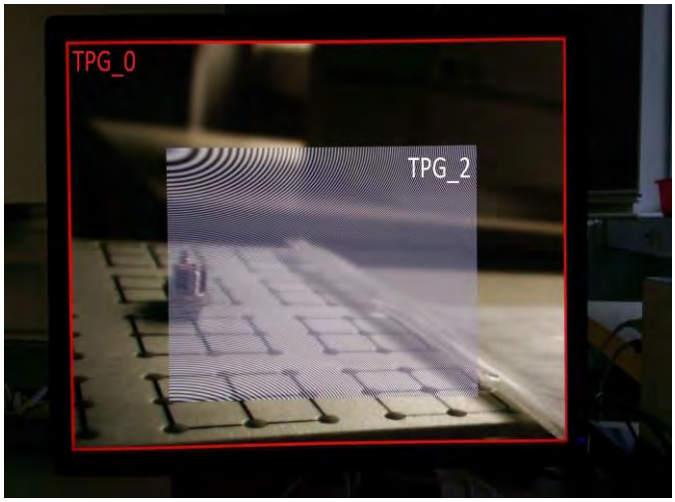


Figure 9: Image obtained with the setup and an overlay example. [HSU]

The system provided a 30-fps rate, which fulfills the regulations requirements and can be considered real-time. In terms of energy, the power consumption is around 2 W, and in terms of board usage, only a 7% of the Flip-Flops, a 12% of the LUTS, 1% of Memories and 5% of the DSPs are used. Therefore, most of the board resources stay available for additional image processing algorithms.

2.5. Upgraded hardware

The thesis starting point was using the existing design and, by IP-Cores, add the object detection algorithm. However, the FPGA board used as ECU was upgraded to a Xilinx Zynq UltraScale+ MPSoC ZCU106 Evaluation Kit, which implied transferring the old set up and code to the new board.

First, a slight comparison of the boards to understand the reason of the upgrade. The complete board information and schematics can be found in Xilinx webpages for the boards ZC702 (5) and ZCU106 (6).

Table 2: Features comparison of ZC702 and ZCU106 boards.

	ZC702	ZCU106
System logic cells (k)	85	504
Memory (Mb)	4,9	38
DSP Slices	220	1,728
Maximum I/O Pins	200	464

As can be observed in the Table 2, the resources that the new board can offer is about 8 times those the ZC702 have. In terms of I/O pins, it doubles the number. Moreover, the ZCU106 board comes with a PCIe interface.

In terms of Design Tools, both boards use the Vivado Design Suite tool, but the ZCU106 comes also with Xilinx SDK and PetaLinux tools. This last one allows deploying a Linux operating system on the Xilinx platform.

Figure 10: Set up with the new board ZCU106. The connections are the same as with the previous board.

Despite being a more powerful board and with much more resources, the replacement of the board caused a compatibility problem as several IP core blocks of the existing design and the camera device are not compatible with the new board. To overpass this setback, the thesis was reconducted to focusing on the conceptual design of an input-independent object detection algorithm, regardless on if the video images are provided in real-time by a camera device or if are prerecorded videos uploaded from a SD card.

3. Object detection algorithm:

3.1. Algorithm functionality

The algorithm should detect and track vehicles such as cars, trucks, motorbikes and buses, as well as pedestrians and bicycles. In first instance, those objects should be highlighted in order to notify the driver. An increased functionality could be a visual color notification to inform about the distance in which those objects and therefore the potential thread that represent.

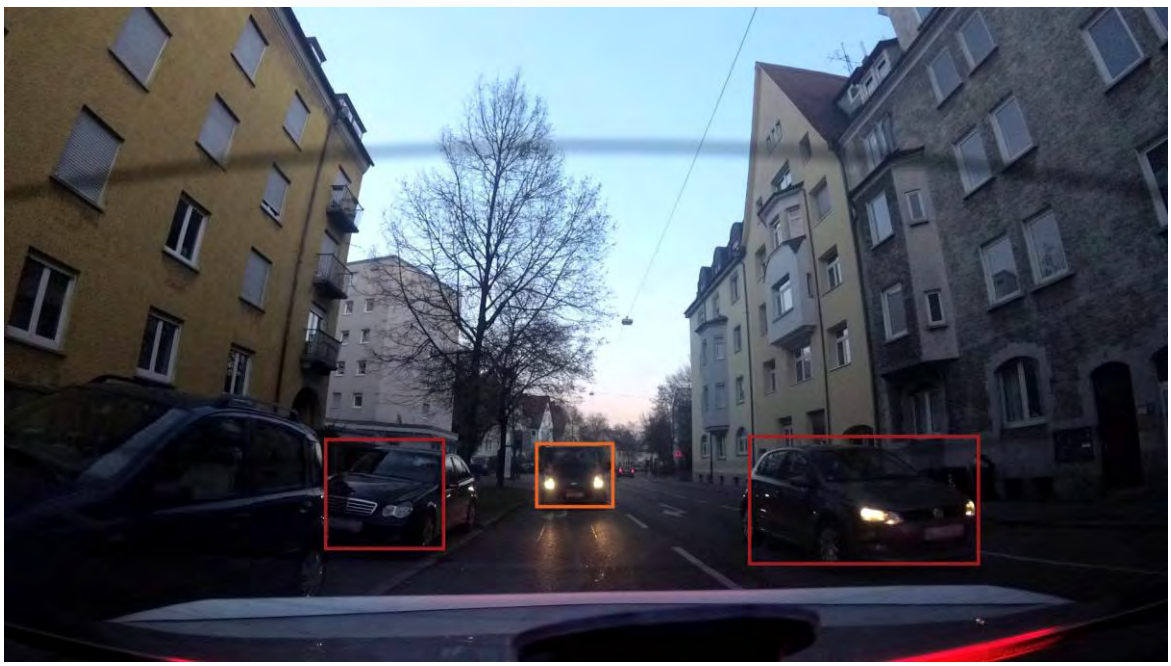


Figure 11: Example of the expected image result

Two different algorithms, one for daylight scenario and one for a low-light scenario are approached, as it is not possible to use only one for both scenarios with a single camera. With a second night-vision camera might be possible to use a single algorithm, but it is not the approach of this research as one of the considered options is to use the parking assistance camera that many cars already incorporate as input device.

Establishing a control algorithm that decides when to use the daylight approach or the low-light one is not part of this thesis. A luminous threshold would have to be determinate as well as the minimum times to consider that has been a change of light environment more than a punctual fluctuation of light.

3.2. Day algorithm

3.2.1. State of the art

From the different algorithms that exist for object detection, in this thesis has been taken into account the Haar, HoG and CNN tools as options to implement the design.

3.2.1.1. Haar

In 2001 the first object detection framework to provide real-time competitive detection rates was created by Paul Viola and Michael Jones, known as Viola-Jones object detection framework. (7) The algorithm presented a high detection rate in real-time for the face detection problem.

The algorithm is based on Haar-like features and integral images. Haar-like features equals to the pixel sum in the white rectangles minus the pixel sum of the black rectangle patrons. The feature value is determined by comparing the feature sum to the feature threshold. The feature set and threshold of an object are generated by training a large number of images with the AdaBoost algorithm, which determine the typical features of a specific object. For each object, a big number of features are calculated. A frontal face contains up to 2135 Haar-features, like the ones showed in Figure 12 for the eyes. The algorithm divides the image in smaller sub-windows and calculates the Haar-features on each one of them.

Figure 12: Haar-features for eyes. Source:[9]

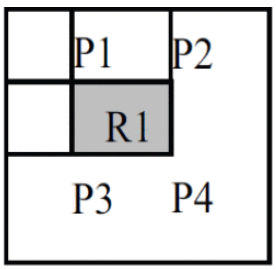


Figure 13: Integral image calculation. Source:[9]

To efficiently calculate the pixel sum of an arbitrary rectangle, the algorithm uses an integral image as an auxiliary data structure. In an integral image, each point stores the pixel sum of a rectangle, starting from the top left corner to this point. With the integral image, calculating the sum of an arbitrary rectangle can be done in constant time, e.g., Sum(R1)=P4−P2−P3+P1, as shown in Figure 13.

In order to reduce the computation due to the huge number of features, a decision cascade is implemented, as seen in Figure 14.

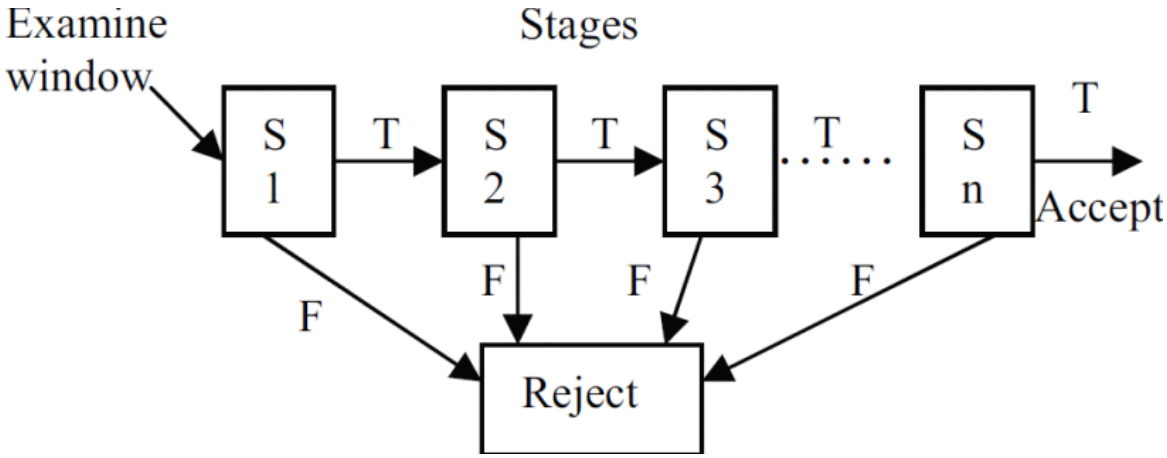


Figure 14: Decision cascade. T = True F = False. Source:[9]

The Haar features are divided into several stages. For example, the frontal face has 22 stages and each stage has from 3 to 200 Haar features. The algorithm calculates the feature value for each feature within one stage and then sums the values to get the stage sum. If the stage sum passes the stage threshold, the algorithm continues to the next stage. Otherwise, the algorithm terminates and rejects the current examine window. If an examine window passes all stages, the algorithm accepts the current window meaning that the object is found.

Finally, the AdaBoost classifier extracts the most relevant features and reduces the computation time.

In terms of FPGA implementations, exist several papers focused on face detection, which might be useful for the pedestrian detection but not for the vehicle detection part. However, the results that provide give an orientation of the potential on using the Haar features for our problem.

Irgens, Bader, Lé, Saxena and Ababei (8) present an implementation that achieves a frame rate of 4,4 fps for detecting up to 10 faces at once, with input images of 320x240. The implementation was completed in a low budget board, which encourages to believe that a more powerful board could provide better results and allow a bigger number of detections.

Chen Huang and Frank Vahid (9) present an implementation also for eye, face and multiple face (complex face = 12 faces) detection and test different number of classifiers. Each classifier has different amount of Haar features.

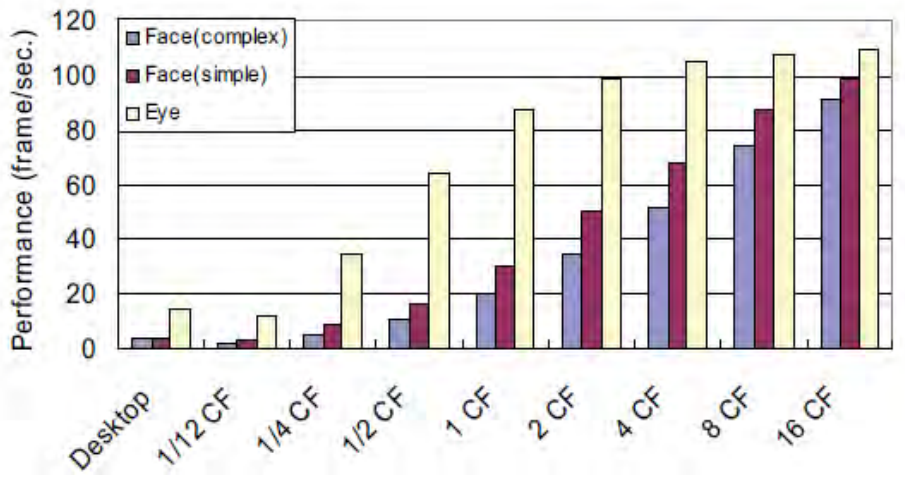


Figure 15: Huang and Vahid performance results. Comparison of a single face, single eye and 12 faces detection Source [9]

As seen in Figure 15 the desktop version gets a performance close to the one that can be obtained with a 1/12 classifier in a FPGA. Being able to run up to 16 classifiers in parallel on FPGA, the increasement on performance by using a FPGA is huge. Among their conclusions, they state that an IP (soft core for object detection utilizing a static or possibly parameterized VHDL or Verilog description would not cover the tremendous difference among generated designs like the one described in their paper. As such, custom generators, including custom design space exploration, may become increasingly necessary for complex applications to be useful across a range of FPGA devices.

Cho, Mirzaei, Oberg and Kastner (10) test two different implementations of the Viola-Jones algorithm, with input images of 320x240 and 640x480 pixels.

Table 3: Results for a 320x240 input images. Source: [10]

# of Faces	Software Classifier	Hardware	
		Single Classifier	Triple Classifier
1	1,256 ms (0.79 fps)	57.131 ms (17.50 fps)	34.712 ms (28.80 fps)
6	1,402 ms (0.71 fps)	64.981 ms (15.39 fps)	37.378 ms (26.75 fps)
11	1,538 ms (0.65 fps)	79.628 ms (12.55 fps)	41.711 ms (23.97 fps)

Table 4: Results for a 640x480 pixel images. Source: [10]

# of Faces	Software Classifier	Hardware	
		Single Classifier	Triple Classifier
1	2,165 ms (0.46 fps)	189.199 ms (5.28 fps)	133.143 ms (7.51 fps)
6	2,919 ms (0.34 fps)	254.254 ms (3.93 fps)	146.745 ms (6.81 fps)
11	3,129 ms (0.31 fps)	260.169 ms (3.84 fps)	152.664 ms (6.55 fps)

From the previous papers it can be extracted that Viola-Jones implementations in FPGA exist but are mainly oriented to face detection. Therefore, to apply Viola-Jones to our problem, the Haar features for all the objects should be computed and that would increment considerably the computation required and therefore will make the frame rate even lower than the ones obtained for face detection. Moreover, the resolutions used in the presented papers were way smaller than the ones that a CMS would provide. Therefore, downscaling the image would be required as well. However, in all cases exists an improvement in terms of performance and low power consumption.

3.2.1.2. HoG

The HoG or histogram of oriented gradients, bases as its names says in gradients. Gradients are vectors that point the direction of pixel intensity variation. Direction and magnitude are calculated, using the equations (1) and (2). Usually the object edges present a larger magnitude.

$$g = \sqrt{g_x^2 + g_y^2} \qquad (1)$$

$$\theta = \arctan \frac{g_y}{g_x} \qquad (2)$$

The values of the pixels are grouped in 8x8 cells. The values of the 64 values of the pixel cell are distributed in a 9-bin histogram, ranging between 0 and 180 degrees, as seen in Figure 16.

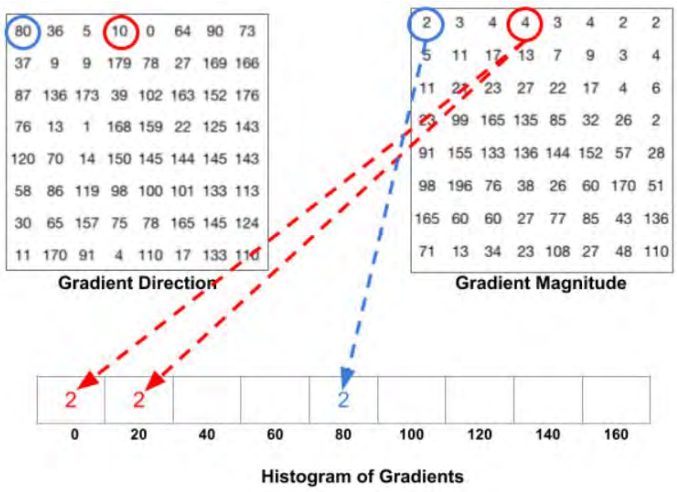


Figure 16: Histogram distribution of the gradient values. If a value stands in the middle of 2 bins, for example 10, it distributes equivalently its value between the 2 neighbour bins, in this case 0 and 20. Source: (37)

The histogram values are then classified with a support vector machine (SVM).

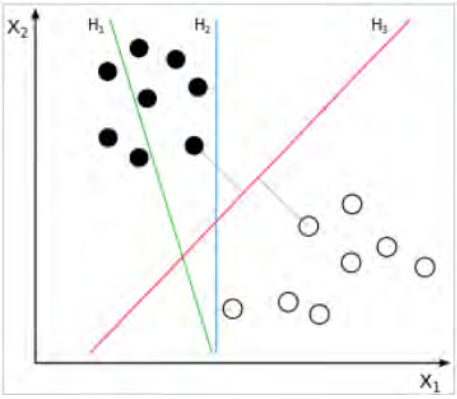


Figure 17: H3 classification border that determine the SVM
Source: (30)

Being the original algorithm with the maximum-margin hyperplane as decision border (Figure 17), non-linear classifiers can be applied, minimizing the error.

A good application for the HoG is pedestrian algorithm, what would cover the respective part of our project, remaining the vehicle part to be solved.

Lee, Son, Choi and Min (11) present and implementation that can detect pedestrians and vehicles at 33 fps on 640x480 pixel resolution images. However, no detection rate nor specification of the number of objects that the system can detect is mentioned.

Hahnle, Saxen, Hisung, Brunsmann and Doll (12) present a multiscale pedestrian detector to detect up to 18 different pedestrian sizes with a frame rate of 64 fps on images with 1920x1080 resolution.

