FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Automatic Traffic Congestion Detection Using Uncontrolled Video Sources.

Pedro Fernando Quintas Loureiro

Report of Project

Master in Informatics and Computing Engineering

Supervisor: Rosaldo Rossetti (Assistant Professor)

Co-supervisor: Rodrigo António Marques Braga (Master)

29^{th} June, 2009

Automatic Traffic Congestion Detection Using Uncontrolled Video Sources.

Pedro Fernando Quintas Loureiro

Report of Project

Master in Informatics and Computing Engineering

Approved in oral examination by the committee:

Chair: Jaime S. Cardoso (Assistant Professor)

External Examiner: Ana Paula Cláudio (Assistant Professor)

Internal Examiner: Rosaldo Rossetti (Assistant Professor)

13^{th} July, 2009

Abstract

Although traffic demand in big cities has far exceeded the capacity of the transportation network a long time ago, traffic volume has been increasing for a long time. The increasing demand has been reflected in an increase of the traffic congestion-related problems. The most noticeable are the money wasted in the unnecessary consumption of fuel and the time lost behind the wheels.

One of the latest novelties in the GPS market is route guidance with real-time traffic information. This type of services allows the calculation of the best route considering the estimated time to travel each road segment based on their current congestion level, thus avoiding congested roads.

For a system like this to work, a great amount of information must be inserted into a database which maintains information about the congestion levels of several roads under surveillance by the system.

NDrive, the company that initially proposed this project, currently has an information service that gathers information from several sources that include several news websites, radio broadcasts, traffic authorities' websites and video streams. The information present in those sources is gathered by a human operator and inserted into the system. The information is then used by NDrive as well as third parties.

Given that real-time route guidance will require a much larger amount of information to be inserted into the database, NDrive wishes to study which possibilities it has to automatise the process of evaluating road congestion levels.

Thus, this project will analyse video processing techniques that could be applied to process and evaluate video streams that are currently published on the Internet as a means to gather such information.

As a result of this project, a modular application was developed with several implemented modules which can be re-used to meet other purposes and can be easily extended.

The implemented modules yield promising results and can be used to estimate traffic features that are reflected by the traffic congestion level.

This dissertation contains a technological study, an explanation of the proposed solution and its details, a deep analysis of the results and the project's conclusions.

Resumo

A exigência de que as grandes cidades são vítimas a nível do volume de tráfego rodoviário excedeu já há muito tempo as capacidades das suas redes de trânsito. Esta exigência crescente que não tem vindo a ser satisfeita traduziu-se num aumento dos problemas relacionados com o congestionamento de tráfego. Dos quais, os mais notórios são o dinheiro gasto no consumo desnecessário de combustíveis e o tempo que os automobilistas perdem "atrás dos volantes".

Uma das novidades mais recentes no mercado dos aparelhos de GPS é o calculo de rotas com recurso a informações de trânsito em tempo real. Estes serviços permitem calcular a melhor rota tendo em consideração o tempo estimado para percorrer cada rua com base no seu nível de congestionamento, evitando assim as ruas mais congestionadas.

Para um sistema destes funcionar propriamente, grandes volumes de informação têm de ser inseridos numa base de dados que mantém informação dos níveis de congestionamento das diversas vias monitorizadas pelo sistema.

A NDrive, a empresa que propôs este projecto inicialmente, possui presentemente um serviço de informação que recolhe de diversas fontes incluindo web-sites de notícias, emissões de rádio, web-sites das instituições responsáveis pela manutenção de estradas e *streams* de vídeo. A informação recolhida nas diversas fontes é inserida manualmente no sistema por uma pessoa. Esta informação é então utilizada pela NDrive e utilizada por terceiros.

Dado que a utilização de um sistema de calculo de rotas com informações em tempo real exigirá que um grande volume de informação seja inserido na base de dados, a NDrive pretende analisar as hipóteses existentes para automatizar o processo de avaliação do nível de congestionamento das vias.

Deste modo, o projecto irá analisar técnicas de processamento de imagem que podem ser utilizadas para analisar e avaliar *streams* de vídeo que são actualmente publicadas na Internet.

Como resultado deste projecto foi desenvolvida uma aplicação modular que implementou diversos módulos que podem ser reutilizados para servir propósitos diferentes dos inicialmente previstos. Esta aplicação pode também ser facilmente estendida. Os módulos implementados apresentam resultados bastante prometedores e podem ser utilizados para estimar características do trânsito que reflectem o estado do congestionamento da via.

Esta dissertação inclui um estudo da tecnologia existente, uma explicação da solução proposta e respectivos detalhes, assim como uma análise profunda dos resultados e as conclusões do projecto.

Acknowledgements

If there is something or someone that I would like to thank for the impressive support that gave me throughout the several stages of this dissertation, that is the on-line community self designated by VMT. This is a group formed by other MIEIC students also doing their final project or dissertation and was created to serve as a self support group. It amazed me how I could be talking to some people that I never met in life and that are doing their projects abroad but at the same time feel like if they were right next to me. We shared the best moments one can have on the Internet, we also shared our deepest concerns about the thesis and mainly we helped each other to survive the best possible way through this difficult task that is doing our Master's dissertation.

A very special thank you for all of you guys! :)

Of course I cannot forget the great help from my supervisor Rosaldo Rossetti and his good friend Rodrigo Braga who helped me giving my first steps into image processing and made excellent reviews of my (bad) English.

I would like to give one last word of appreciation to Professor Velhote Correia whose impressive knowledge in the area of image processing gave me great ideas for the possibly most interesting and important developments of my project, and all this in just one quick meeting that took no longer than 90 minutes.

To all of the above and to everyone else that I could not mention,

thank you!

Pedro Fernando Quintas Loureiro
PS: my thesis was also supervised by JC and HK!

"A tese é tipo uma linha semi-recta.. nao tem fim.."