Globally, high resolution and frame rate implementations can be achieved on a FPGA, but mainly focused on the pedestrian detection problem. The detection of different classes of objects has not been tested and therefore would be needed to adapt and test the algorithm to check its feasibility, and a decrease in efficiency.

3.2.1.3. CNN

The cutting edge of object detection are the convolutional neural networks. CNN try to mimic the human neuronal connections done to recognize and classify images. Image classification is the task of taking an input image and outputting a class (a cat, dog, etc.) or a probability of classes that best describes the image.

When we see an image, we are able to immediately characterize the scene and give each object a label. The input for computers, is an array of pixel values, usually in terms of width x height x dimension (3 if it's an RGB image, 1 if it is grayscale). Each of these numbers is given a value from 0 to 255 which describes the pixel intensity at that point. The idea is that from this array of numbers it will output the probability of the image being a certain class (.80 for cat, .15 for dog, .05 for bird, etc.), and label it as the more probable.

The human neuronal cells responsible for image recognition get triggered by specific kind of features, for example with some cells get triggered by vertical edges and others by horizontal edges. The same way, the computer is able perform image classification by looking for low level features such as edges and curves, through different layers. Series of convolutional, nonlinear, pooling (down sampling), and fully connected layers, provide the output.

The first layer in a CNN is always a Convolutional Layer. A filter, also known as kernel, formed by weights and with a considerably smaller size than the input image but respecting the dimension, is sliding and convoluting with the input image.

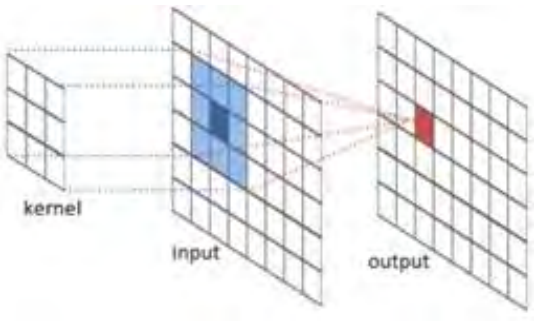


Figure 18: Convolutional layer of a CNN. The kernel is convoluted over the input image. Source: (37)

As the filter is sliding, or convolving, around the input image, it is multiplying the values in the filter with the original pixel values of the image. These multiplications are all summed up into a single number output. The resulting matrix is called activation map or feature map.

Several convolutional layers can be used in the CNN, together with ReLU (Rectified Linear Units), pool and dropout layers. After each conv layer, it is convention to apply a nonlinear or activation layer. The purpose of this layer is to introduce nonlinearity to a system that has just been computing linear operations during the convolutional layers. The ReLU layer applies the function f(x) = max(0, x) to all of the values in the input, changing all the negative values to 0.

Pooling layers take a filter (normally of size 2x2) and a stride of the same length, and then applies it to the input values and outputs the maximum number in every subregion that the filter convolves around. This case is known as the max-pooling, but also average pooling (Figure 19) and L2-norm pooling can be used. The intuitive reasoning behind this layer is that once we know that a specific feature is in the original input image, its exact location is not as important as its relative location to the other features.

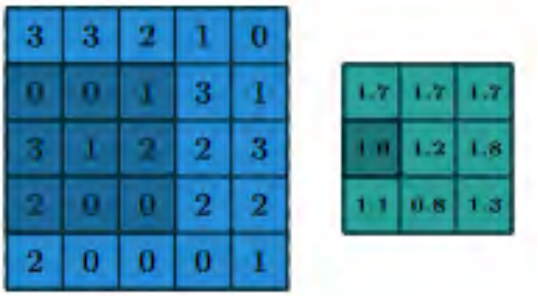


Figure 19: Example of average pooling layer. Down sampling by averaging the values of a 3x3 cell. Source: (37)

Dropout layers have a very specific function in neural networks. One problem can appear when after training, the weights of the network are so tuned to the training examples they are given that the network doesn't perform well when given new examples. This layer drops out a random set of activations in that layer by setting them to zero. This layer is only used during training, and not during test time.

The final layer of the CNN is the fully connected layer. This layer takes the input from the previous layer and outputs an N dimensional vector where N is the number of classes that the program has to choose from, and each number represents the probability of a certain class. To calculate the probability, it looks at the output of the previous layer and determines which features most correlate to a particular class. For example, if the program is predicting that some image is a bird, it will have high values in the activation maps that represent high level features like wings or a beak.

The most important part of the CNN is the training. Before the CNN starts, the weights are randomized. The filters don't know how to look for edges and curves. The way the computer is able to adjust its filter weights is through a training process called backpropagation. During the training, labelled images are introduced into the system. As the CNN knows that the result should be probability of 1, it takes the calculated value with the existing weights and calculates the mean square error between the expected value and the obtained one. By optimization, the CNN calculates the optimal weights with a set of training images.

Once the CNN is trained with the training dataset, the test image dataset is used to check the right behavior of the network.

CNN are mainly focused on single object image classification, but this project focuses on multiple object detection. This problem, which object detection researchers met before, is solved by using region-based CNN, R-CNN. The original image is divided in possible regions of interest, and the CNN runs over those regions, as seen in Figure 20.

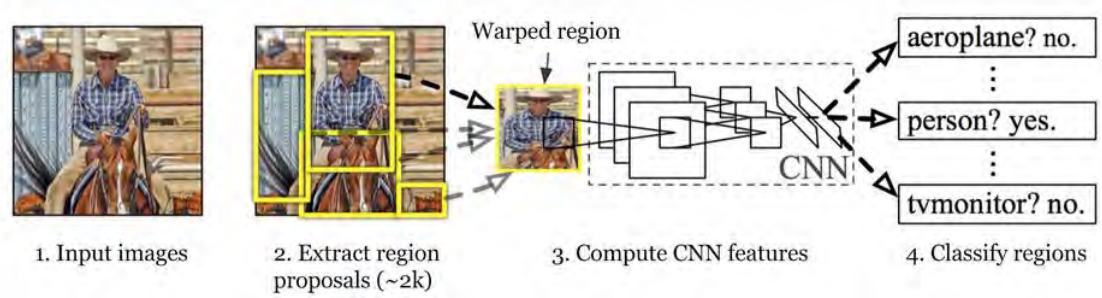


Figure 20: R-CNN stages. Division by the R-CNN into Regions of Interest. Source: (13)

R-CNN however calculates usually up to 2000 regions of interest (13), which requires a lot of time and therefore is not suitable for real-time execution.

In order to overcome those limitations, Fast R-CNN and Faster R-CNN where created. Instead of getting the ROIs of the original image, this one is the input to the convolutional layer and the ROI are decided from the resulting feature map. As the Fast R-CNN uses selective search to get the ROI, the Faster R-CNN uses a parallel network to obtain them, and therefore the system is getting trained constantly.

The increasement in terms of speed in the Fast and Faster R-CNN is considerable, as can be seen in Figure 22, which makes R-CNN suitable for real-time situations.

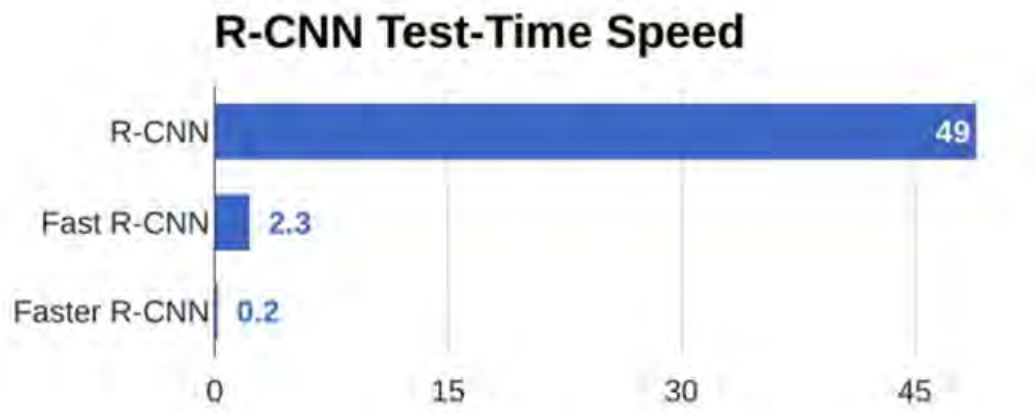


Figure 21: Comparison of the different R-CNN test-time speed. Fast R-CNN reduces by 21 the test-time, and the Faster R-CNN up to 240 times respect the original network design. Source: (13)

Most of the detection systems apply the model to an image at multiple locations and scales in order to detect different object sizes. That usually implies dividing the original image into smaller ones and running the neural network over them. High scoring regions of the image are considered detections. An alternative approach is taken by the You Only Look Once algorithm, YOLO (14). A single neural network is applied to the full image. This network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities to obtain the final detections, as seen in

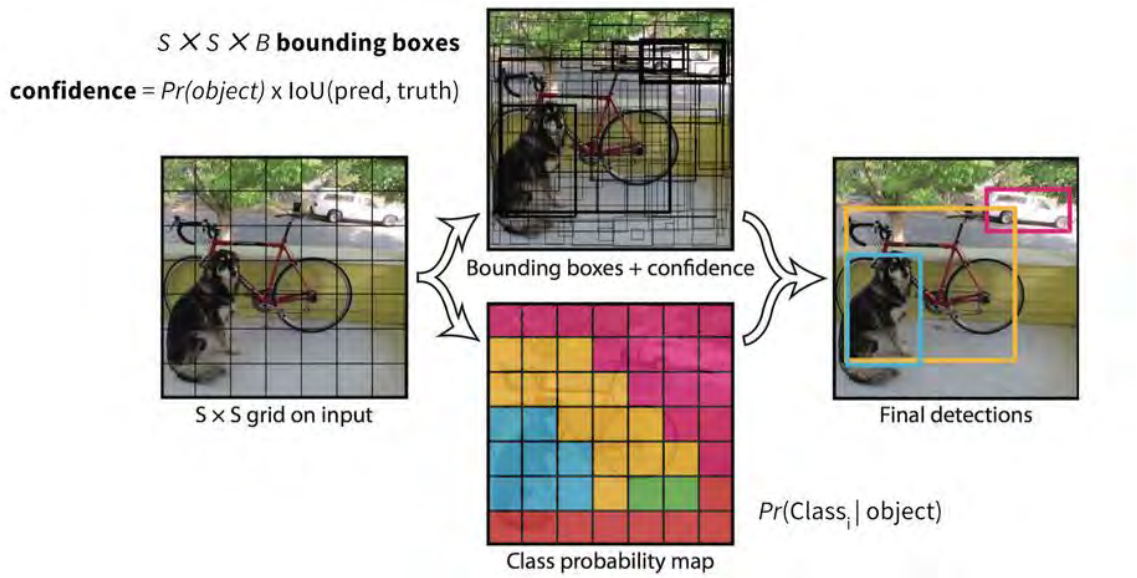


Figure 22: YOLO methodology. Bounding boxes and probabilities are calculated separately and then merged to propose the final detection bounding box. Source: (14)

Table 5: YOLO vs R-CNN performance. YOLO offers a much higher frame rate than R-CNN, and is suitable for real-time situations. The mAP is the common metric for object detection algorithms and represents the average precision of the system. Source: [14]

Detection Frameworks	Train	mAP	FPS
Fast R-CNN	2007+2012	70.0	0.5
Faster R-CNN VGG-16	2007+2012	73.2	7
Faster R-CNN ResNet	2007+2012	76.4	5
YOLO	2007+2012	63.4	45
YOLOv2 288 × 288	2007+2012	69.0	91
YOLOv2 352 × 352	2007+2012	73.7	81
YOLOv2 416 × 416	2007+2012	76.8	67
YOLOv2 480 × 480	2007+2012	77.8	59
YOLOv2 544 × 544	2007+2012	**78.6**	40

YOLO presents a higher frame rate than R-CNN, but it presents problems on detecting small objects within the image, by constraints of the algorithm architecture.

Every CNN can be defined with different number and distribution of layers, different weights and object classes to be detected. Many CNN exist, but most of them cover many more object classes than the ones we need for our project, which translates into a missusage of resources. Therefore, the best way to get the ideal CNN is to create and adapt a new one for our specific problem, with our own dataset. Designing and training your own CNN from scratch can take weeks. That is the reason why in the context of machine learning, it is common to use the concept of “transfer learning”, which is using the information obtained by solving a similar problem and apply it to the new problem, in this case the process of taking a pre-trained model and “fine-tuning” the model with your own dataset. Keeping the already calculated weights, the last layer is changed for the custom classifier and the network is trained normally.

In terms of resources, CNN require a lot of them and running one on an average computer in real-time is not possible even with GPU and CUDA parallel computing. Therefore, software desings require expensive equipment and a big power consumption. FPGA’s, with the parallel computing, could be a good hardware solution and a potential tool for real-time application.

Jason Chong's presentation (15) presentation on machine learning compared the iteration time of different number of CPUs working together in front of one single FPGA. As can be seen in Figure

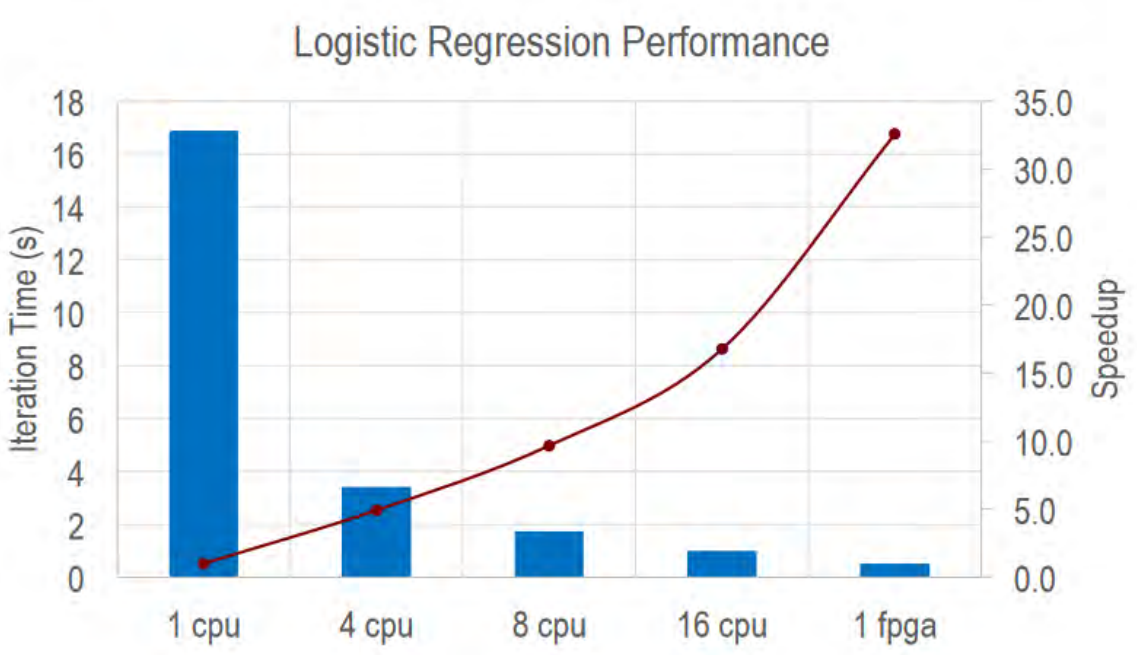


Figure 23: Iteration Time (s) for different number of CPU vs a single FPGA. The speedup by using an FPGA is considerably remarkable.

In the same presentation, a proposal system for object detection in real-time is presented, but using a cluster of FPGA instead a single one.

However, CNN in FPGA is still a combination under development. The interest on combining them is real, to the point that Xilinx has spend human and economical resources on testing them. Xilinx recently acquired DEEPHi (16), a technological company specialized in FPGA-based deep learning real-time video recognition systems, which cover among others pedestrian and vehicle tracking. A Xilinx development team presented as well a YOLO implementation on FPGA (17) at the Annual Conference on Neural Information Processing Systems (NIPS) 2017. They achieved 16 fps with a precision of 50%, using a FPGA adapted version of the Tiny YOLO, a light version of YOLO. They provided an open source code of the implementation, but runs on a special framework for Python that not all FPGA's support. (18)

Ford and Xilinx (19) did work as well together for 10 months on a CMS fpga system, using in this case multiple cameras, SDSoC C++ algorithms and obtaining a frame rate of 12 frames per second.

Xilinx also presents a Machine Learning demo using YOLO algorithm on SDSoC 2018.3 for the ZCU102, ZCU104, Ultra96 and DPB1303 FPGA boards.

A paper by Zhao, Niu, Wu, Luk and Liu (20) presented an FPGA implementation of the YOLO algorithm, comparing it with other 3 platforms and obtaining the results observable in the Table 6.

Table 6: CNN performance on 4 different platforms. It compares the results obtained with a FPGA vs those obtained with a CPU/GPU.

	x86 CPU	ARM CPU	FPGA	GPU
Platform	Intel Core i7	ARMv7-A	Zynq (zc706)	GeForce Titan X
Num. of Cores	8 (4 used)	2 (2 used)	-	-
Compiler	GNU GCC	GNU GCC	Vivado (2016.2)	CUDA (v7.5)
Compile Flags	`-Ofast`	`-Ofast`	-	`-Ofast`
Clock	3.07 GHz	Up to 1GHz	200 MHz	1531 MHz
Technology	45 nm	28 nm	24 nm	16 nm
YOLO (Tiny)	**1.12**s	**36.92**s	-	0.0037s (178W)
YOLO (GoogLeNet)	13.54s	430.6s	**0.744**s (1.167W)	0.010s (230W)
Faster RCNN (ZF)	2.547s	71.53s	-	0.043s (69W)
Faster RCNN (VGG16)	6.224s	Failed	0.875s (1.167W)	0.062s (81W)

Their implementation offered a frame rate of barely 1 frame per second, which wouldn't really suitable for our problem. Moreover, even if FPGA performs better than a CPU, GPU with CUDA it is way faster.