André Oliveira, on VMT forum

Contents

List of Figures

List of Tables

Acronyms

AI	Artificial Intelligence
EP	Estradas de Portugal
FEUP	Faculdade de Engenharia da Universidade do Porto(Faculty of Engineering, University of Porto)
GIS	Geographic Information System
GPS	Global Positioning System
GUI	Graphic User Interface
HSV	Hue, Saturation, Value (model used to represent colour)
IEEE	Institute of Electrical and Electronics Engineers
PDA	Personal Digital Assistant
RGB	Red Green Blue (model used to represent colour)
UI	User Interface
UML	Unified Modeling Language
XML	Extensible Markup Language

Chapter 1

Introduction

This chapter contains a brief introduction to the points of the project. It starts herein by explaining the context, then gives an insight into the project motivations, objectives and scope and finishes explaining the structure of this document, providing an overview of the remaining chapters. It is important to notice that all of the subjects explained here are explained again with more detail in the respective chapters.

1.1 Scope

Traffic volume has been growing ever since. Only in the last years, with the rise of oil prices this growth seems to have been slowing down [Fed09, Jam07]. However, the demand in big cities has far exceeded the capacity of the transportation network a long time ago. The latest published studies in 437 urban areas in the U.S. have estimated that 4.19 billions of hours were wasted due transportation delays and 2.87 billions of gallons (10.87 billions of litters) of fuel were unnecessarily consumed, summing a total cost of 78.2 billion dollars during the year of 2005[SL07].

Cities have taken measures to reduce traffic congestion problems. Some examples include junction improvement or reduction to avoid that movement in one direction forces vehicles moving in other directions to have to stop; blocking access of particular vehicles to certain areas at the most busy moments of the day; allowing only particular vehicles to be driven on some days of the week; setting congestion tolls at peak hours; creating bus lanes or special lanes for vehicles with more than 2 or 3 people inside; better urban planning, among others.

Before taking any measure, it is fundamental to have a deep understanding of the transportation network system for that particular area. This requires to monitor the network throughout a period of time in which the amount of demand and current capacity is

evaluated on several points of the network.

Currently, information is collected by the responsible organisations with the use of loop detectors, radars or dedicated video cameras and sometimes is released to the community through their website. News services will gather information from several sources including phone reports from drivers and broadcast the collected information through radio and/or their websites.

In nearly all of developed countries worldwide there are websites that allow the public to see videos in real time or photos taken within a very small time frame, e.g. up to 5 minutes. The main use of this information is to inform drivers which the best way to get to their destinations on that day is.

NDrive

NDrive is a Portuguese company founded in 2001 that is in the business of GIS and mobile market. It has been on a continuous and profitable growth with the expansion of its portfolio and its market presence all over the world selling their products in more than 40 different countries.

NDrive provides an advanced navigation experience through the use of GPS devices, with clear and precise turn-by-turn spoken and visual instructions, including street and place names for door-to-door navigation with detailed local information, tailored for business or entertainment use [NDrc].

The products offered by NDrive range from GPS devices to navigation software that can be installed on PDAs and mobile phones. Extra maps can be bought and accessories are available at retailers.

Currently under development, the next addition to NDrive's service portfolio is route calculation based on real time information. The system in use requires a human operator that has to manually gather information from several sources that include several news websites, radio broadcasts, traffic authorities' websites and video streams and insert the relevant information into an internal application in the form of a performance scale ranging from 1 to 5, representing the freeflow and congested regimes, respectively.

1.2 Motivation and objectives

Under these circumstances, NDrive has proposed FEUP to analyse the possibilities of developing a traffic monitoring system, capable of analysing public video streams and automatically evaluating traffic conditions with the intent of helping the human controller to focus on the most important tasks.

Such a system has capabilities to judge the traffic conditions from captured videos, providing valuable information to the algorithms that calculate the best routes with real

time information. Other potential uses of the information generated by this system are improving the understanding of the needs of traffic networks in the areas where it is employed; providing data to simulation and statistical studies and supporting decisions made that are affected and affect the traffic network like the ones made during urban planning and design.

One objective for this system is to use the infrastructure of cameras already available throughout Portugal. This will profit from an underused infrastructure which the only process that it receives is made by manual human inspection and will benefit a large number of people directly through the broadcast of more reliable traffic information and better route guidance results, but also indirectly by reducing the number of drivers getting into already over-saturated routes which benefits the whole community in urban areas.

One implication of the use of an existing infrastructure is that it is not expectable to have control on the cameras. It is impossible to place the cameras on the best locations, pointing to the best directions for the algorithms to produce the best results. This will require robustness from the algorithms employed by the system to be developed. On the other hand, this will enable it to be used anywhere in the world making it easy to integrate it into all of NDrive's operation locations.

Another advantage of the use of an existing infrastructure is a great reduction in the costs involved with the system: all the cameras are available on-line therefore it is only required Internet connection to access that information. The operating cost is not higher than the current cost as the proposed system will not require skilled professionals. This way, the only costs inherent to the use of this system are Internet access, one computer to run the algorithms and the energy consumption associated to the computer use. Another positive factor that supports this option when compared to other alternatives is the low impact on routes as it does not require direct intervention to install special devices that could measure vehicle speeds or count them.

1.3 Project Description

This project will carry out a study on image processing techniques and options to assess the road congestion levels, creating one prototypical application to address the needs of NDrive.

Analysing all the objectives, it is expected that the proposed system fulfils the following requirements:

- make no assumptions about the data sources, camera positions, environmental conditions and infrastructure proving the information;
- it must be able to evaluate traffic conditions in real time;

- it should be able to work automatically, without requiring human control or supervision;
- it should be able to learn and benefit from human examples;
- the developed application should be as generic as possible and easily extensible to allow new techniques to be installed and used.

1.4 Organisation of this Dissertation

Besides the present introduction, this dissertation has six additional chapters, as follows.

Chapter 2 presents the problem statement, in which it analyses the current status of this project's context. The objectives and motivation of the project are explained and it presents the methodology used during the project development.

Chapter 3 contains a technological study where projects with a similar scope and image processing techniques focused on traffic monitoring are analysed. It also contains an explanation of data-mining techniques and neural networks. This chapter finishes by providing an insight into the existing traffic information services and alternatives to traffic estimation.

Chapter 4 proposes the solution. It starts by explaining which information is to be extracted and then suggests methods to perform the evaluation of the extracted information and to learn from examples. It finishes by proposing methods to increase the extensibility of the application to be developed.

Chapter 5 presents the prototypical development carried out during this project. It analyses the program architecture as well as every modules and algorithms implemented.

Chapter 6 contains the analysis of results. All of the implemented modules and algorithms are here carefully analysed. Several examples of real videos processed and their results are presented in this chapter. It also contains a profiling analysis carried out on every implemented module and has analysed the effect of using the different synchronisation options.

Finally, Chapter 7 presents the conclusions of this project. It discusses the project's success, general conclusions, innovations and limitations. Additionally, it provides an overview of what the future work could be.

Chapter 2

Problem Statement

In Chapter 1, a short introduction to the problem was made. In this chapter, the problem is once again explained, this time with more detail than before. Some details that were disregarded in the introduction will be revealed in this chapter. Also, implications arising from objectives, restrictions and limitations will be further analysed.

The description is divided into three steps: first, the context of the activities related to the project and its current status is explained. Then, the project objectives are analysed. The third step is justifying the purpose of the project by stating its motivation. The chapter finishes with a summary of all the subjects herein dealt with.

2.1 *Status quo*

NDrive, the proponent of this project, has a large community of clients that use their devices and software in several countries of the world. When a client requests one route between two points, the device will access the information that it has about both points and the possible paths between them. Additional restrictions may be added like avoiding tolls. The algorithm will then return the best path that fulfils the requirements of the user under one of the following categories: shortest distance or fastest to travel [NDrd].

Although the calculation of a route distance is an easy task, estimating how long it will take is not. Currently, each road segment present in the maps has an expected average speed which represents the average speed at which that segment is usually travelled. If the user requests the fastest route between two points, each road segment in a possible path will contribute with the time that it takes to be travelled at the expected travelling speed. The best path will be the one with shortest total time to be travelled.

This algorithm uses information that is statically present in the map data. As a consequence, that information is present on the map itself and is only updated when a new

version of the map is downloaded into the device. This makes it impossible to use updated information to estimate the travel times of routes.

Another feature of NDrive's software is the dynamic data support. Dependent on local data providers, this feature enables the user to have access to weather reports and forecasts, open pharmacies and cultural events [NDra]. The next addition to this features list under development is the use of traffic services to avoid or optimise traffic problems [NDrb].

On a slightly different business, NDrive also collects traffic information from Portugal. The information is gathered from different types of sources: radio news services, websites and traffic cameras freely available on the Internet. There is an application that streamlines the process of gathering information and inserting it into the system. For example, when one particular camera is being observed, if the operator labels it with a congestion level, the system knows which is the road under observation and will automatically assign that congestion level to the corresponding road. The congestion level of each road is a value between 1 and 5.

The traffic cameras used by NDrive are available on the Internet [Câ, Est, Lus, Bri]. These are public cameras therefore can be accessed by anyone interested in the information that they stream up. It is not known any application that automatically uses the information from these cameras to improve the quality of traffic information available to the public.

2.2 Objectives

With this project NDrive expects to have a deeper understanding of the possibilities to automatise its process of evaluating the traffic congestion levels of cities throughout Portugal.

The main source of information with which the project should work is video information from cameras located on the streets and highways. Streams currently available on the Internet are good examples of sources under that category.

The application should be extensible and developed with the general purpose in mind. Generality means it should make as few assumptions as possible about the data sources and the utilisation context of the application. Extensibility means that the application should be designed in a way that it makes further developments easy.

Having the general purpose in mind can be interpreted from two perspectives both applicable to this project. The first perspective is that the context of use may change. For example, the videos may be retrieved from different sources or the extracted information may be used in a different context. The second perspective is that there can be made no expectations about the video used to retrieve information from. This includes

both the video characteristics such as resolution, colour depth or frame rate and camera characteristics such as the position, height, pan, and so forth.

The extensibility of the application is related to its general purpose character, specially with the first perspective explained in the last paragraph. As an example, if it is decided to extract a different type of information from videos to use with a new purpose, it should not be needed to implement all features of the application that are already present in other parts of it.

One final objective for the application is the ability to learn from examples, aiming at improving the reliability of the results produced when analysing videos.

2.3 Motivation

The success of this project will enable NDrive to handle a much higher volume of information that will be required in order to use real-time congestion information when calculating routes.

The use of visual information alternatively to other means has the advantage of requiring no intervention to install special devices that could count vehicles or measure their speeds. Another advantage is that it is easier to understand what a camera is *'seeing'* and therefore debugging and optimisation are easier to make as the operators (any human being) have a very deep and natural knowledge in interpreting visual information.

This application will make use of an already existing infrastructure. This has two advantages: it will not require to install an expensive system of cameras and communications throughout Portugal while making more use of an underused, already installed infrastructure.

The general purpose will also bring new possible uses for the information extracted from videos. This will offer a low-cost traffic monitoring system that could potentially be used by other interested institutions. Low-cost means that it could be running throughout all year, 24 hours a day providing the most complete traffic information that could be retrieved.

Information retrieved from videos can be stored and sent in a much more compact format than images. For example, NDrive could install a camera in a non monitored road and instead of sending the raw video to be evaluated at the central server, it could run the algorithm when retrieving the image and only communicate the results of it, saving a great amount of bandwidth. For use in statistical or historical analysis, one alternative to store videos using a huge amount of disk space is storing the extracted values from the videos. This allows easy and fast calculation to studies and avoids that before each video is used in those studies, it be analysed by one person that would evaluate the conditions of traffic over time.

2.4 Methodology

This project started with an initial research period. During research, books were read, a list of papers from IEEE community were analysed, research at the library was carried out and the Internet was used to search for articles on image processing and traffic monitoring. The results of this research are presented in the chapter named Technological Study. This period took around three weeks and included the analysis of libraries for image processing and the development of some experimental code.

Then a development period followed where a more concrete applications started to be designed and developed. Several remakes and frequent refactories were necessary in order to produce a better quality software.

During the last two weeks of April and the first one of May, a paper was written and submitted to IEEE's Intelligent Transportation Systems Conference. This paper[LRB09] was focused on the initial research carried out in the beginning of the project and in the first experiments realised during the development of the application was submitted.

After the paper's submission, the development continued until the end of the first week of June. Since then, this dissertation has been written. Some of the contents of the paper were adapted to be part of this dissertation.