Another group that is researching on neural network deployment over FPGA, is the Christos Bouganis team at the London College. As seen in Figure 24, a comparison of the different neural networks implementations existing nowadays classifying them with 5 characteristics.

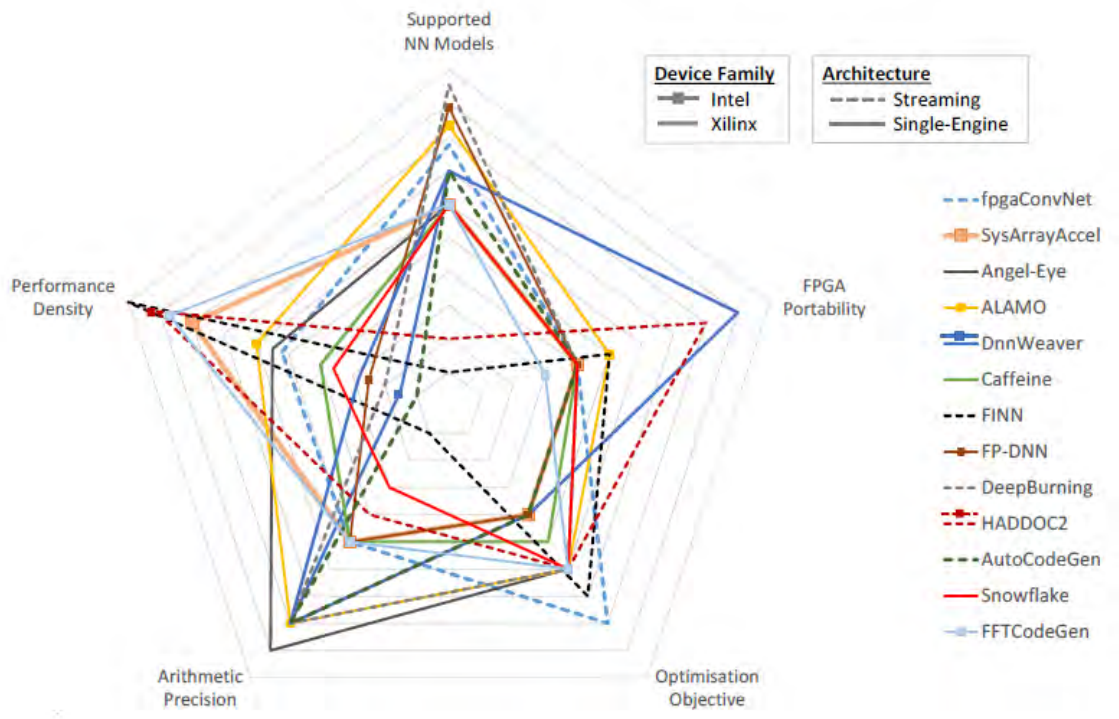


Figure 24: Bouganis team comparison on existing FPGA neural networks implementations. Of the five characteristics that are presented, the most weakened and at the same time more critical from a personal point of view is the FPGA portability. Most of the existing implementations are locked to the used FPGA model and not portable to the chosen board. (30)

In the previous comparison, the implementation that presented a theoretical major portability is the DnnWeaver (21). In this paper a a framework that automatically generates a synthesizable accelerator for a given FPGA-DNN pair from a Caffe (22) deep learning framework. In order to test their system, a benchmark testing on the three different FPGA boards shown in Table 7 was completed.

Table 7: FPGA devices tested. Source: (21)

	Xilinx Zynq ZC702	Altera Stratix V SGSD5	Altera Arria 10 GX115
FPGA Capacity	53K LUTs	172K LUTs	427K LUTS
	106K Flip-Flops	690K Flip-Flops	1708K Flip-Flops
Peak Frequency	250 MHz	800 MHz	800 MHz
BRAM	630 KB	5035 KB	6782 KB
MACC Count	220	1590	1518
Evaluation Kit Price	$895	$6,995	$4495
Technology	TSMC 28nm	TSMC 28nm	TSMC 20nm

The DNN used and its functionality are the ones seen in Table 8, and the CPU and GPU used for comparison the ones seen in Table 9.

Table 8: Benchmark DNN dataset, functionality and weights size. (21)

Network	Data set	Domain	Model Size (MegaBytes)
CIFAR-10 Full	CIFAR-10	Object Recognition	0.17 MB
LeNet	MNIST	Handwritten Digit Recognition	0.82 MB
NiN	ImageNet	Object detection and classification	14.50 MB
Djinn ASR	Kaldi	Speech-to-text decoder	48.40 MB
AlexNet	ImageNet	Object detection and classification	116.26 MB
VGG-CNN-S	ImageNet	Object detection and classification	196.26 MB
Overfeat	ImageNet	Object detection and classification	278.30 MB
VGG-16	ImageNet	Object detection and classification	323.87 MB

Table 9: CPU and GPUs used for the comparison. (21)

Platform	Cores	Clock freq (GHz)	Memory (GB)	TDP (W)	Technology (nm)	Cost
ARM Cortex A15	4+1	2.300	2 (shared)	5	28	$191
Intel Xeon E3-1246 v3	4	3.600	16	84	22	$290
Tegra K1 GPU	192	0.852	2 (shared)	5	28	$191
NVIDIA GTX650Ti	768	0.928	1	110	28	$150
Tesla K40	2880	0.875	12	235	28	$2,599

The first evaluation taken out is in terms of resource usage, as seen in Table 10.

Table 10: Resources utilization results. (21)

	Xilinx Zynq ZC702				Altera Stratix V SGSD5				Altera Arria 10 GX115			
Benchmark DNN	LUTs (Total: 53200) Utilization	BRAM (Bytes) (Total: 630KB) Utilization	Flip-Flops (Total: 106400) Utilization	DSP Slices (Total : 220) Utilization	LUTs (Total: Utilization	BRAM (Bytes) (Total: 5035KB) Utilization	Flip-Flops (Total: 690000) Utilization	DSP Slices (Total: 1590) Utilization	LUTs (Total: Utilization	BRAM (Bytes) (Total: 6782KB) Utilization	Flip-Flops (Total: 1708800) Utilization	DSP Slices (Total: 1518) Utilization
Cifar-10 Full	61.44%	98.57%	30.77%	61.82%	85.29%	95.33%	46.90%	37.74%	84.99%	84.92%	45.85%	94.86%
Djinn ASR	42.43%	100.00%	18.25%	63.64%	53.98%	85.80%	28.12%	36.23%	68.96%	94.36%	39.27%	98.81%
LeNet	47.57%	100.00%	21.90%	61.82%	66.53%	80.44%	32.83%	33.96%	84.99%	84.92%	45.85%	94.86%
VGG_CNN_S	62.22%	97.14%	29.73%	61.82%	88.07%	78.50%	50.81%	37.04%	84.64%	89.64%	47.59%	88.54%
VGG_16	65.92%	100.00%	31.23%	63.64%	87.65%	78.20%	50.68%	37.42%	84.64%	89.64%	47.59%	88.54%
AlexNet	64.56%	100.00%	30.78%	63.64%	86.70%	77.16%	50.04%	37.23%	82.19%	86.69%	46.24%	88.54%
NiN	68.59%	100.00%	34.62%	63.64%	86.70%	77.16%	50.04%	37.23%	84.64%	89.64%	47.59%	88.54%
Overfeat	61.52%	94.29%	29.28%	60.00%	84.68%	75.07%	48.79%	36.98%	85.57%	86.25%	48.19%	88.93%

In terms of speed-up comparing to a CPU, the results are the ones presented in Figure 25.

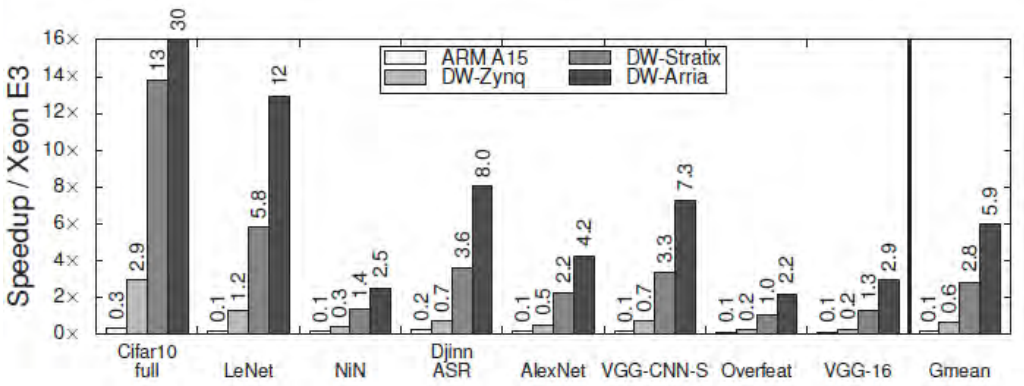


Figure 25: Speedup in comparison with a CPU (Intel Xeon E3-1246) of the 3 FPGA boards and a smartphone ARM A15 processor. Some of the networks perform worst on the CPU than the FPGA. (21)

On the other hand, comparing with a GPU (NVIDIA GTX650Ti), the following results seen in Figure 26 are obtained.

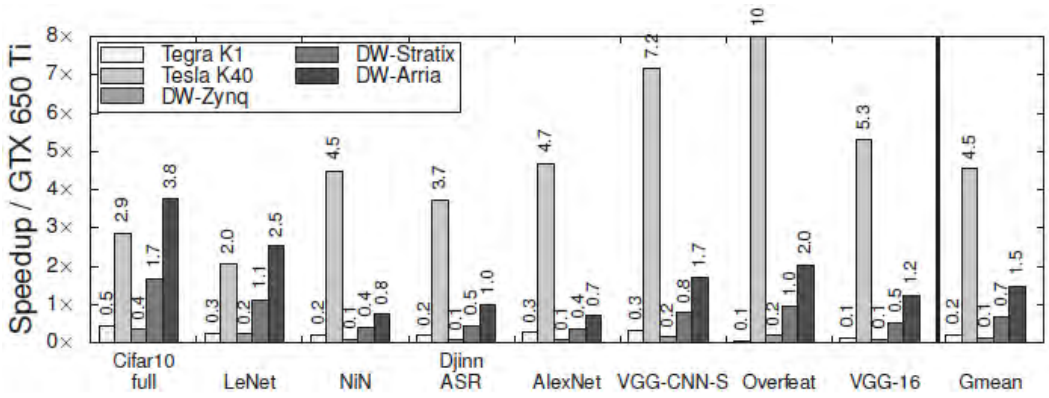


Figure 26: Speedup in comparison with GPU (Nvidia GTX 650Ti) of the 3 FPGA boards and 2 GPU's. The GPU's generally outmark the speedup results obtained with the FPGA's.

Muhammad K A Hamdan presents a vhdl auto-generation tool for CNN on FPGA (23). The tool presents a grafic interface in java in where the user can define up to 25 layers within the convolutional, pooling, fully-connected and LRN types, as seen in Figure 27.

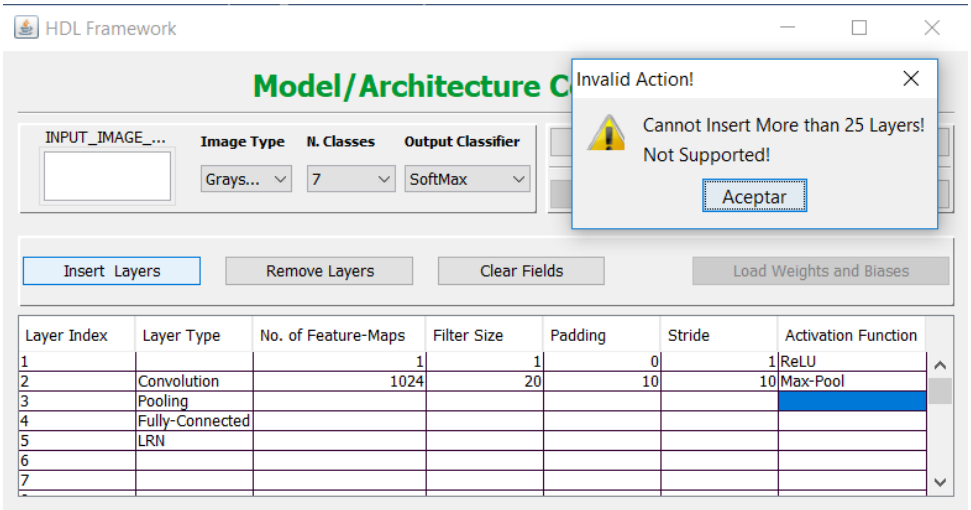


Figure 27: CNN design framework. The values that can be chosen for each column are within the range of the shown values in the screenshot. Therefore, the freedom of design that this tool offers is constrained.

However, the existing tool allows only to chose between two different board models as seen in Figure 28.

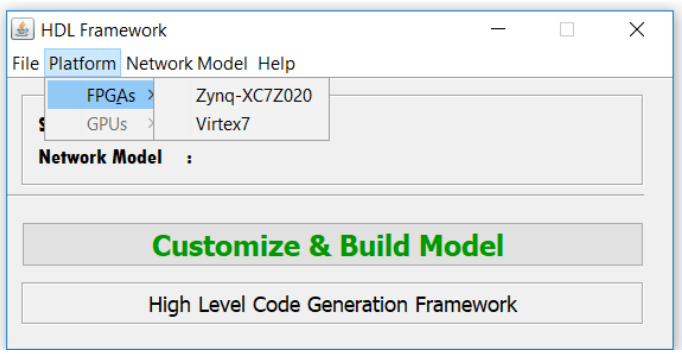


Figure 28: Zynq-XC7Z020 and Virtex7 are the 2 options that offers the program. The skeleton menu shows a deactivated option for GPUs, as well as RNN networks in the network submenu, as options to be further developed.

One programming language that is often used for CNN is Python. Therefore, having a Python framework for FPGA sound like the right idea, and that is what Xilinx did with PYNQ (24). For example, in the demonstration in NIPS of the tiny YOLO algorithm mentioned before (17), they used a bineural network based on the FINN (25) framework over the PYNQ platform.

However, there are only three officially supported boards: Pynq-Z1 from Digilent, Pynq-Z2 from TUL and ZCU104 from Xilinx. An additional community board, the Avnet Ultra96, also supports PYNQ. Besides the officially suported boards, it is stated in the PYNQ webpage that it can be used on other boards, as long as they met the following requirements:

- Any Zynq/Zynq Ultrascale+ device (including single-core)
- >=512 MB DRAM
- SD Card (>=8GB) or other bootable source
- Network connection (Ethernet or WiFi)
- UART
- USB

If the board fulfills these requirements, the image of PYNQ needs to be prebuilt. For that, an external board repository is needed, with the specs file and board specific packages.

3.2.2. Scientific discussion

From the state-of-the-art analysis several conclusions can be extracted. First of all, in the Haar-HoG-CNN comparison, the last ones are the most suitable for detecting multiple class objects. The Haar are mainly focused on face-recognition and the HoG on pedestrian detection, while CNN can be designed to detect multiple classes of objects.

In terms of CNN, the first discussion is the usage either of sliding windows vs the single shot algorithms, as YOLO. For an application in real-time as is the camera monitoring systems, fast processing times are important. Sliding windows can provide a more precise result in images with multitude of objects of different sizes, but the execution times can be of multiple seconds. As our problem, presents a limited number of objects to be detected in the region of interest for its characteristics, this high precision high detection range can be sacrificed in favor of speedup. As seen in the Table 5, YOLOv2 algorithm for images of 544x544 pixels provides a frame rate of 40 frames per second and precision of 78.6, both higher than those obtained with R-CNN (sliding windows). The second ones offer a frame rate in optimal cases around 5 frames per second, so they should be discarded for our real-time scenario, and therefore YOLO be the main focus.

Independently of the CNN chosen, the Hardware used and the Software approach, the training of the neural networks to obtain the optimal weights should be always executed beforehand in a software environment. This training can take even a week, but the classes and input objects for the training can be personalized so it is a necessary and useful time investment.

In terms of FPGA compatibilities and performances with CNN, the existing results are not yet promising. The Xilinx NIPS presentation (17) provided a frame rate of 16 frames per second but with a 50% precision. The Zhao et al. (20) got a frame rate lower than 2 frames per second with the complete version of YOLO, which improves the precision, but this frame rate would not cover the real-time requirement of this problem. Real-time is a critical requirement, and high precision detection it is not by itself in order to make a CMS functional and legal to drive around with, but the idea is to enhance the information that a simple CMS or mirror would give and therefore high precisions of =>80% would be recommended.

Comparing the resources available in the board used in the HS Ulm and those of the DNNWeaver (21) and Zhao et al. (20) papers, an approximate estimation of the behavior that this board could provide can be obtained.

Table 11: Comparison of the resources the different board equipped. In blue, the ones used in DNNWeaver paper, in green the one used in HS Ulm and in orange the one used in Zhao paper. The ZCU106 board should provide a slightly better performance than the Altera Arria.

	Xilinx Zynq ZC702	Altera Stratix	Altera Arria 10 GX115	Xilinx ZCU106	Xilinx Zynq ZC706
LUTs (K)	53	172	427	504	218
Flip-Flop (K)	106	690	1708	460,8	437
BRAM (KB)	630	5035	6782	11000	19200
DSPSlices	220	1590	1518	1728	900

It can be inferred that the ZCU106 would provide a behavior slightly better than the Altera Arria board and that the ZC706 a behavior better than the ZC702 and close to the Altera Stratix one. In Figure 25, Altera Aria provided a speedup 10 times bigger than the ZC702 and 3 times bigger than Altera Stratix, so it could be approximated that the ZCU106 could provide a speedup 4-5 times bigger than the ZC706. As the ZC706 offered with YOLO a frame rate near to 2 frames per second, that means a performance around 8-10 frames per second. Those numbers could be enough for the object detection, if the real capacities of the board meet this approximation.