2.5 Summary

In this chapter, a more detailed analysis of the problem was presented. The new addition to NDrive's portfolio, route calculation with real-time traffic information will require a greater capacity of processing traffic information. This project has the objective of developing a general, multi-purpose application that extracts the traffic congestion level from a video stream and has the capacity of learning from examples. The reasons to do this, besides improving NDrive's capacity to offer a service to their clients, are the low-cost and intervention-free solution and the creation of information in a much more compact way which allows better quantitative and qualitative analysis.

Chapter 3

Technological Study

This chapter reports on the state of art in several areas. Its objective is to provide a deep knowledge that enables a supported reasoning about the subjects, algorithms and technical decisions discussed in this dissertation and also gives an overview of the context where this project is to be employed, allowing a better understanding of the needs and restrictions of the problem.

The first four sections are the most technical ones. The first will analyse projects with similar objectives and the following ones discuss the latest developments on image processing techniques, data mining and neural networks. The last sections will give an overview of related areas: traffic information publicly available and other methodologies to measure traffic without using images. All the information herein presented is summarised in the end of the chapter.

3.1 Similar Projects

In this section a description of several works by other authors is presented. It identifies works whose scope has similarities with the scope of this project and provides an overview of them.

Robust techniques for background subtraction in urban traffic video [SK04] provides information on techniques using background subtraction applied in detection of moving vehicles and pedestrians, ranging from the most simplistic to the more complex algorithms currently available.

An improved adaptive background mixture model for real-time tracking Aiming at optimisation, this project[KB01] presents a methodology that improves the algorithms based on adaptive background mixture models by using more intelligent update

equations which allows it to gain speed and more accuracy, and also to adapt more effectively to changing environments. Another feature is the ability to detect shadows.

Multiplicative Background-Foreground Estimation presents one alternative which proposes a new method to decompose the background and foreground in video frames[Por05]. It is capable of detecting moving and static lines through the use of different filters and is at the same time robust enough to cope with sudden illumination changes and computationally feasible to be implemented and used in real time systems. The results shown in the paper prove its effectiveness decomposing background and foreground from video frames.

Foreground object detection from videos containing complex background presents a new method for detection and segmentation of foreground objects from video with both stationary and moving background objects and subject to gradual or sudden changes. According to the authors[LHGT03], *"the convergence of the learning process is proved and a formula to select a proper learning rate is also derived. Experiments have shown promising results in extracting foreground objects from many complex backgrounds including wavering tree branches, flickering screens and water surfaces, moving escalators, opening and closing doors, switching lights and shadows of moving objects"*.

An Enhanced Background Estimation Algorithm for Vehicle Detection addresses a common problem in urban areas which proposes an enhanced version of the sigma delta background estimation method[VTBM08] which is optimised for urban traffic scenes that are frequently affected by slow moving or temporarily stopped vehicles.

Moving Object Refining in Traffic Monitoring Applications addresses the problem of moving object refining by processing the background subtraction results[WYQ+07]. It is able to remove sudden illumination changes, local reflected regions and moving cast shadows. The results demonstrate that this solution is efficient for a wide range of different conditions that may occur.

Vehicle Detection in Congested Traffic Situations offers a Hidden Markov Model based algorithm [YZMW07] to detect vehicles capable of dealing with vehicle occlusion in order to solve the problem of detection of vehicles under congested conditions. It first extracts features from the images and then classifies them into one of three categories: road, head and body of a vehicle. Vehicles are detected by analysing the sequence of categories found.

Context-Adaptive Approach for Vehicle Detection introduces a vision-based vehicle detection method that considers the lighting context of the images to apply the best classifier algorithm for that situation. In its implementation[AZXB07], four categories of light were found: daylight, low night, night and saturation. The results show considerable improvement in the detection performance with the use of context adaptive scheme.

Accurate Speed Measurement from Vehicle Trajectories discusses methods[AS07] for detecting and tracking vehicles that are able to measure their speed at a distance of 70 or 100 meters (for incoming or rear viewed vehicles). The detection uses a block based algorithm and tracking is made with a variant of extended Lucas Kanade template matching algorithm. This algorithm has detected vehicles with a speed accuracy of 2.3% for 95% of the vehicles, both in day and night time sequences.

Automatic Daytime Road Traffic Control and Monitoring System has a more particular but not less interesting application. It presents a system[ASB08] for automatic daytime road traffic control and monitoring system that retrieves traffic information like average speed, dimension and vehicles counting through computer vision approaches. It uses frame differencing algorithm and texture information to extract moving objects from the scene and then removes shadows from the foreground objects with morphological operators. Objects are afterwards tracked using a Kalman filtering process and parameters like position, dimensions, distance and speed are measured. Results from real outdoor videos show accuracy under daytime interurban traffic conditions.

Multi camera Based Traffic Flow Characterisation & Classification describes a different methodology for traffic flow characterisation. It applies a multi camera system[KGT07], with a omnidirectional camera and a pan-tilt-zoom camera. The latter is used to refine the information retrieved by the omni directional camera.

Multiple Vehicles Detection based on Scale-Invariant Feature Transform suggests a different approach to vehicle segmentation: quad-tree segmentation[CSY07]. To accomplish the task of tracking the vehicles, a scale invariant feature transform is used. It is able to extract parameters like vehicle count, speed and class.This approach has proved to be effective and robust specially when the phenomenon of vehicle occlusion occurs, in the cases of a vehicle changing its lane and when the affine shape of vehicle changes due car movement.

Tracking Ground Vehicles in Heavy-traffic Video targeting at heavily congested traffic situations, this paper[YMW+07] proposes an algorithm that detects and tracks vehicle corners and then groups them into vehicles. The results show that this algorithm is effective under the intended circumstances.

Analysis of movement in sequence of images proposes a methodology for the analysis of motion[Cor95]. It analyses techniques for detection, measurement and characterisation of movement and also focuses on edge detection. Although it is generalised for any type of images, the system was tested on image sequences containing traffic information.

Detecção de movimento e sua aplicação à monitorização de tráfego rodoviário (in English: movement detection and its application to traffic monitoring[Ale97]) has a similar objective to the objective of this thesis. It implemented a 2D motion detection method – Statistical Analysis of Difference of Images – to detect motion in real time using low cost equipment.