In order to find the real capacities of the board, a CNN system should be implemented on it. The CNN vhdl automation tool (23) offered a too constrained tool, and after trying the demo code on Vivado 2018.3, the resulting project synthesized and compiled but produced no behavior at all. The schematic showed just a row of output buffers connected to VCC and GND. Being either due to the under-development version of this tool or an incompatibility with the Vivado tool, it would not seem the best option to implement the desired CNN.

Considering that Python is the most intuitive and easy-access tool for CNN, using PYNQ platform seems the best approach. However, the board used in HS Ulm (ZCU106) is not one of the officially supported boards and consequently running on it the PYNQ environment would require building an image for this board, if possible.

Even if the goal of this project is approaching the usage of FPGA boards for CNN, both papers of DNNWeaver (21) and Zhao et al. (20) present way better results with the usage of GPU's, which also happen to be way cheaper than a FPGA with similar results. Still, the FPGA outcomes GPU's in terms of power consumption and this could be the main advantage on it.

3.3. Low light algorithm

3.3.1. Differences towards day-light situation

In low light situations, the information that presents the input image obtained through a daylight camera is mainly light sources, either from vehicles, street lights, building windows or even the moon, as seen in Figure 29.

Figure 29: Sample of low-light situation. The light sources and its reflections are the only visible thing. Besides the two frontal lights of a vehicle, three street lights are visible. The artifacts that appear in this picture won't be present in a CMS as the cameras and preprocessing used in those systems reduce the presence of those.

Therefore, as the features that could be detected by the daylight situations are not captured due to the lack of light, the same algorithms can't be used. Moreover, pedestrians are not detectable as they don't present any source of light. However, low-light scenarios can't be ignored, as it is in the night time when visibility for drivers is the lowest and more risk of suffering an accident exists.

Even if detecting which concrete vehicle is approaching would be a complicated issue, the sole detection of an object approaching will provide an increased input of information compared to the conventional mirrors. Moreover, the blinding caused often by the light haze of the following vehicles would be reduced.

Another complication in the low-light scenarios compared to the daylight ones, is the appearance of reflections on the ground and the presence of irrelevant light sources as the streetlights or tunnel lights might be. In consequence, the implemented algorithm should be able to discern the relevant information.

3.3.2. State of the art

Most of the existing vehicle detection systems and methods mainly focus on vehicle detection on daytime light conditions with a daylight camera sensor as an input. However, in low-light conditions this kind of cameras obtain pretty bad quality images, as seen in Figure 30.

In order to work on low-light scenarios, Wang et al. (26) proposed using infrared cameras and apply deep networks classifiers as is usually done in daylight scenarios. The results showed an accuracy of 92% and a frame rate of 25 frames per second.

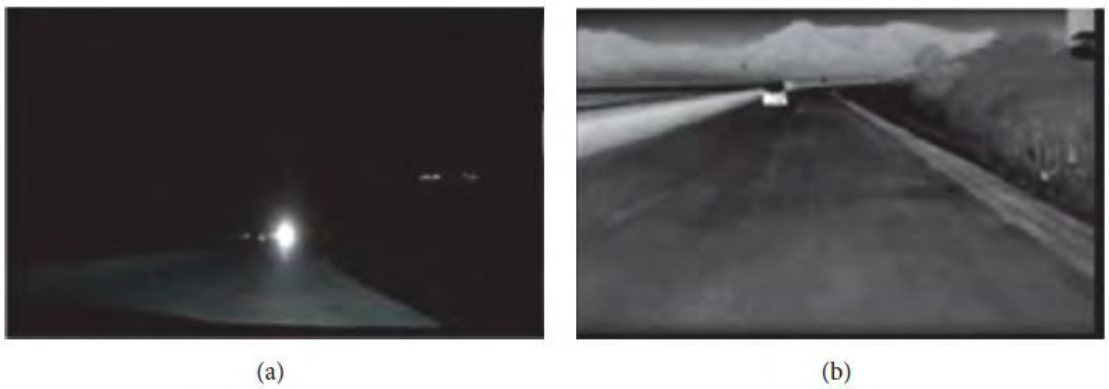


Figure 30: a) Image obtained with a daylight camera in night conditions. Only a light of a vehicle approaching can be distinguished b) Image obtained with an infrared camera in night conditions. A vehicle is clearly recognizable. Source: [26]

However, the scope of this project was to use a single camera for all-kind of light scenarios. Lopez et al. (27) focus on the single camera as well, in this case to design an intelligent light beam control. This project included detecting vehicles by detecting their light sources. Reflections on the floor and street lights made the detection in a single frame harder, so they added a temporal coherence analysis to detect objects in multiple frames, by analyzing the steadyness of the detection confidence on consecutive frames. With this technique they achieved a precission of 96 % and a frame rate of 5 fps when using only multiple frame detection and 50 fps when using only single frame detection.

The source light detection was completed by detecting blobs (binary large objects), seen in the image as white objects on a black background. Basing on this approach, blob detection, a self-created algorithm is designed in the next chapter.

3.3.3. Algorithm design

A demo implementation of the low-light algorithm in Matlab is presented in the following lines. Designing it in Matlab would allow to easy test, correct and optimize the algorithm concept and functionality, using the multiple predefined functions for image processing that the program offers. But of course, the goal is to implement it on a FPGA. Matlab have an add-on that theoretically generates the vhdl code from a Matlab code, and therefore it could ease the translation from one language to another.

As the low-lights images will basically show a point light sources on a dark background, the problem can be converted into a circular object detection problem. A similar approach to the coin detection algorithm that is often used for starting using image analysis in Matlab can be done.

The first step, should be downsizing the image to reduce the processing time. Next step should be converting the input image into a binary image, just black and white, as seen in Figure 31. First, the image is converted to grayscale and then a threshold is applied to binarize the image.

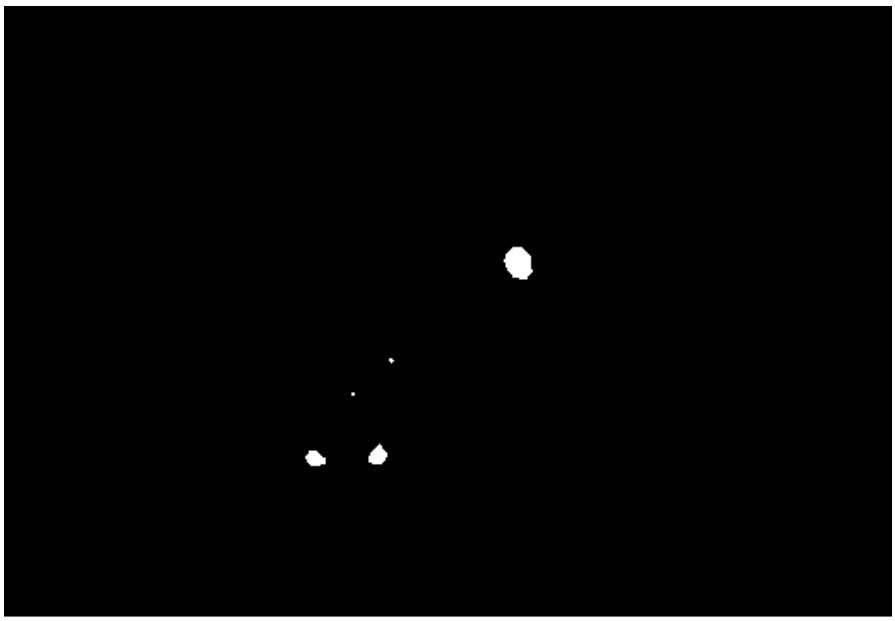


Figure 31: Binarized sample night image. At first sight the two vehicle lights can be easily recognized. In addition, the street lights and reflections on the ground cand still be seen and therefore further processing to eliminate them is needed.

In order to eliminate undesired small lights, erosion is applied to the image obtaining the image shown in Figure 32.

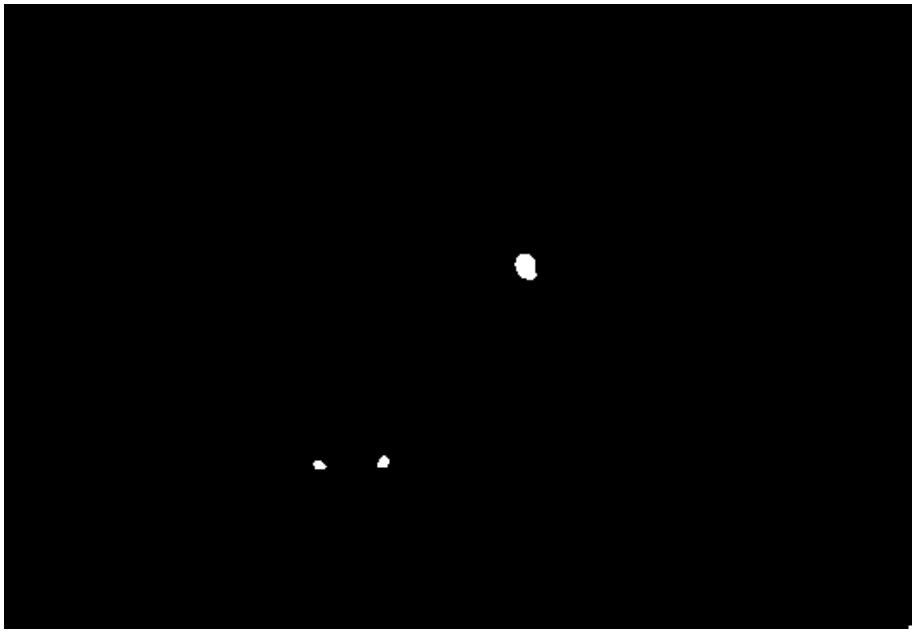


Figure 32: Eroded sample night image. The small point lights as the far street lights are deleted.

The interest lights also had been eroded. Therefore, for better detection a dilation is applied to recover the original size of the lights, as shown in Figure 33.

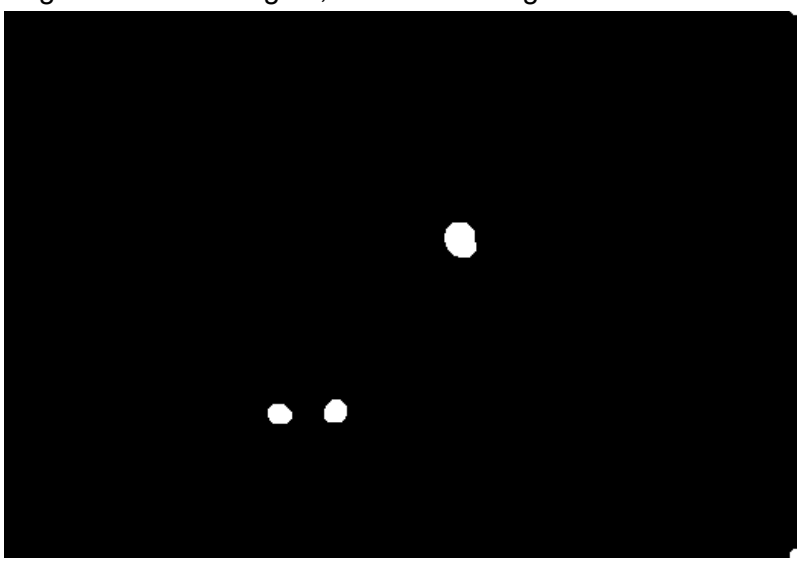


Figure 33: Sample night image after dilation.

Then the image is ready for light detection. In order to do that, the Matlab function ‘regionprops’ finds the diameter and center position of the circular shapes in the image. The function itself returns the major and minor axis, therefore the radius needs to be calculated from it. With the center position and the radius, the Matlab function ‘viscircles’ overlays a frame to the detected lights as seen in Figure 34.

Figure 34: Final image with the overlaid detected point-lights. As seen, a street light is also detected, which is not a relevant information for our problem and causes a false positive.

The street lights are also detected on the image, and this is not a relevant information. As in the daylight algorithm, a region of interest needs to be defined to dismiss irrelevant objects that would cause false positives. Like that we detect only the objects of interest, as seen in Figure 35.

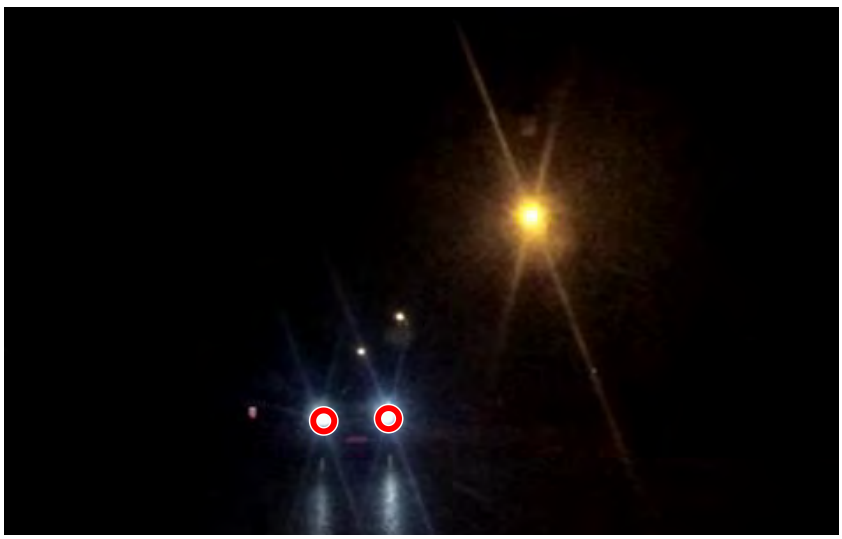


Figure 35: Sample processed image after applying the region of interest restriction.

As point lights are detected, final step would be detecting the vehicle itself. The biggest problem for this situation are the motorcicles, as it would be hard to determine if it is one of those vehicles or a single light of a car is being detected. A car can be determined by having two lights of similar radius in the same horizontal line, as seen in Figure 36.

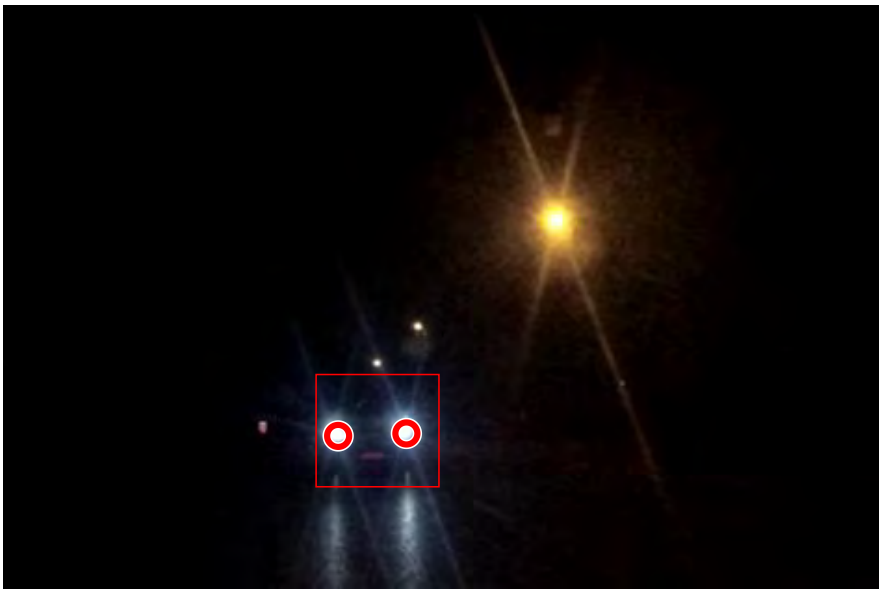


Figure 36: Bounding box around the vehicle, using the lights as reference.

3.3.4. Testing and results

Lowlight scenes were recorded, more specifically night situation, mainly in highway and a few instants of urban environment. The recording started at dusk and lasted until dark night. The algorithm was tested over cuts of this video, first only using the point light detection and afterwards adding the vehicle detection estimation. A set of 10 samples of different situations of these videos are shown in the following pictures, some of which present some flaws of the algorithm to be corrected.

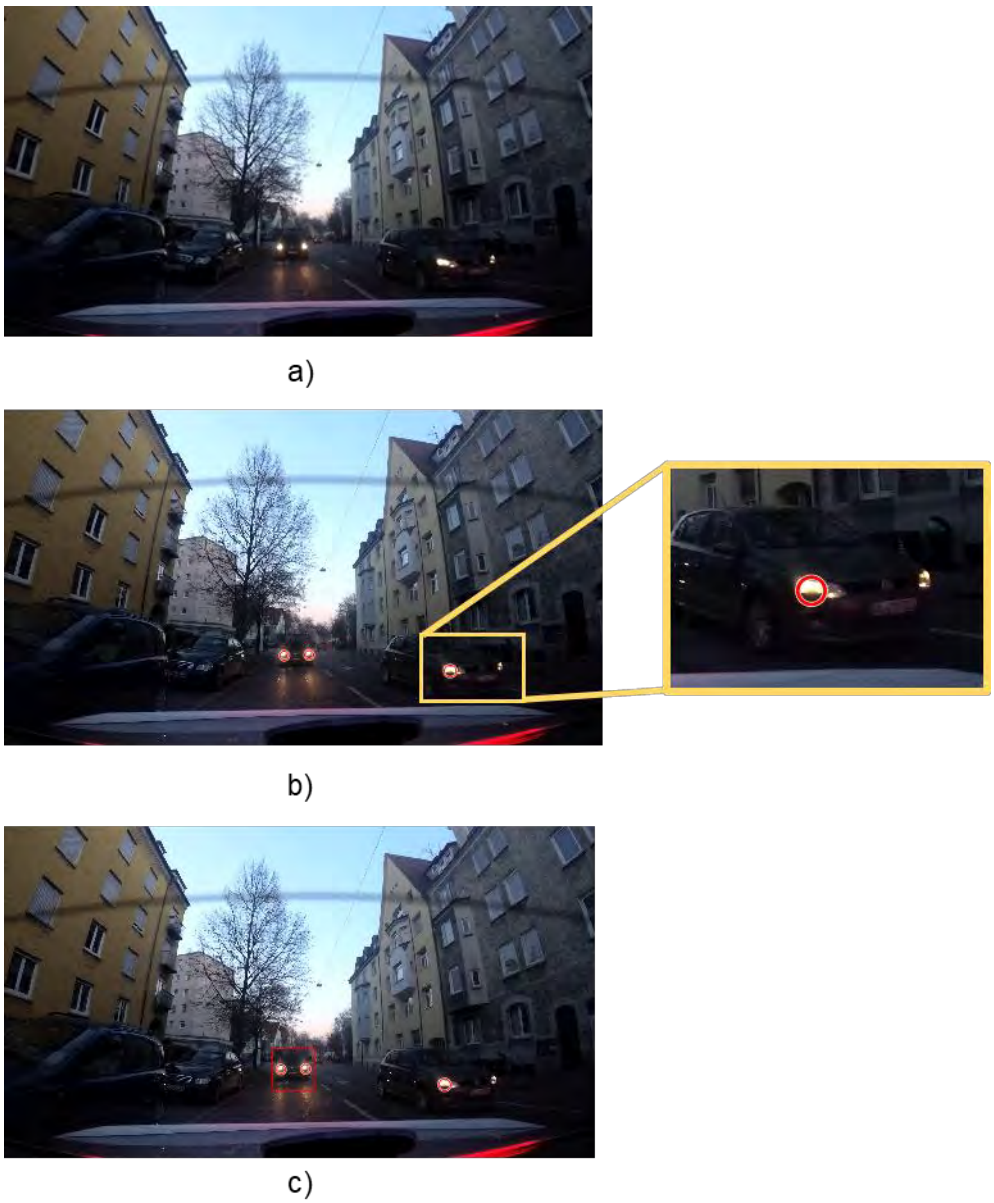


Figure 37: In this scenario the level is high enough probably for using the daylight algorithm, but the cars already use lights and therefore a test of the low-light one can be applied. a) urban scene. b) point light detection. The right light of the neighbour car is not detected. Due to be the farther one and in a position of overtaking the reference car, the intensity of this light is low. c) the vehicle detection logically only detects one couple of lights, therefore one car.

In the first scenario shown in Figure 37, which is in the boundaries of daylight scenarios, 3 lights out of 4 are detected and in consequence only bounded 1 car out of 2. In terms of approaching danger information, the system correctly enhances that there is a car behind and an undefined light source close next and behind the reference car. Fine tuning the binarization threshold or the erosion/dilation parameters, could improve the detection of the missing light. When being overtaken, the farther light disappears from the point of reference field of view. Therefore, a better way to keep the detected object overlaid even when the angle for the FOV darkens one of the point lights, would be adding object tracking as in the previous instants the complete car would be appearing in the image. To do that, memory should be added storing the previous bounding boxes and source point lights position and sizes, movement prediction through motion vectors could be applied to predict next position of the bounding boxes, error correction of the prediction and the actual detection should be applied and extra logic to enable bounding boxes where single light points are detected should be added. But in terms of this thesis, the goal was to approach the ways to implement those algorithms first, a complete optimal design of this single low-light algorithm would require more time resources and escapes the boundaries of this thesis.

Table 12: Figure 37 detection results.

	In image	Detected	Missed	Accuracy
Point lights	4	3	1	75%
Vehicles	2	1	1	50%

Although the object detection is not compulsory for CMS and is just an upgrade in terms of information providing, the accuracy should be of at least 70 % in a conformist approach. If the system also connects and affects ADAS, which could mean directly interact with brakes and direction, this accuracy should be close to 100 %. Therefore, the values obtained in Table 12 would not be enough for ADAS system, and would be poor for information enhancement system, despite that highlighted the closest dangers.

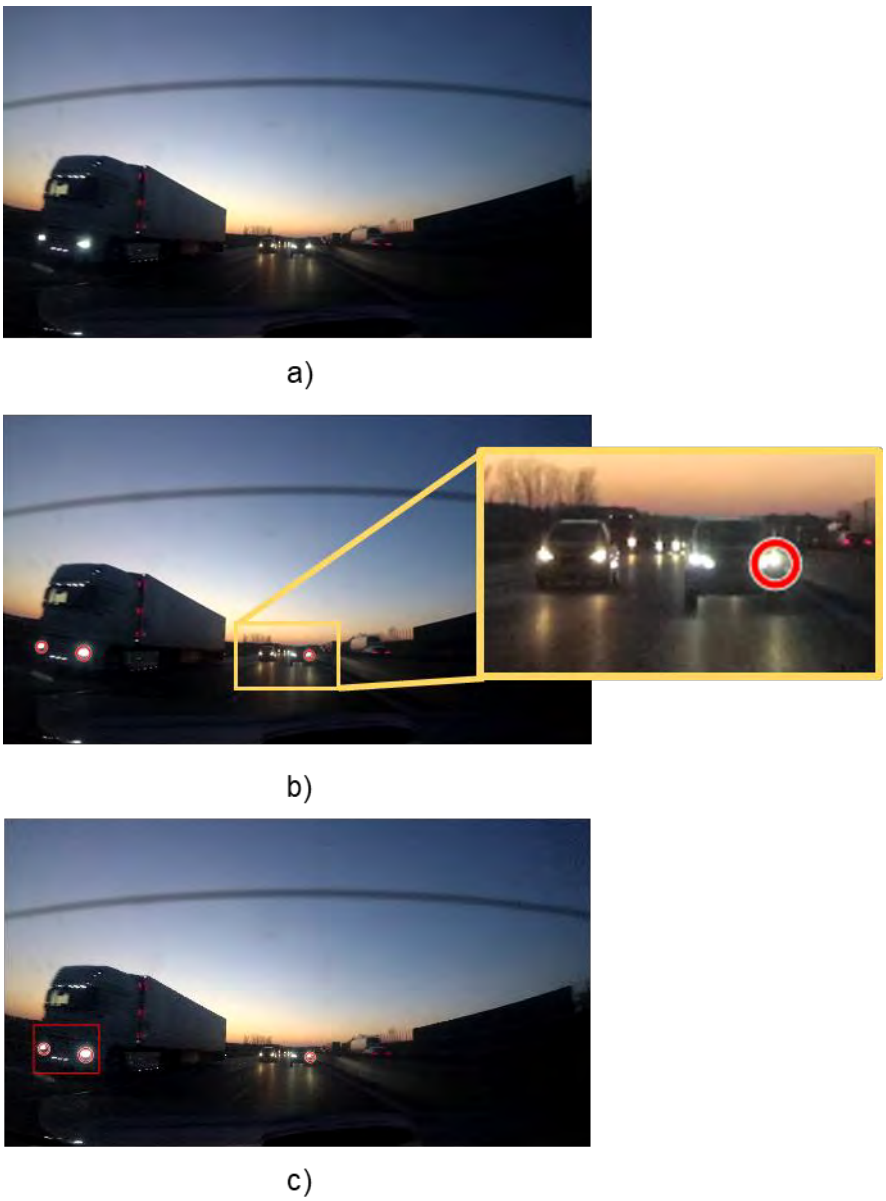


Figure 38: This scenario is also in the impasse between daylight and low-light situations. a) Highway dusk scenario b) Bad detection precision in this case. The farther behind car lights are not detected and being a highway, they should. c) The closest danger, which is the truck, is detected. Farther vehicles are not.

Table 13: Figure 38 detection results.

	In image	Detected	Missed	Accuracy
Point lights	8	3	5	38%
Vehicles	4	1	3	25%

The results obtained in the highway dusk situation are really bad. Even if the biggest danger is detected, as is the truck closely behind, the farther cars that might be approaching at high speed as is a highway are not even detected.

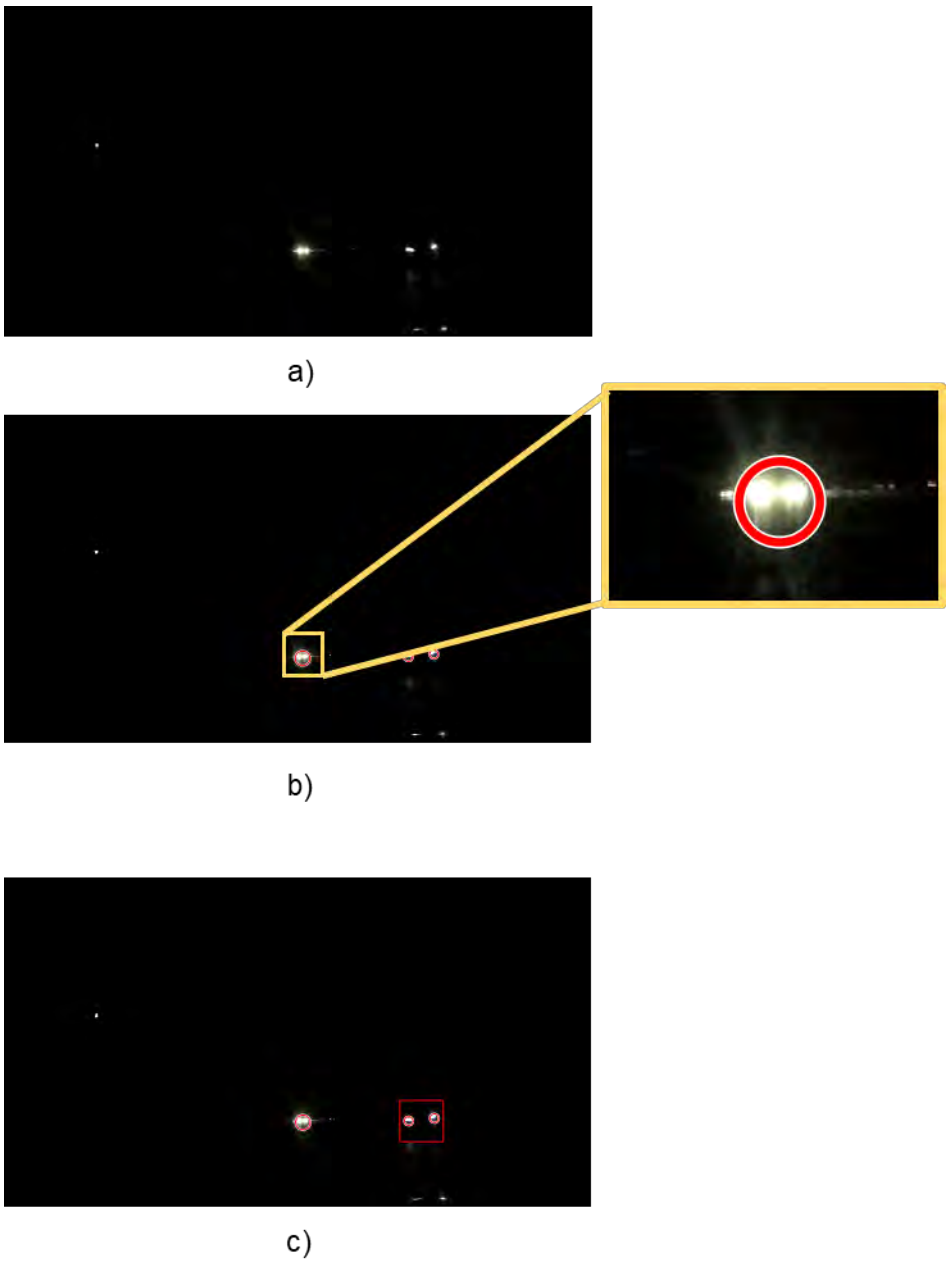


Figure 39: a) Highway night situation with a close car overtaking and a far car behind. b) The farther car lights are detected as one. c) The close car overtaking is rightly bounded, not the far one.

Table 14: Figure 39 detection results.

	In image	Detected	Missed	Accuracy
Point lights	4	3	1	75%
Vehicles	2	1	1	50%

In Figure 39 scenario, the most dangerous object which is the overtaking car is detected, but the farther vehicle is wrongly detected as a single light instead of two of them. Even if the fact that a car that far is detected, not like in the dusk situation, the single like light detected is bigger than those of the close car and it could wrongly trick the driver or ADAS system to think that the car behind is closer than it happens to be. New logic to distinguish this situation should be added to the code or either it should be considered the car is far enough not to be a danger and therefore do not enhance it, to avoid misinterpretations.

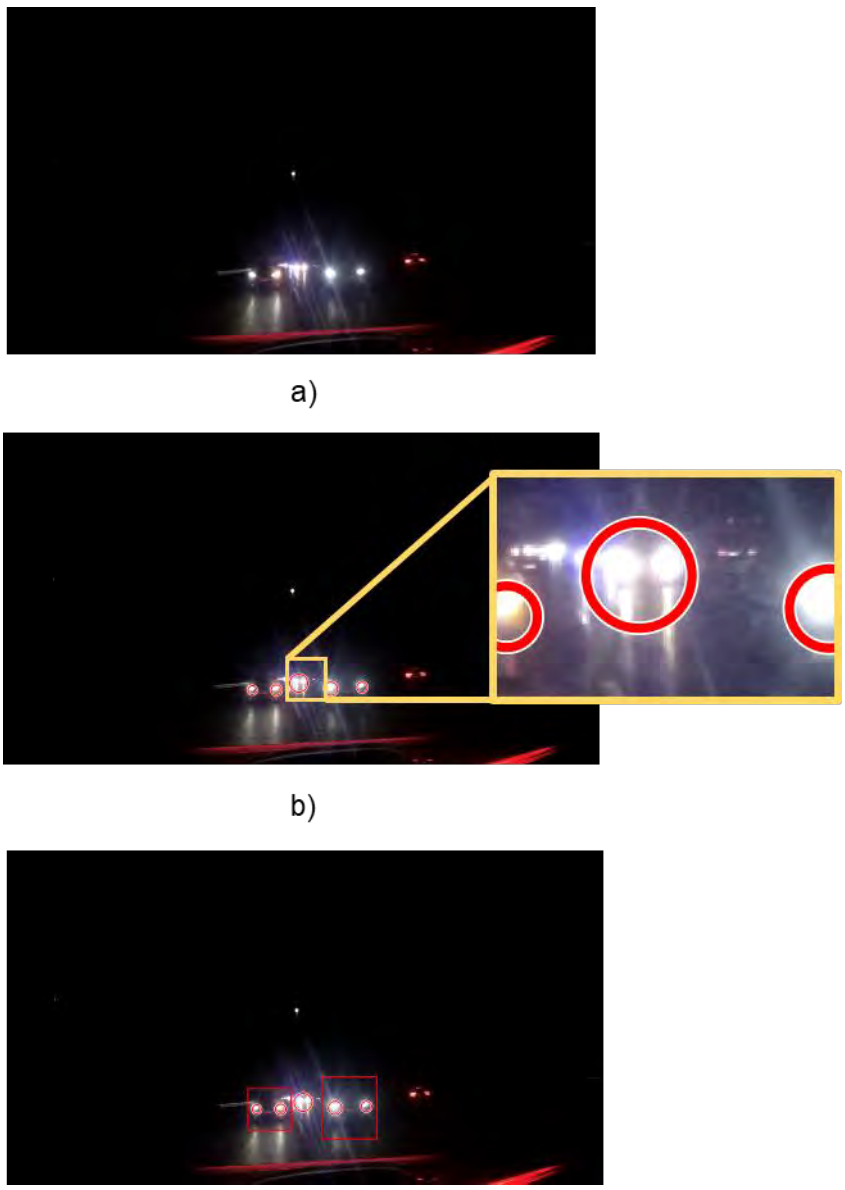


Figure 40: a) Highway close cars behind scenario. b) The farther car behind, which is not a potential danger yet as there is a second car between the reference car and this car, is detected as a single light. c) The close cars are correctly detected.

Table 15: Figure 40 detection results.

	In image	Detected	Missed	Accuracy
Point lights	6	5	1	83%
Vehicles	3	2	1	67%

As in the previous scenario, the far car lights are detected as a single one, which what could lead to misinterpretation. The potential dangers, which are the cars closely behind, both of them are correctly detected and therefore the system works properly.

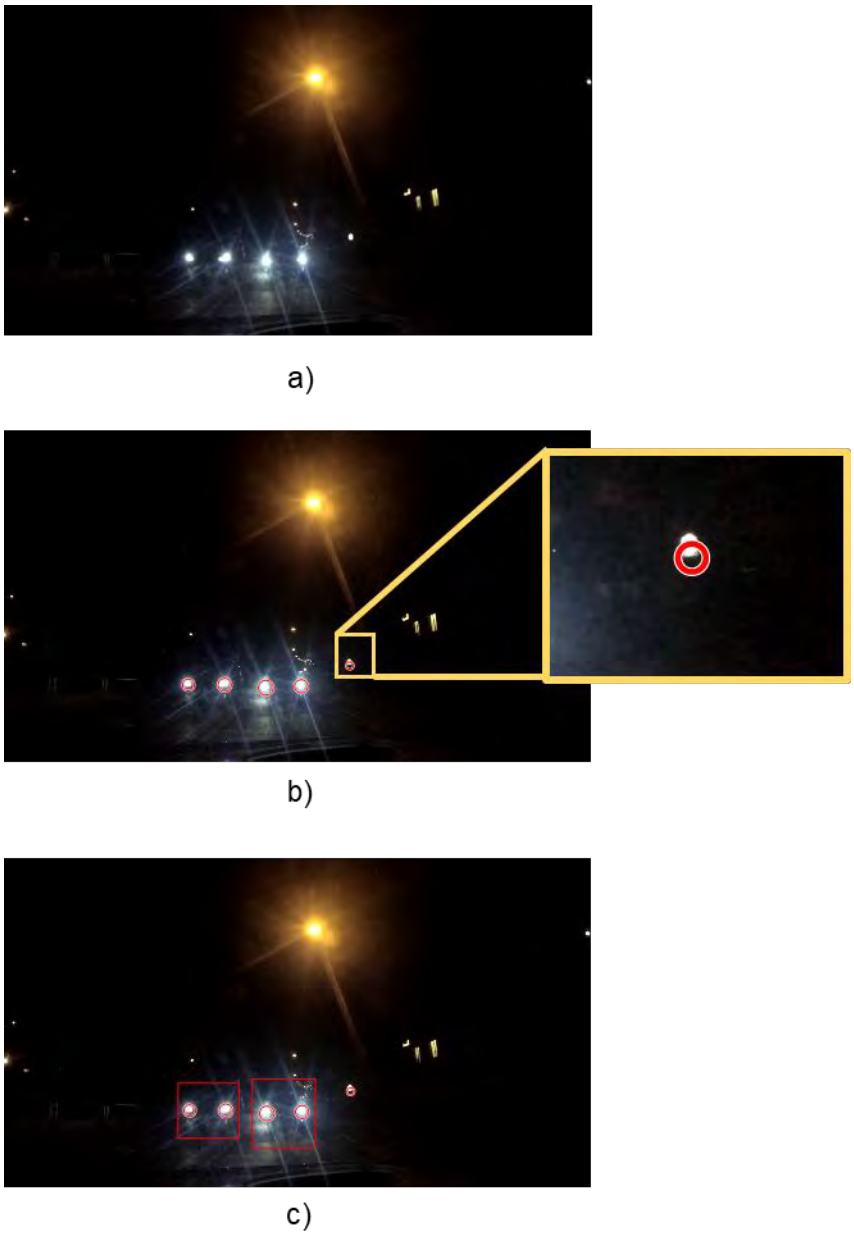


Figure 41: a) Interurban road with traffic lights and house windows lights. b) A house's window is mistaken by a vehicle light c) The closely behind are correctly detected.