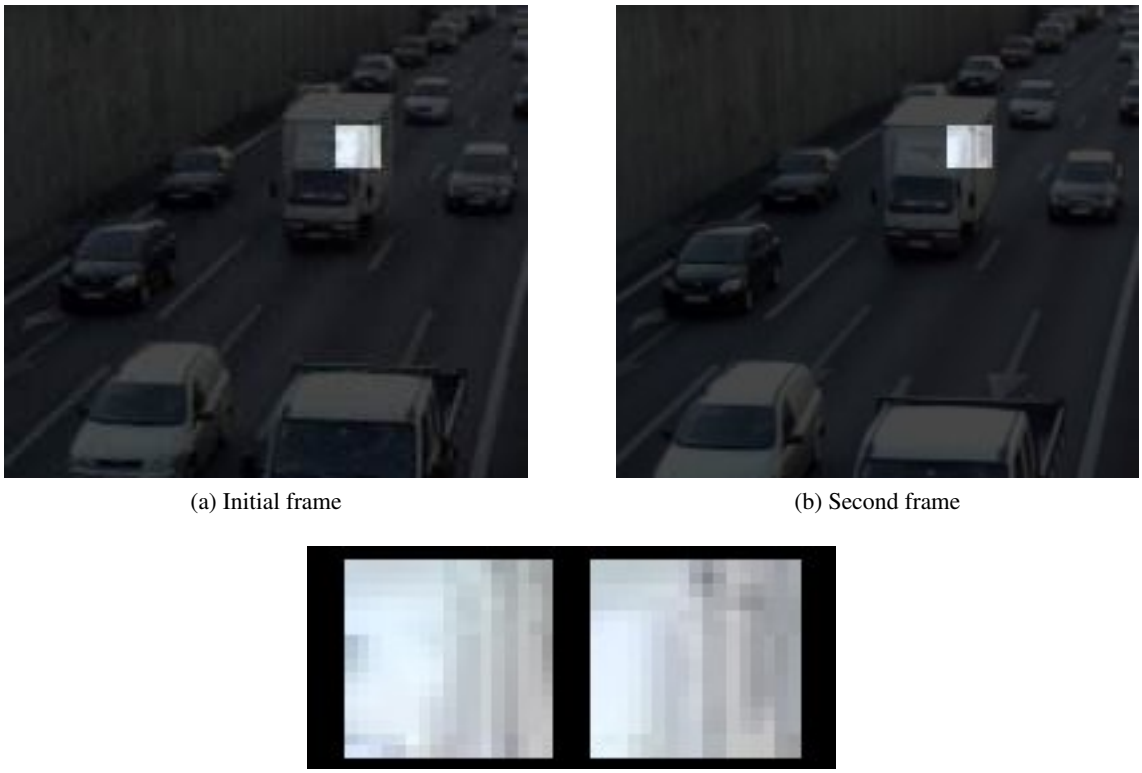
3.2 Image Processing

On this section several techniques and algorithms used in image processing are explained. The selected techniques are the ones which are mostly used in traffic monitoring applications.

Motion detection

There are several techniques to segment vehicles from the background of a scene picture or a video frame. The most widespread techniques involve the calculation of the optical flow. Optical flow tries to find vector fields that describe the way that the image is changing. In other words, it tries to detect where certain parts of the image have moved in the next frame. There are several ways of achieving that. [Smi97] and [BB95] describe some of the possible implementations of these algorithms. Optical flow techniques usually have two steps: first, detect feature points and then track them over the frames.

Feature points are special points that are distinguishable from its neighbours. Each algorithm may use different strategies to detect and track these points. One alternative is calculating the difference between two consecutive frames. Pixels for which difference is bigger than a threshold are considered to be moving. However, this technique does not detect the direction of movement and can only be used for simpler applications. Additionally, this algorithm suffers from the aperture problem. This problem happens when a surface with a smooth colour moves. As the colour is similar in all parts of the surface, movement may not be detected in areas of the captured image where the surface was already present because the difference is not sufficient to consider it a motion and thus, it is interpreted as image noise or small variations in illumination. This problem is demonstrated in fig. 3.1. It has two consecutive frames with the same region highlighted. In fig. 3.1c the same region in the two frames is compared and augmented. Although



(a) Initial frame

(b) Second frame

(c) Highlighted square comparison

Figure 3.1: Aperture problem example

they display a different part of a moving vehicle, focusing on smaller portions only is not enough to perceive any difference.

Background Subtraction

A different approach is used by background subtraction methods but they are also able to detect moving objects. This has one premise: the background will have small changes over time and foreground objects like cars are moving objects which will contrast with the background. Therefore, if foreground objects are detected it is safe to consider them as moving objects. Ideally, for traffic monitoring use, this background image would be the same frame without moving objects, showing the scene as if there were no cars or other moving objects on it. There are some alternatives to accomplish this.

The easiest method is choosing a fixed image that is the background image. After subtracting one image to another, those pixels whose difference is bigger than a threshold are considered to be moving pixels. This is usually too simplistic as changing illumination conditions, weather, noise or small movements of the camera may cause big differences when there are no moving objects in those regions.

(a) Input image (b) Background Estimation (c) Foreground mask (in red) applied to input

Figure 3.2: Background estimation example

Better options involve making a background estimation image, based on the frames captured by the camera. There are several methods to estimate the background and foreground of a sequence of frames. [Pic04] presents an interesting overview of these methods. The main difficulties of these algorithms happen in situations with very slow traffic movement or temporarily stopped vehicles as they start blending with the conceptual background. [VTBM08] offers a solution to this and [Por05] proposes an alternative to use time-varying background and foreground intrinsic images. Both of the last two studies achieved very good results in terms of successfully segmenting background and foreground. Fig. 3.2 shows the results of a background estimation algorithm after running for around 30 seconds. It has a background estimation which is very close to the real background and was able to successfully identify moving objects without any extra processing or calculation.