Table 16: Figure 41 detection results. Presence of a false positive

	In image	Detected	Missed	Accuracy
Point lights	4	5	-1	125%
Vehicles	2	2	0	100%

In this case the vehicles are correctly detected, but a false positive occurs as the system detects a house window as a vehicle light.

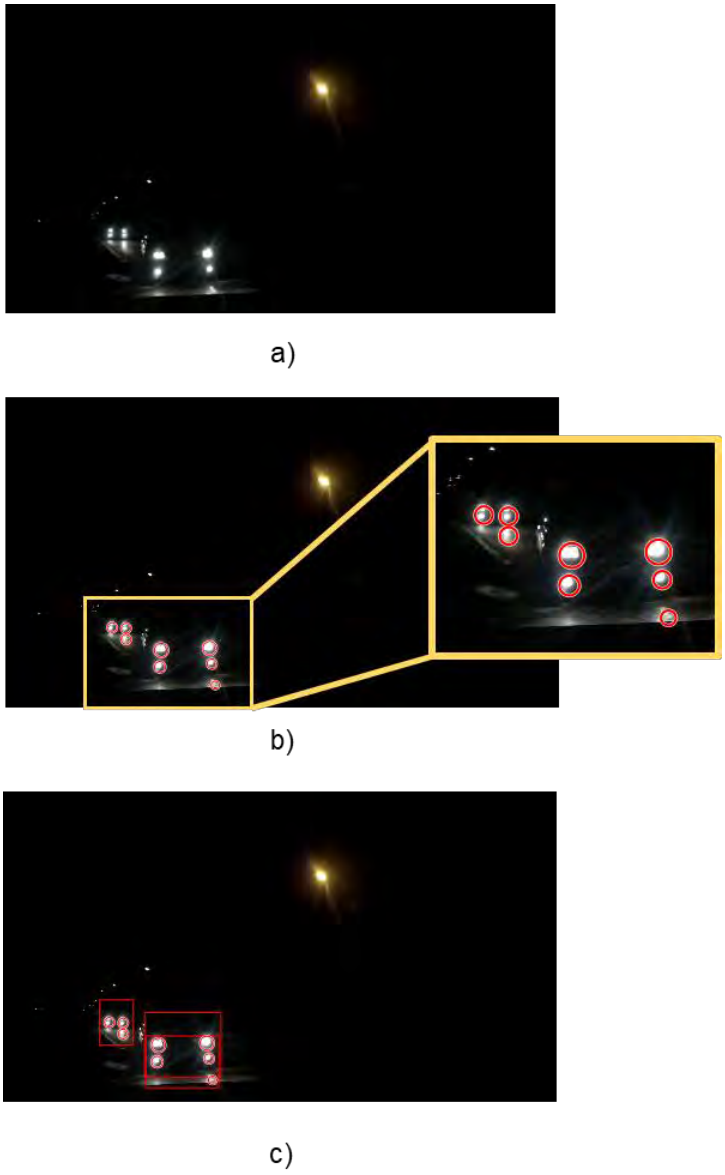


Figure 42: a) Slight turn close cars situation b) Some cars include extra lights for when taking a turn. Those extra lights and the reflection that generate as are closer to the ground are also detected by the system. c) The system detects correctly the cars behind but includes extra bounds to the vehicles due to the car.

Table 17: Figure 42 detection results. 2 false positives.

	In image	Detected	Missed	Accuracy
Point lights	8	10	-2	125%
Vehicles	2	2	0	100%

In this scenario, two false positives are generated due to the ground reflections of the light's sources. A second bound is added to a single vehicle due to the extra turning lights. With the fog lamps could happen as well, therefore an extra logic for these situations should be added to the code to keep it just a single bounding box. The dangers are correctly detected.

a)

b)

c)

Figure 43: a) Bus behind in interurban road. b) Correct detection of the lights c) Correct detection of the vehicle

Table 18: Figure 43 detection results.

	In image	Detected	Missed	Accuracy
Point lights	2	2	0	100%
Vehicles	1	1	0	100%

In this scenario the vehicle is correctly detected.

a)

b)

c)

Figure 44: a) Bus and car in parallel b) Correct detection of the lights c) Correct bounding of the vehicles.

Table 19: Figure 44 detection results.

	In image	Detected	Missed	Accuracy
Point lights	4	4	0	100%
Vehicles	2	2	0	100%

In this scenario the vehicles are correctly detected.

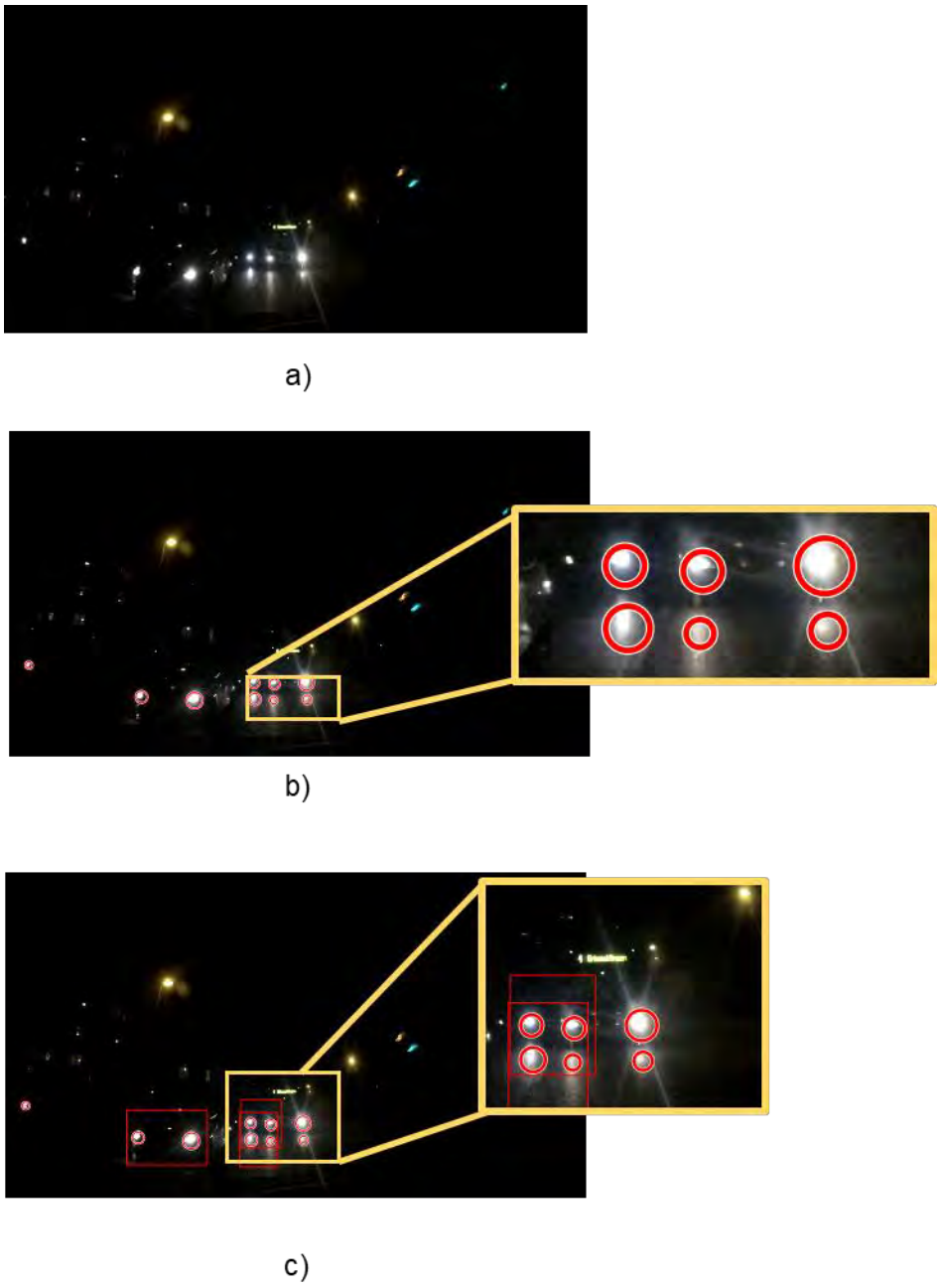


Figure 45: a) Multiple vehicle urban scenario. A car is partially blocked and only one lights seen. b) Reflections on the ground are detected c) The partially blocked car light and one of the bus lights are mistakenly bounded as a single vehicle, while the bus is not

Table 20: Figure 45 detection results. 3 false positive.

	In image	Detected	Missed	Accuracy
Point lights	6	9	-3	150%
Vehicles	3	1	2	33%

In this scenario again reflections are detected on the ground. Some extra processing should be done to avoid them. Even though, as the reflections are close to the lights the effect is a bigger bounding box which translates in the perception that the potential danger is closer than it really is, which is better than not detecting at all. The lights of two close vehicles, the car and the bus, are detected together as a single vehicle. With object tracking, this situation could be avoided, as is seen in Figures 43 and 44 the bus has been detected previously.

One situation that was not tested, as there were no motorbikes at all circulating around, is the detection of those. The single point lights would be detected, as happened with partially blocked cars, but hardly would be a way to decide if it is a bike or the only visible light of another vehicle. However, detecting the source lights and overlaying them to the image is already and enhancement of the potential objects approaching, and indistinctly of it is a motorbike a car or a truck, in neither case the driver would want to crash into them.

From the results obtained the following issues need to be corrected or added:

- Reflection detection: Preprocessing to reduce the effect of reflections, if possible, should be applied.

- Object tracking: Movement estimation to track the objects should be added.

- Multiple light treatment: As some vehicles have extra lights for turning or fog situations, the treatment of those as part of the same vehicle should be implemented.
- Distance estimation: From the radius and bounding boxes size, an estimation of the distance could be calculated.

As well other low-lights situations like tunnels should be tested, to detect additional flaws of the algorithm. For a first simple code approach to the low-light algorithm, the behavior of it provides a relevant information addition to the CMS. Next step before upgrading it, should be testing the conversion to VHDL and its implementation on FPGA.

3.3.5. Conversion to VHDL

Matlab presents an add-on oriented to converting matlab code to vhdl.

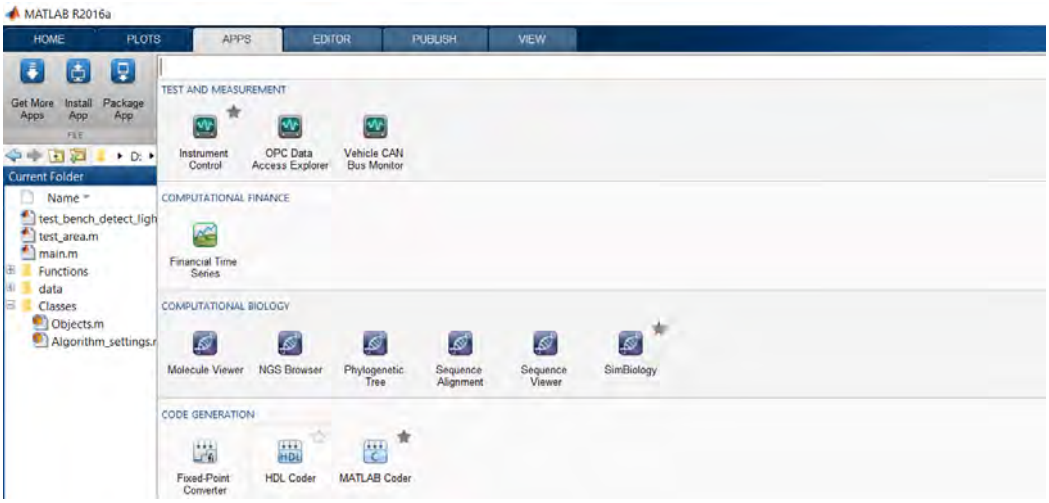


Figure 46: Matlab screenshot. In Apps/Code Generation, th HDL coder can be found.

The coder requires a function to be converted and a testbench. This project had four different functions as seen in Figure 47.

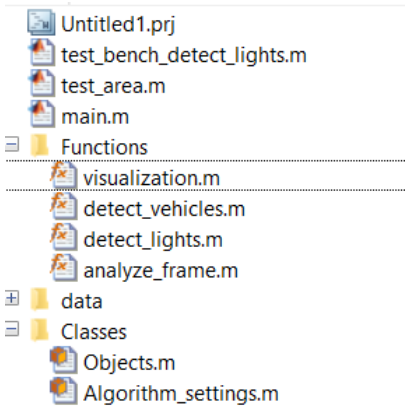


Figure 47: Folder distribution of the Matlab project. Separation in Classes, Data and Functions.

The most important function is ‘analyze_frames’ which is the one returning the centers and dimensions of the circles and bounding boxes detected. This function calls ‘detect_lights’ and ‘detect_vehicles’ functions. The ‘visualization’ function is just for displaying on matlab and therefore it has not to be converted to vhdl.

So, the input to the converter is the ‘analyze_frames’ function and the ‘main’ function as the testbench. However, the coder has some restrictions in terms of code and function that can be used as input.

At first instance, the project was using classes and the coder does not support this kind of structures. In consequence, all the settings constants had to be hardcoded to the main function. Once this problem was solved, the use of tables was flagged out as incompatible. The ‘regionprops’ function that detected the light objects can present either the results in a table or a structure, so the second format was used to overcome this problem.

When all the code compatibilities were solved, an unbeatable problem appeared. The Matlab coder for hdl requires the target board model to keep on with the conversion. The Matlab version used dated from 2016, while the board targeted dates from 2018, and therefore doesn’t appear in the board catalog that Matlab had. A 2018 version of Matlab was installed, but it did not come with the hdl coder add-on, and therefore the conversion couldn’t be completed.

4. Overall concept

In order to provide a complete CMS with object and tracking detection, the methodology needed for the CMS object detection design is presented. This thesis presents a proposal in terms of minimum value needed to cover the proposed problem, basing on the fact that the processor resources are limited. However, more ideal solutions are mentioned.

The first thing to take into consideration is the multiple scenarios in which a vehicle can circulate. In terms of light, we can meet with the situations represented in Figure 48.

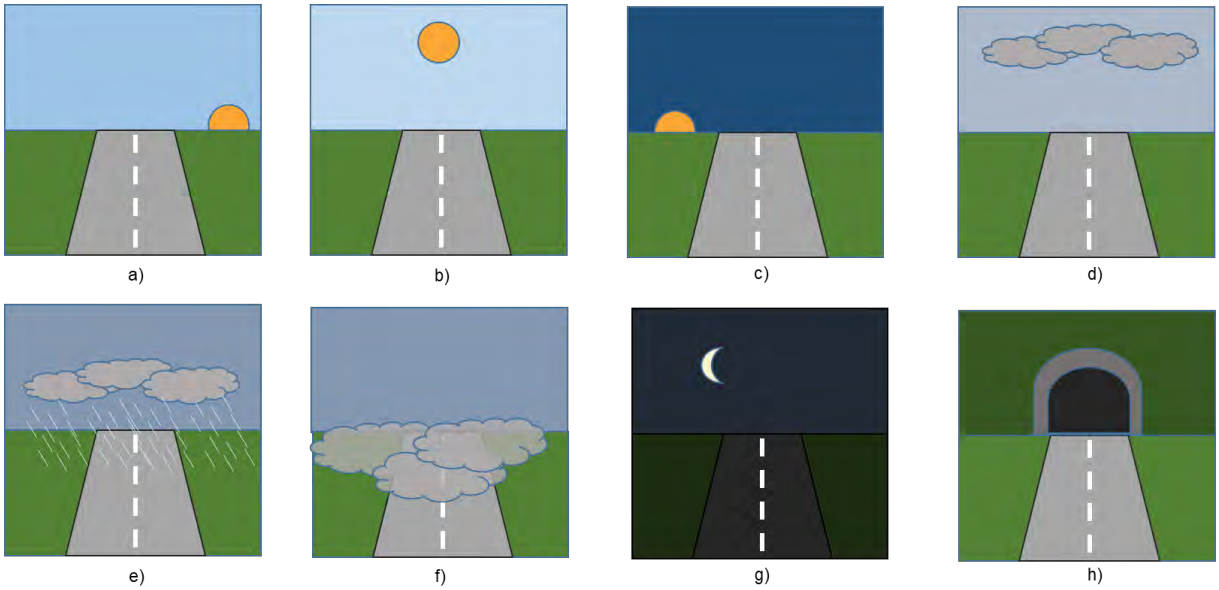


Figure 48: Different light condition scenarios. a) Sunrise b) Daylight c) Sunset d) Cloudy e) Heavy precipitation (rain either snow) f) foggy g) night time h) tunnels.

Sunrise and sunset can be a problem when the sun is behind the car, but with the right lens and image processing for diminishing the glare it is possible to obtain a cleaner image and indirect vision than with conventional mirrors, as in those situations many times the sun reflection can be blinding.

Low light situations as night time or tunnels should be treated differently than daylight situations, by means of a different camera device or algorithm approach as visibility is considerably reduced.

The more challenging scenarios are those of heavy rain and fog, as the visibility is extremely reduced to the point of barely being able to see the vehicles farther than a couple of meters of the own vehicle. In this case, a safer solution would be using radar or laser scanning instead of camera systems, but is more expensive solution and out of the scope of this thesis.

In this project, the clear daylight and low light scenarios are considered, ignoring the other scenarios. For a complete designing of a CMS system all the scenarios should be considered and tested.

The way to approach the multiples scenario problem is to use multiple tools in our system, either in terms of hardware devices or software algorithms, as shown in Figure 49.

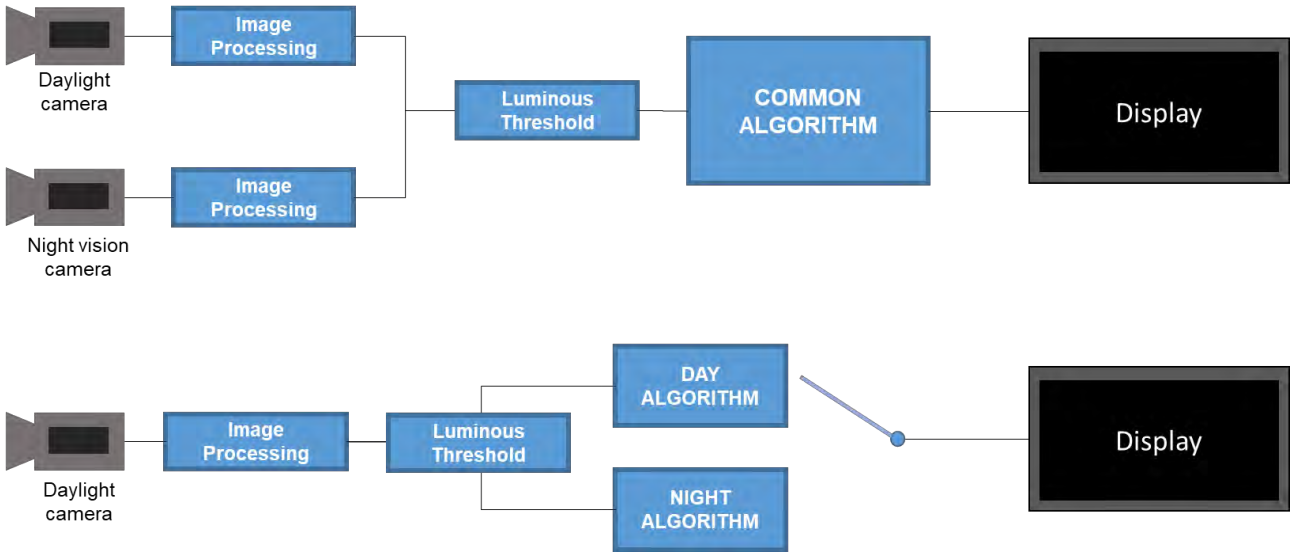


Figure 49: Dual vs single camera systems. The top schematic shows a proposal of a CMS system using a low-light vision and a daylight camera, using on both cameras the same algorithm, the daylight one. In order to choose which input to process, a luminous threshold could be used. If processing resources are not a limitation, both images could be processed and the best predictions used or combined. The bottom schematic shows a single camera input proposal, in which a luminous threshold determines which algorithm to use, if the daylight or low-light situation. If processing resources are not a limitation, as many algorithms as required for the different scenarios could be used.

For this thesis the second scenario has been considered, using two separate algorithms for day and low-light situations. Using a single camera would be economically cheaper and easier in terms of installation. Moreover, already existing parking camera that many cars include could be used as well for this purpose.

The complete design of the luminous threshold is not part of this thesis, but the following considerations should to be taken into account while designing it:

- Duration of the light change: Light fluctuations might happen due to shadows while crossing under a bridge, driving through a forest or high buildings in a city center. The system should not jump between algorithms unless the light change is long enough to consider a change of scenario.

- Hysteresis cycle: As many other sensor-based systems, like air conditioning, an hysteresis cycle needs to be applied so the system does not constantly jump between algorithm when the luminosity is close to the threshold value.

The next step, as an object detection and tracking algorithm, is to define the objects that we want to detect. The basic objects to be detected are:

- Pedestrians
- Bicycles
- Cars
- Buses
- Trucks
- Motorcycles

In this thesis framework, where a single camera is used, for low-light situations it wouldn't be possible to detect pedestrians. For the other objects, it would be easy to detect their lights but harder to classify them into the different vehicle types.

The objects that we want to detect can be found in different environments. The speeds and risk situations that can be reached and found in a highway or inside a city differ considerably.

In interurban roads mainly motorized vehicles like cars and trucks can be found in this environment, by means of being mainly transportation paths. This includes highways in which pedestrians and cyclist are not allowed and shouldn't be the main objects to be detected, while high speeds are reached and therefore a higher frame rate and being able to detect smaller objects (vehicles approaching from far) is more critical than the amount and variety of objects to be detected.

As represented in the Figure 50, the most dangerous situations to check with the rear-view elements on a highway are the change of lanes and safety distance with the car straight behind, although that depends more on that car that in the will of the on-board driver.

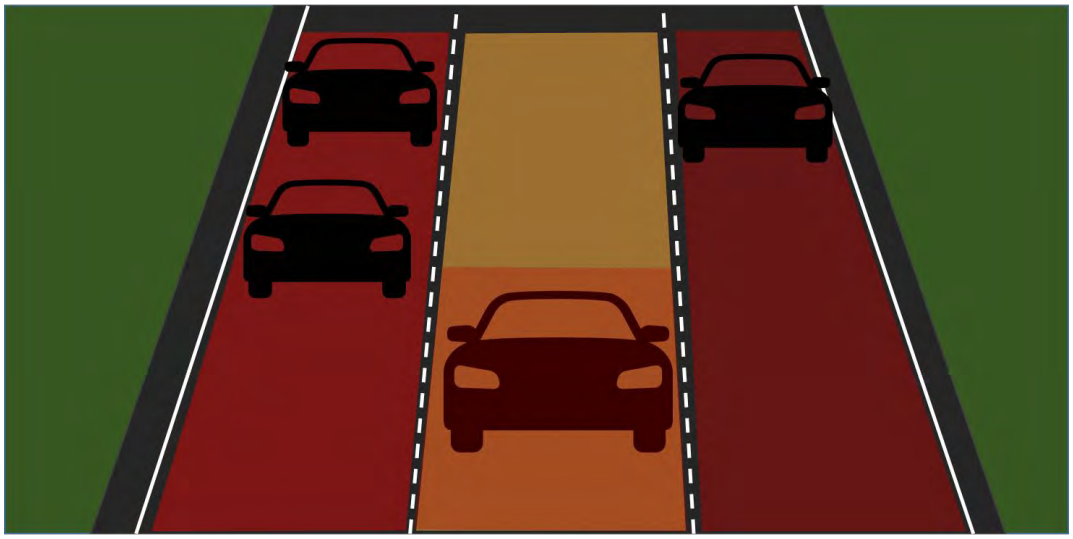


Figure 50: Considering the view from a vehicle circulating in the middle of a 3-lane highway, the dangerous areas for the driver are the adjacent lanes, in case the driver wants to change the lane, and the immediate posterior car in case a sudden breaking is needed.

Considering the safety distance that two vehicles should keep, which is a thumb-like rule of 2 seconds, the approximately distance between 2 consecutives cars can be calculated. For a speed of 120 km/h, which is 33 m/s, would suppose a distance of 60 meters. Covering that distance on the indirect field of view, the regulation requirements are met. Therefore, detecting the immediate behind vehicle, considering a vehicle might be almost at the same level side-by-side and the vehicles behind them, would be enough to offer a safe vision to the driver. In other words, **5 vehicles** to be detected on this sort of roads would be an acceptable measure.

On the other hand, the velocities in the highway are high, therefore the frame rate of the detection algorithm should be high enough to detect the approaching vehicle. In the extreme case of a German autobahns, where the biggest relative speed differences might happen, a vehicle could be approaching 90 km/h faster than the preceding car. That is 25 m/s. It would sound a reasonable measure, per every frame actualization would imply not more than 2 meters advance by the previous car, that is **15 fps** in a highway scenario.

Urban areas present an opposite situation, a lot of objects to detect from different classes as seen in Figure 51, at low speed. Pedestrians and bicycles are critical in this scenario, due to the danger that supposes for those to collide with a motorized vehicle.

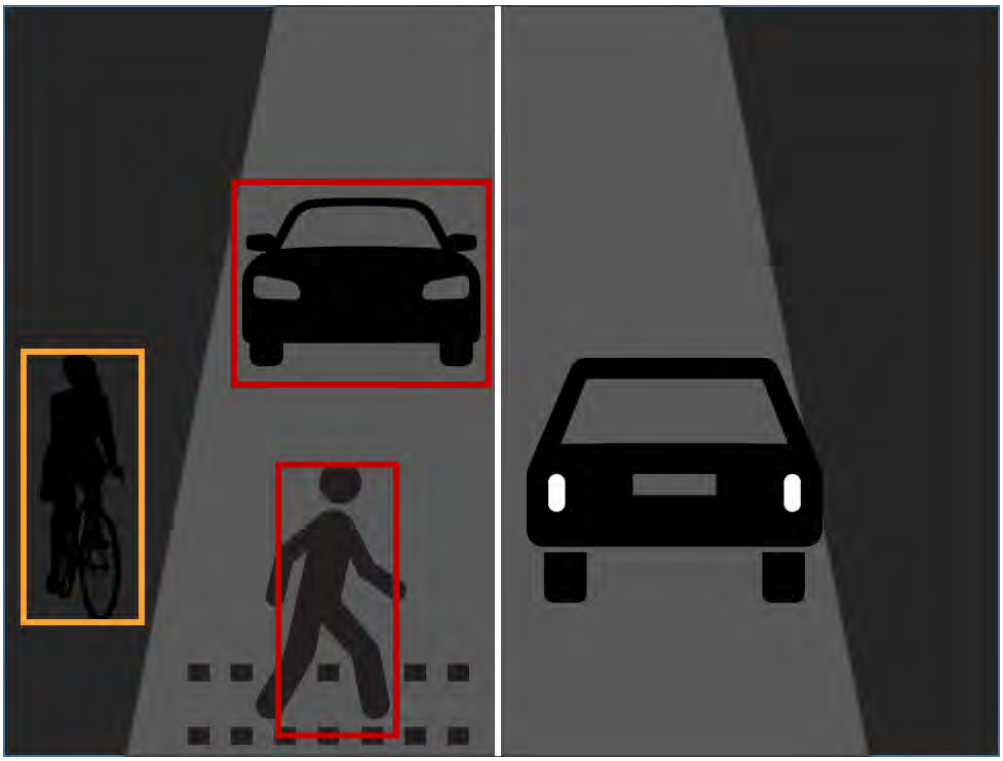


Figure 51:Considering the point of view from of a vehicle driving through a double direction street, the most dangerous situations are streets intersections in which the driver wants to make a turn, sudden breaking caused by traffic lights and pedestrians crossing and parking. For this last scenario the vehicles riding on the opposite direction should be also detected.

In this environment, the 5 objects established in the interurban case would not be enough. As in terms of vehicles, 5 would be enough, the system should be able to detect multiple pedestrians and bicycles. By thumb-rule, considering an average car is 2 meters width and leaving 2 meters more by side, and that an average person occupies a width of 0.5 meters, in an extreme case where several people stand shoulder by shoulder behind a car, that would add to 12 pedestrians to detect. Therefore, a CMS in urban environments should detect at least **12 objects** to assure a minimum safety functionality.

In terms of speed, vehicles are allowed to circulate maximum at 50 km/h, which is 14 m/s. Therefore, the 15 fps proposed for the interurban is more than enough for this scenario.

Summarizing, the proposed system should be able to detect at least 12 objects in the daylight scenario, and 5 in the low-light scenario as pedestrians are not detectable. A frame rate of 15 frames per second would be suitable for high speed situations.

Considering that a CMS by regulation should provide a 30 fps frame rate, the processing load can be lighten by applying the object detection algorithm in one of every two frames or less. As the action does not change as fast, the bounding boxes calculated in the previous frame can be kept in the actual one to create a sensation of continuity, as seen in figure

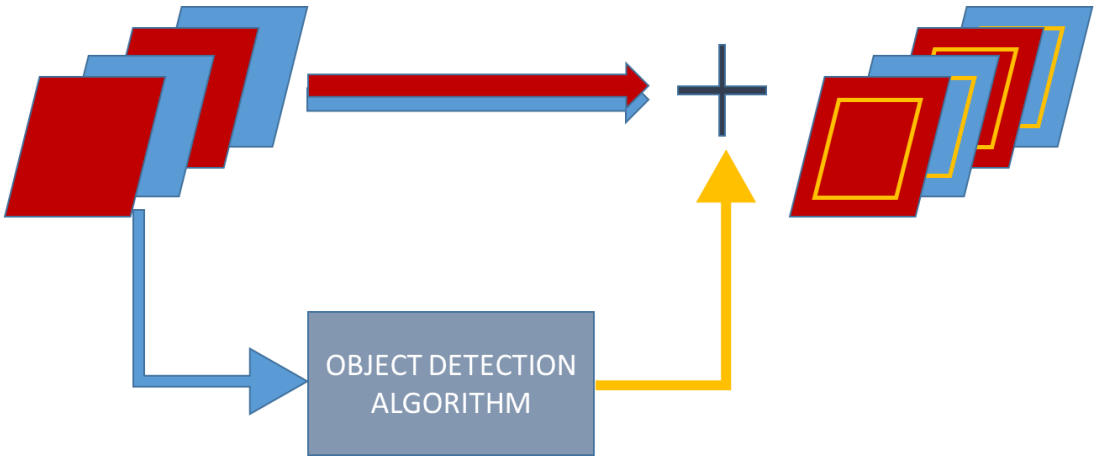


Figure 52: Down sampling of the object detection. In this example, one of every two frames is processed. In this case, the blue images are processed and the red ones bypassed. The bounding boxes calculated for every blue frame, are applied also to the following red one.

As mentioned before, the 15 fps is based on high velocity roads in the extreme case of Germany, as the rest of countries would not meet with such extreme relative velocities between vehicles as speed limits exist. In a more advanced system, a scene recognition could be applied, for example, every 1 second in order to determine in which environment the vehicle is driving along and adapt the frame rate to it.

In this project the detection is done frame-by-frame, but tracking algorithms can be added to predict the trajectory of the objects and therefore reduce the image processing. Instead of running the object detection that often, optical flow can be calculated to predict movement of the objects and determine the growth and position of the bounding boxes. However, as the background is not static applying optical flow prediction is more complicated.

Basing on the research done of the state-of-the-art technologies in object detection and tracking, the most promising algorithm to use, for the daylight scenario, would be the Convolutional Neural Networks. On a first instance, for a single FPGA, running a CNN for object detection would be already an achievement. In a FPGA cloud scenario, where the resources would be considerably increased, a second neural network to predict the movement and therefore add object tracking could be applied. With the combination of both networks, a complete object detection and tracking system would be provided.

Besides the training of the network itself, the format in which the input information is presented is important. By reducing the irrelevant information, the processing time is reduced.

1. In the case that the camera covers a wider area than the field of view required, a region of interest should be selected. In the Figure 54, building and trees are observable. These elements are not relevant for our detection object, as the potential dangers are at ground level. Therefore, a region interest on the horizontal level of the car is selected and cropped, as seen in Figure 53.

Figure 54: Sample input image

Figure 53: Region of interest for the sample input image

2. While the recorded image should have a high resolution for the direct display, for the processing it can be reduced into a smaller one. For example, if the input has a resolution of 1920×1080 pixels and the region of interested is reduced to 1920x540 pixels, it can be rescaled to 960x270 pixels.
3. It is common for CNN to convert the image to grayscale before running the object detection. As in this case the color of the objects is not a relevant information, the image is converted to grayscale as seen in Figure 55.

Figure 55: Grayscale ROI of the input image

4. As the best kind of algorithms for real-time detection are the single shot algorithms like YOLO, the entire image is inputted at once without sliding windows. The same applies for the low-light algorithm. The algorithm returns the bounding boxes position and sizes on the treated image, as seen in Figure 56.

Figure 56: The algorithm returns the position and sizes of the bounding boxes (bottom). For better understanding, graphically would be seen as in the top image when overlapping with the treated image.

5. As the image has been scaled prior to the algorithm processing, the bounding boxes position and sizes are relative to the downscaled region of interest of the image, and therefore an upscaling and an offset relative to the original image should be applied before adding to the bypassed image. The final result would be such as seen in Figure 57, with the bounding boxes overlaid to the original image.