Vehicle Segmentation

Vehicle segmentation is probably the most frequent type of information that is extracted. Its objective is to distinguish between vehicles from the background and also the vehicles from each other. One common method to detect vehicles in a video sequence uses one of the previously analysed motion detection or background estimation to detect the cars. [CSY07] has a solution that compares the current frame with the background estimation using a quad-tree decomposition to find the pixels where cars are found. As stated in its experimental results, occluding vehicles may be grouped together as one object under heavy congested traffic conditions. [YZMW07] uses a Hidden Markov Model to detect cars under congested conditions, where occlusion is frequent. [YMW+07] has a different approach that does not require the background estimation, instead it detects information about the corners of the vehicles. Both [AS07, CHFH07] have a good survey of other vehicle detection algorithms and the latter proposes a way to combine several of these methods.

Shadow Subtraction

Other common task is shadow removal. This task has several benefits: the shadow of a moving object is also moving and hence could be considered a moving object; if the shadow of a car is considered together with the car, it may seem that the car is bigger than in reality it is; other example is when the shadow of a car connects it to another car, bringing in the possibility for a not optimised algorithm to merge both cars. [ASB08] suggests a solution to remove shadows from images that uses top-hat transformations and morphological operators whereas [WYQ+07] proposes a solution to remove shadows based on a single Gaussian shadow model. [PT03] describes other alternatives and has references with further information about them.

Speed Estimation

Estimating the speed of vehicles is an important task because it can be used as a source of information for most of the traffic monitoring applications. This task is challenging because of the difficulty, the lack of precision of the used means and highly sensible data: not even humans, in videos with good quality, can precisely tell whose pixels belong to a car or not, blur from cars moving at high speeds makes this even more difficult. The exact position of the camera is not known (and can't be very precisely calculated). Vibrations and movement of the camera cause unpredictable variations as well. All these difficulties together cause enormous variations when geometrical information that relies on the position of the camera and observed cars is used. This happens because the methods employed are highly sensible: small changes in one variable may induce into big differences in the final estimated velocity. [AS07] achieves the estimation of speed based on the point of contact of the car with the road and [ASB08] uses a geometrical equation to estimate it.

Feature tracking

Tracking is also one important task. The most common type of tracking is vehicle tracking but it is also possible to track other features like velocity vectors. [CSY07] has an overview of tracking algorithms and tracks vehicles using scale invariant feature transform. This algorithm, applied to each vehicle, will describe its features like pixel values, key point locations and orientations into a 128 dimensional vector, which can be tracked in the following frame. [YMW+07] instead detects the corners of vehicles, tracks them using a Kalman filtering and groups the corners into vehicles.

3.3 Data Mining

Data mining is a process that extracts hidden patterns from a data set. These patterns have a very important meaning and their awareness enables great advantages to the information owner.

One example of this type of use is detection of patterns in clients [Ede], improving client relationship management and stock management. From these patterns it is possible to know what to expect from particular clients: their buying habits, if they are loyal customers or not, how long will they keep using the services, what special promotion could improve the relation with that client, evaluate the importance of a client and so forth.

Data mining, in a very simplistic perspective, is a three-step process[Ede99]:

Acquisition: the first step is acquiring information to process and describe it. Describing the information may include the use of statistical analysis over the acquired data, calculation of values, and so on.

Modelling: in this step, a predictive model is built based on patterns determined from known results. In order to have higher degree of confidence, from the acquired data, some samples, instead of being used for training – the training set – are used for testing the results – testing set. This way, when a model is created from the training set, it will be tested with the testing set.

Verification: after successfully creating a model, it is tested with real information. It is important to measure if the patterns detected in the samples, in fact, exist in the real world.

Over-fitting

In the explanation of the modelling step, it was introduced the concept of training and testing sets. These are used to address one problem: over-fitting. Over-fitting is one problem that arises from trying to extract patterns from very complex data, represented by a big number of variables with a relatively small amount of data. One over-fitted model describes noise in the data instead of real relationships between the information[FPSS96].

The phenomenon of over-fitting can be detected with the use of a testing set whose data is not used to train the model. Throughout normal training, the training error – error of model's estimation compared with the training set – and validation error – error of model's estimation compared with the testing set – continue to decrease. When over-fitting starts to occur, the training error continues to decrease but the validation error starts to increase[Wikb].

One over-fitted model will model patterns present in the training data that do not exist in the general data. This reduces the predictive capacity of the model when tested with data not used during the training stage.

Data Mining Uses

Data modelling can perform one of four classes of tasks:

Classification: in this type of problems, data must be arranged into a predefined set of groups. The algorithmic techniques used for classification modelling are Nearest Neighbour, Naive Bayes Classifiers and Neural Networks.

Clustering: clustering is a problem similar to classification. However, in this type of problems, there are no predefined groups so the algorithm must select the best way to arrange data.

Regression: the common use for this type of techniques is modelling some unknown function with the least error. A family of algorithms, namely Genetic Algorithms, can be applied in these problems.

Rule learning: these methods try to find relationships between the variables in order to make predictions.

Algorithms

As the core of the problem to be solved is classification, only the data mining algorithms used in classification problems are here analysed. Readers are referred to [FPSS96, Ede99, BST99, Moo] for a complete list of techniques.

Nearest Neighbour

K-nearest neighbour is one of the simplest classifiers used in machine learning. This method [CD07] uses a classified set of examples and upon reception of new data to classify, it will find the nearest neighbours and the data is classified with the most frequent class in the K nearest neighbours [WL08].

Naive Bayes Classifiers

Naive Bayes Classifiers are based on applying Bayes' Theorem, assuming strong (hence naive) independence between the variables. This type of classifier assumes that the features that characterise one class are independent from each other. This means that if one class is represented by the presence or absence of N features the classifier will assign each of those variables a contribution to classify the data into that class[DPP97] instead

of classifying by the presence of the features as a whole. This means that the classification of a sample data could be 10% of some class, which should be discarded because it has low likelihood to belong to that class.

Naive Bayes Classifiers use a simple algorithm that is very fast and can be trained to classify patterns involving a high number of attributes. Although the error rate for a Naive Bayes classifier is higher than some of the alternatives such as Neural Networks, it is still manageable for a range of applications and is more computationally efficient than Neural Networks[Vis].