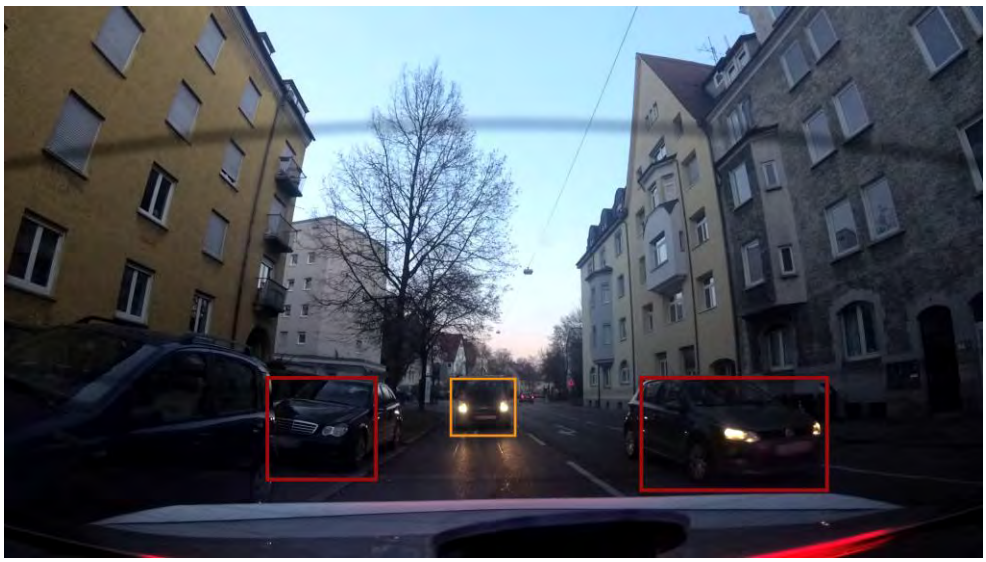


Figure 57: Sample of the final image to be displayed

In all the approaches to implement the CNN in the FPGA, the weights that define its behavior need to be calculated previously. Therefore, the first step should be designing and training the CNN on software. Python offers a lot of libraries and functions to work with Caffe and Tensorflow, two of the main open-source machine learning frameworks, as well as Keras, a neural network library written in Python. Moreover, several tutorials on how to design your own CNN with the desired classes to detect and using an own dataset, exist on the Internet.

The first step to design our CNN is to define our own dataset. There are several APIs like the ones from Google and Bing, that allow to easily search and download a big number of images. Those images should be classified on two separate folders, one for training and one for testing and validating.

Once designed the CNN, either from scratch or by “transfer learning”, which is preferred, the network should be trained with the training images. Afterwards, the test images are used to confirm that the network functions correctly. Once validated, the weights can be saved for later upload to the FPGA implementation.

As said, the most accessible language to run a CNN is Python, therefore the easiest way to have it running on the FPGA would be using this language on the board. To do that, PYNQ should be installed to the board. The target board ZCU106 is not officially supported, but the documentation of PYNQ have a section for using it on unsupported boards. Therefore, the first step should be trying to build an image for our target board. Once the system is running, the next step is adapting the previously designed and trained network to this platform. This process corresponds to the daylight algorithm.

For the night algorithm, the first step should be optimizing the proposed code by correcting the mentioned flaws and adding an approximation of the objects distance. Then, using the right Matlab version with the vhdl coder add-on, the code should be converted to vhdl adapting the original to the coder constraints. Another option could be hand translate the Matlab to Vhdl, if the time and skills necessary for it are available.

Besides running both algorithms on the same board, which might not be possible as the resources are limited even for just the neural network, the connection to the camera, preprocessing of the images, the formatting for displaying and the reproduction of those should be also implemented on the FPGA, what would mean more resources consumption.

Therefore, running both algorithms plus the basic CMS functionality in a single FPGA, with a real-time performance and high precision is quite an optimistic goal. However, for the final goal of connecting to a remote FPGA cluster, those requirements could be met.

Once the system is implemented, a complete testing of the different scenario situations mentioned before should be executed. The object detection is not a compulsory requirement for CMS, but if implemented, some minimum detection ratios should be meet to make the system worth deployed. For just an informative functionality, a precision of at least of 70%, being not ambitious, should be met. For an ADAS interference functionality, the precision should be around 95%, even higher if it involves breaks or direction control. As mentioned, it should have a real-time behavior meeting the regulation requirements of 30 frames per second on daylight scenarios and 15 frames per second on low-light situations.

Covering all the aspects mentioned in here, would provide a complete CMS system with object detection.

5. Budget

5.1. Components cost

Considering the setup provided by the HS Ulm, the cost of the components needed to deploy an experimental Camera Monitoring System is the one shown in the Table 21.

Table 21: Set-up components cost

CMS	**Cost**
Zynq UltraScale+ MPSoC ZCU106 (6)	1.791,86 €
Camera (28)	196,51 €
Display (29)	19,99 €
SD Card (8Gb) (30)	6,16 €
HDMI cable (31)	2,14 €
	2.016,66 €

The FPGA board includes the power supply, ethernet and USB cables.

5.2. Software cost

In order to program a Xilinx FPGA board, the proprietary software Vivado is needed. However, when buying a board, a license restricted to that board is provided for free. On the other hand, for the low-light scenario, a Matlab demonstration has been provided and therefore the program is needed. There is a discount license for students, which reduced the software cost to the following prices shown in Table 22.

Table 22: Software cost

Program	**License**	**Cost**
Xilinx Vivado (6)	Restricted to the board	Free
Matlab (32)	Student	35,00 €
		35,00 €

5.3. Manpower cost

In order to calculate the manpower cost, will be used as orientation the fact that this project computes as 12 ECTS and 1 ECTS equals to 30 hours. The price of a junior engineer will be calculated as 12 € per hour. Additionally, the tutors offered counselling during the project development. Considering that they offered a 1 hour of counselling out of 10 hours of project and that a senior engineer cost per hour is 18 €, the total manpower cost is the one shown in the Table 23.

Table 23: Manpower costs

	Total hours	Cost
Junior engineer (12 €/h)	360 h	4.320,00 €
Senior engineer (15 €/h)	36 h	540,00 €
		4.860,00 €

5.4. Total cost

The total cost of the project, including hardware, software and manpower for a prototype CMS based on a FPGA, would add up to the value of Table 24.

Table 24: Total cost of the project

CMS	2.016,66 €
Software	35,00 €
Manpower cost	4.860,00 €
Total	**6.911,66 €**

An optimized design and the creation of this system for serial vehicle production would reduce the prize and make it suitable as an ADAS for commercial vehicles. In the actuality, the change of lane assistant, collision detection or dynamic cruise control are extras sold around the 2000 € to the costumers. Therefore, a CMS could be also introduced as an extra with a realistic price, specially in the luxury car sector.

6. Further development

The next step following this thesis should be implementing the overall concept approach, coding both the neural network for daylight situation and the low-light point detection algorithm and implement them on the FPGA. Afterwards, the behavior of the system should be tested in terms of latency, power consumption and resource usage, to determine if the system meets the real-time requirements in a realistic power and resource budget.

An alternative way to approach the system would be to study how the camera and corresponding object detection algorithms could interact with other sensors in vehicles, as could be radars, lidars, proximity detection or line detector, in order to enrich the information obtained.

Once the object detection algorithm is running on real-time, the system should be locally installed in different vehicles for real scenario testing. The different tests would provide useful data to improve the system and show its flaws, as well as an empiric demonstration of the advantages towards regular mirrors.

In terms of the complete ongoing research, both the communication between the camera on the car and the cloud in one direction and back from the cluster to the screen device installed on the vehicle, should be designed and implemented. Once the video stream on real-time is properly uploaded to the cloud and back, the object detection should be added to the cloud FPGA cluster and the complete system tested in the different scenarios.

The on-cloud application would require also to study and design the required network devices that should be installed both in vehicles and roads to be able to have constant communication between the car system and the servers running the algorithm, as it is not a feature that the vehicle can leave aside. Network security, interferences, handovers and channel capacities would have to be sized as well for the telecommunication network, focusing on the upcoming 5G technology.

7. Conclusions:

In this thesis, a deep research on object detection algorithms to be applied in the camera monitoring systems has been performed. The different state of the art algorithms has been approached and several implementations compared to define the best one to cover the required problem.

Convolutional neural networks are the most suitable option for object detection, and the steps to design and implement this technology has been described. Some tools and implementations of CNN on FPGA have been presented. In general terms, CNN require a lot of processing resources. The results shown in the literature trade precision for frame rates close to real-time application, using board-constraint implementations.

In terms of FPGA resources, it is difficult to obtain both the speed required by the regulations and the precisions needed to add a real value in terms of information enhancement and definitely not enough for active ADAS.

An optimal algorithm code should be easily portable from one FPGA board to another, however this has been proven a flaw for this kind of platforms so far. As new improved FPGA hardware appear in the market, it is required to rewrite the code every time to adapt it to the new board, which is an undesired limitation.

Another counterpoint for using FPGA boards, is the fact that GPU's offer more economical and resourceful boards that provide better results that do allow real-time application without trading off precision. On the other hand, FPGA boards outstand GPU in terms of power consumption. An interoperable approach mixing both platforms could be an option to consider for future upgrading and should be taken into account while programming the inputs and outputs of the different blocks. For example, the output from the board that obtains the images in the vehicle should be reusable by any kind of board on the data processing end.

In general terms, using a FPGA board as a single ECU for a camera monitoring system with object detection on the nowadays boards and with the existing framework for CNN deployment, might not be the best option but is still a feasible one. To do so, it will require high programming skills and time investment for the optimal code development and implementation.

Bibliography:

1. Standardization, International Organization for. ISO 16505:2015 Road vehicles. 2015.

2. 6, UN Regulation No. 46 Revision. Uniform provisions concerning the approval of devices for indirect vision and of motor vehicles with regard to the installation of these devices,. 1 July 2016.

3. Terzis, Anestis. Handbook of Camera Monitor Systems - The Automotive Mirror-Replacement Technology based on ISO 16505. s.l. : Springer International Publishing, March 2016.

4. —. „Digital Mirrors – International Regulation and System Design based on hybrid Image Processing“. Brussels : In Proceedings of the 5th AutoSense Conference, Sept. 2018.

5. Xilinx zc702 board. [Online] https://www.xilinx.com/products/boards-and-kits/ek-z7-zc702-g.html#hardware.

6. Xilinx ZCU106. [Online] https://www.xilinx.com/content/xilinx/en/products/boards-and-kits/zcu106.html#hardware.

7. Jones, Paul Viola and Michael. Robust Real-time Object Detection. Vancouver, Canada : s.n., 2001.

8. Irgens, Bader, Lé, Saxena and Ababei. An efficicient and cost effective FPGA based implementation of the Viola-Jones face detection algorithm. Marquette : s.n., 2016.

9. Vahid, Chen Huang and Frank. Scalable object detection accelerators on FPGAs using custom design space exploration. 2011.

10. Cho, Mirzaei, Oberg and Kastner. PGA-Based Face Detection System Using Haar Classifiers. University of California : s.n., 2009.

11. Lee, Son, Choi and Min. HOG Feature Extractor Circuit for Real-time Human and Vehicle Detection. Korea Electronics Technology Institute : s.n., 2012.

12. Hahnle, Saxen, Hisung, Brunsmann and Doll. FPGA-based Real-Time Pedestrian Detection on High-Resolution Images. University of Applied Sciences Aschaffenburg,Germany : s.n., 2013.

13. R-CNN, fast R-CNN, faster R-CNN and YOLO object detection algorithms. [Online] https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e.

14. Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. 2016.

15. Chong, Jason. Machine Learning on FPGAs . UCLA : s.n., 2015.

16. DEEPHi. [Online] http://www.deephi.com/.

17. Forum xilinx. [Online] https://forums.xilinx.com/t5/Xcell-Daily-Blog-Archived/Tincy-YOLO-a-real-time-low-latency-low-power-object-detection/ba-p/815840.

18. [Online] https://github.com/Xilinx/ml-suite/tree/master/apps/yolo.

19. Nagasamy, Vijay. Accelerating ADAS Computer Vision Application Development at Ford using SDSoC. October 2, 2018.

20. Ruizhe Zhao, Xinyu Niu, Yajie Wu, Wayne Luk and Qiang Liu. Optimizing CNN-based Object Detection Algorithms on Embedded FPGA Platforms. Imperial College London : s.n., 2017.

21. H. Sharma, J. Park, D. Mahajan, E. Amaro, J. K. Kim, C. Shao, A. Mishra, H. Esmaeilzadeh. From High-Level Deep Neural Models to FPGAs. s.l. : 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2016.

22. Caffe deep learning. [Online] http://caffe.berkeleyvision.org/.

23. Hamdan, Muhammad K A. VHDL auto-generation tool for optimized hardware acceleration of convolutional neural networks on FPGA. Iowa State University : s.n., 2018.

24. Xilinx. PYNQ. [En línea] http://www.pynq.io/.

25. Yaman Umuroglu, Nicholas J. Fraser, Giulio Gambardella, Michaela Blott, Philip Leong, Magnus Jahre and Kees Vissers. FINN: A Framework for Fast, Scalable Binarized Neural Network Inference. Xilinx Research Labs, Norwegian University of Science and Technology, University of Sydney : s.n., 2016.

26. Yingfeng Cai, Xiaoqiang Sun, Hai Wang, Long Chen and Haobin Jia. Night-Time Vehicle Detection Algorithm Based on Visual Saliency and Deep Learning. s.l. : Jiangsu University, China, 2016.

27. Antonio Lopez, Jorg Hilgenstock, Andreas Busse, Ramon Baldrich, Felipe Lumbreras, and Joan Serrat. Nighttime Vehicle Detection for Intelligent Headlight Control. Barcelona : s.n.

28. Image sensor supplier. [Online] https://www.mouser.es/ProductDetail/ON-Semiconductor/NOIV1SN2000A-QDC?qs=BJFa2WV4K9lDnvJ9SN4huA==.

29. Screen monitor supplier. [Online] https://tienda-first.com/monitores-con-taras/21729-tft-19-fujitsu-b19-3p-grado-b-ver-fotos-tft-19-con-altavoces-5-4-resolucion-1280x1024-dot-pitch-0-294-mm-respuesta-5-ms-contrast.html?utm_campaign=GShopping&utm_medium=&gclid=CjwKCAjwtYXmBRAOEiwAYsyl3EYAE.

30. SD Card supplier. [Online] https://es.rs-online.com/web/p/products/6957334/?grossPrice=Y&cm_mmc=ES-PLA-DS3A-_-google-_-PLA_ES_ES_CatchAll-_-Ad+Group+Catch+All-_-PRODUCT_GROUP&matchtype=&pla-293946777986&gclid=CjwKCAjwtYXmBRAOEiwAYsyl3IIpWH13XRjZgmFXqwY0u1-4n1hchJsAifSJ22cUi60iWA30W.

31. Informatics components supplier. [Online] https://www.pccomponentes.com/equip-cable-hdmi-20-3d-macho-macho-alta-calidad-18m.

32. Matlab Pricing and Licensing. [Online] https://de.mathworks.com/pricing-licensing.html?prodcode=ML&intendeduse=student.

33. Advani, Tanabe, Irick, Sampson and Narayanan. A scalable architecture for multi-class visual object detection. University of Auckland : s.n., 2017.

34. Joseph Redmon, Ali Farhadi. YOLO9000: Better, Faster, Stronger. 2017.

35. Stylianos I. Venieris, Alexandros Kouris and Christos-Savvas Bouganis. Toolflows for Mapping Convolutional Neural Networks on FPGAs: A Survey and Future Directions. ACM Computing Surveys : s.n., 2018.

36. Boser, Bernhard E., Guyon, Isabelle M. y Vapnik, Vladimir N. A training algorithm for optimal margin classifiers. COLT : s.n., 1992.

37. Object recognition for dummies. [Online] https://lilianweng.github.io/lil-log/2017/10/29/object-recognition-for-dummies-part-1.html.

Glossary

Word: English. *Castellano.* Català
CMS: Camera Monitoring System. *Sistema de monitoreado por cámara*. Sistema de monitorització amb càmera.
ECU: Electronic Control Unit. *Centralita electrónica*. Centraleta electrónica.
CMOS: Complementary metal–oxide–semiconductor. *Semiconductor complementario de óxido* metálico. Metall Òxid Semiconductor Complementari.
FOV: Field of Vision. *Campo de visión*. Camp de visió.
ROI: Region of Interest. *Región de interés.* Regió d'interès.
ADAS: Advanced driver-assistance systems. *Sistemas avanzados de asistencia a la conducción*. Sistemes avançats d'assistència a la conducció.
FPGA: Field-Programmable Gate Array. *Matriz de puertas programables*. Matriu de portes programables.
GPU: Graphics Processing Unit. *Unidad de procesamiento gráfico.* Unitat de Procés Gràfic
CPU: Central Processing Unit. *Unidad central de procesamiento.* Unitat central de processament.
YOLO: You Only Look Once. *Una sola observación.* Una sola observación.
CNN: Convolutional Neural Network. *Redes neuronales convolucionales.* Xarxes neuronals convoluncionals.
ISO: International Organization for Standardization. *Organización Internacional de Normalización.* Organització Internacional per a l'Estandardització.
IP: Intellectual Property. *Propiedad intellectual.* Propietat intellectual.
UN: United Nations. *Naciones Unidas.* Nacions Unides.
MTF: Modulation Transfer Function. *Función de Transferencia de Modulación.* Funció de Transferència de Modulació
HS: Hochschule. *Escuela superior.* Escola superior.
IoT: Internet of Things. *Internet de las cosas.* Internet de les coses.
HDMI: High-Definition Multimedia Interface. *Interfície multimedia de alta definición.* Interficie multimedia d'alta definición.
RGB: Red Green Blue scale. *Escala rojo-verde-azul.* Escala vermell-verd-blau
YCbCr: Luma blue Chrominance red Chrominance space. *Espacio de luminância, cromatura roja y cromatura azul.* Espai de luminancia, crominancia vermella i crominancia blava.
DSP: Digital Signal Processor. *Procesador de señales digitales.* Processador de senyals digitals.
SD: Secure Digital. *Seguridad digital.* Seguretat digital.
HoG: Histogram of Gradients. *Histograma de gradientes.* Histograma de gradients.
SVM: Support Vector Machine. *Máquinas de soporte vectorial.* Màquina de vectors de suport.
CUDA: Compute Unified Device Architecture
USB: Universal Serial Bus. *Bus en serie universal.* Bus en sèrie universal.
UART: Universal Asynchronous Receiver-Transmitter. *Transmisor-Receptor asíncrono universal.* Transmissor-Receptor asíncron universal.
BLOB: Binary Large Objects. *Objetos binarios grandes.* Objectes binaris grans.
VHDL: VHSIC Hardware Description Language. *Lenguaje de descripción hardware VHSIC.* Llenguatge de descripción hardware VHSIC
VHSIC: Very High Speed Integrated Circuit. *Circuito integrado de muy alta velocidad.* Circuit integrat de molt alta velocitat.