Neural Networks

Neural networks try to resemble the way animal brains work[Gur97]. It is constituted of simple processing elements: nodes or neurons. Neurons are connected to each other (the number and type of connections may vary) and organised in layers. Each network has at least two layers: input and output layer which receive the inputs and return the final result of the network. Hidden, intermediary layers may exist. These layers will increase the complexity of the network and enable them to model more complex relations. Each connection between a pair of neurons has an associated weight which reflects the amount of effect that the output of one neuron will affect the result of the other (the output of one neuron becomes the input of the other). The output of a neuron is calculated with an activation function which gathers the input from the neurons connected to it, combines them in one of several ways, usually using a linear combination and produces the result by applying the value of the combined results to the activation function.

A neural network can be trained by changing the weights of the connections. Several activation functions, learning methods and topologies can be employed giving Neural Networks a great range of possibilities to change, improve and optimise.

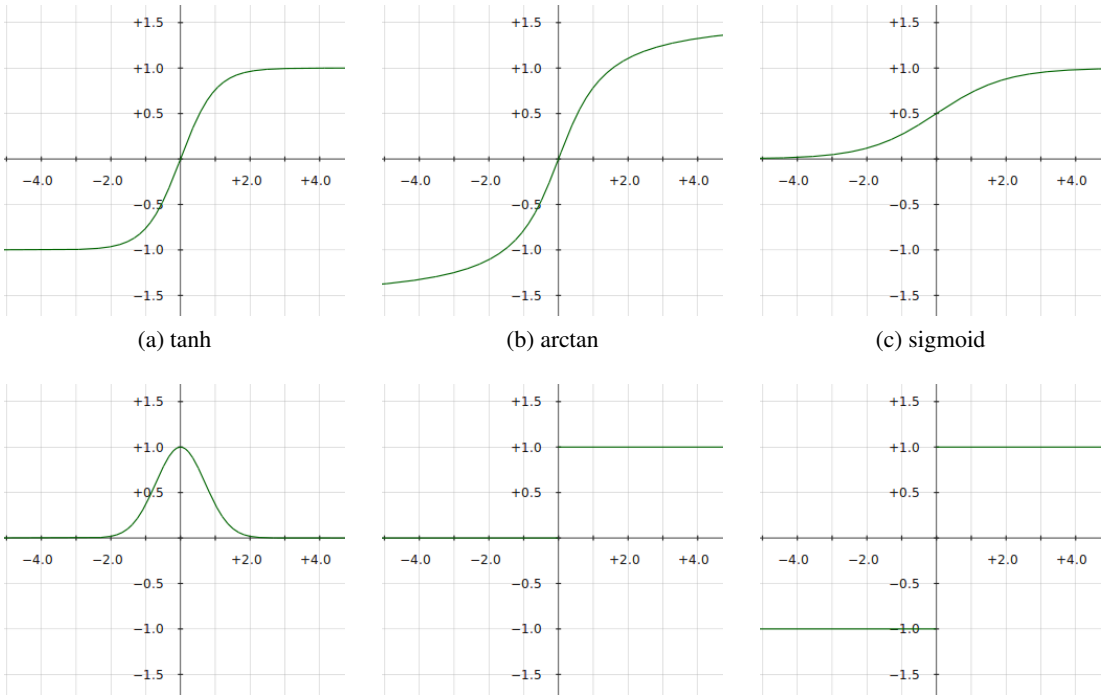
3.4 Neural Networks

As it is explained in section 4.2, Neural Networks were chosen to classify the data. This section describes them with more detail than last section's preview.

Activation Functions

The activation function is the function that maps the inputs of a neuron to its output. Activation functions play a major role in the performance of Neural Networks as they are the enablers of non linearity solving capacity[Lut].

Typical options to activations functions are sigmoidal functions such as logistic and tanh and the Gaussian function. Functions that produce both positive and negative values



(a) tanh (b) arctan (c) sigmoid

(d) gaussian (e) step (f) signum

Figure 3.3: Activation function examples

such as tanh or arctan usually can be trained faster than functions that produce only positive values such as the logistic function. Other possible activation functions are signun function or step function. This are known as hard non-linear functions whereas the others are known as soft non-linear functions[Lea]. When observing the corresponding graphics of these functions in fig. 3.3 it is easy to realise which are the hard and soft non-linear functions.

Neural Network Topology

There are two main network topologies: feed-forward and feedback networks. The former does not allow cycles to be created among the neurons' connections[Lut]. This means that the output of one neuron will never be part of the input of one neuron in the same or previous layers. The latter topology permits the existence of cycles in the network. Due this difference, after being given an input, the network must iterate potentially consuming a long time before producing the answer. Another characteristic of this network topology is the higher degree of difficulty involved in training them.

The number of hidden layers – layers placed between input and output layer that have no direct connections to the inputs and outputs of the network – may add a great deal of complexity to the networks thus increasing the time required to train the network and the

amount of sample data. However, the existence of hidden layers brings the capacity to model complex relations which may be fundamental to the problem in analysis.

There is no way to mathematically calculate the optimal number of hidden networks. Although typically more complex problems may require more hidden layers, it is possible that a given complex problem can be solved by a network with a relatively small number of layers. The solution to this problem is experimentation. Neural networks should be trained using a different number of hidden layers and the best ratio between modelling capacity and training time must be found. There are some approaches[Mat02] that allow the topology of the network to be created during the training.

Learning Methods

There are three types of learning: supervised, unsupervised and reinforcement learning.

In supervised learning[Lut], the sample includes the information about the expected result. With this information, during training the network will adapt the weights of the connections to meet the expected results. When the training stage finishes, inputs from which the expected answer are known, are presented to the network but without the expected result. The response from the network is compared with the expected results. This difference is the measurement of the quality of the network.

Unsupervised learning is a type of learning in which there is no information about the expected answers. This type of networks are used in clustering problems[Lea] and are expected to find salient features among the data and group them.

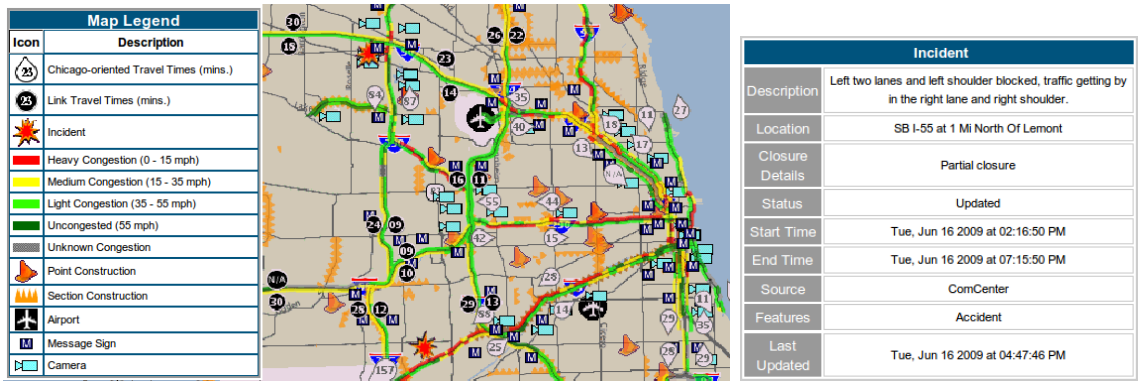
Reinforcement learning is in between the two learning models explained before. Although there is no information about the expected answers, the neural network will receive feedback from the environment – usually this is applied to agents which can interact with their environment. The feedback from the environment can be good or bad and based on that connection weights are adjusted.

3.5 Traffic Information Services

Worldwide

There are examples of sources of traffic information throughout the world that provide a wide and summarised range of information for the served regions. Due the fact that every developed country has a number of services to deliver information to the community, only the most remarkable services are here mentioned.

Illinois Department of Transportation[oT] created a website that successfully aggregates real-time traffic information from several sources and displays information to the user in a map. Additionally a news feed is available for users to receive the latest updates. Information retrieved includes travel time and congestion data, camera snapshots,



(a) Map overview (b) Information detail for some point

Figure 3.4: Traffic information in a map of Chicago

dynamic message sign legend information, lane closure and construction information, and incident data. In order to enable third individuals, organisations or companies to use the same information that they retrieve, there is a special form for them to apply for granting access to that information. Fig. 3.4 shows the map while showing particular information for one point that was clicked.

Triangle Software offers a set of services in the area of traffic monitoring and forecasting. It provides real-time nationwide traffic information broadcasting to the United States, being used by over 15 TV operators and monitoring over 65 cities[Sof, Wika]. These applications differ from the remaining due their usage in the United States and the great amount of information processed everyday. It offers, after paid registration, the possibility to watch personalised routes, real-time traffic-based routing, estimation of travel times, alerts to cell phones, trip advising to avoid peak hours and forecasting services[tT].

In Portugal

In Portugal, there are several traffic information providers. This survey has identified 4 main providers of information: Brisa, Lusoponte, Estradas de Portugal (EP) and Porto City Council.

On Brisa's website[Bri], the community has access to a service integrated with Google maps that shows several types of information: service and rest areas available along the highways, camera snapshots, incidents and information currently displayed in panels. Inside rest areas, there is a very complete list of subcategories that the user may specify: gas stations, optionally with GPL; parking for cars, caravans and trucks; auto support, bathrooms, showers, lunch area, playgrounds, and many other facilities.

Lusoponte[Lus] is in charge of maintaining and exploring two important bridges in Lisbon: *Ponte 25 de Abril* and *Ponte Vasco da Gama*. This way they also offer traffic information for these key points in Lisbon by making broadcasting snapshots which are updated every 5 minutes.

Estradas de Portugal[Est] offers a service that is integrated both with Google Maps and Visual Earth. On this map it is possible to see the messages displayed in the highways, see alerts for accident and access to snapshots or videos with real-time information (however, this video is provided through flash and can not be directly accessed). It also provides statistical analysis of traffic congestion levels over time, referencing the points where the data was measured in the map.

Porto City Council[Câ] offers a list of 52 web cams on a map where the user can decide which cameras wants to see. These cameras are placed throughout Porto and are offered in several streams. Each stream several cameras has aggregated and shows one camera at a time, rotating the camera being displayed every few seconds.

TV channels and radio stations also provide information. Some of these, such as RTP/Antena 1, simply redirect to EP's website. Others, such as Radio Comercial or Cidade FM, have their own source of information. This source is very likely to be based on the services described in this section but enabling the broadcast of information through the radio to a high number of drivers and also allowing users to do information reports by phone. This improves the quality of service as they receive updated information directly from drivers in exchange for routing advices.

3.6 Alternatives to Traffic Estimation

With the objective of enabling a true complete overview of traffic estimation techniques, some literature can be consulted. However, these techniques fall outside the scope of this project.

Arizona Department of Transportation[Sks01] performed a study on non-traditional traffic counting methods. It includes an assessment of currently available technologies, a survey of the State Department of Transportation practises, and a review of the literature.

Minnesota Department of Transportation[oT97] ordered experiments to be conducted on the performance of various non-intrusive traffic detection, counting, speed measuring and classification technologies. A technology, to be considered non-intrusive, should be installed, calibrated and maintained without closing traffic lanes. Experiments were made in a variety of traffic and climatic conditions in urban freeway and intersection locations so it could be a reliable and complete source of information.

3.7 Summary

This chapter has reported on the state of art in several ways. In the analysis of related works, it revealed several situations in which there was a need of processing videos to address particular issues. Most of the approaches were based on the difference between two images that may be the background estimation or the previous frame or relied on visual

cues. The tasks performed involved estimating features like speed or ambient conditions in order to improve the effectiveness through the use of specialised algorithms.

It also provided a theoretical background that has explained the techniques that can be applied to most of the vehicle monitoring applications tasks. It has analysed motion estimation algorithms, image subtraction techniques, vehicle segmentation and tracking strategies.

An overview of data mining techniques was presented. It has explained what it is, the main steps which it involves: acquisition, modelling and verification; has explained the problem of overfitting-ness and differentiated the family of problems addressed by data mining: classification, clustering, regression and rule learning problems. Additionally, algorithms used to address classification problems in data mining were clarified. These algorithms include K-Nearest Neighbour, Naive Bayes Classifier and Neural Networks.

Neural Networks were explained with more detail as they were identified as the best option to use in this project. What activations functions are and examples of them were given. Network topology was approached and three types of learning were discussed: supervised, unsupervised and reinforcement learning.

The current status of traffic information available throughout the world was reported. It has focused on particular applications used in the most congested areas of the world. It also made a status report of the situation in Portugal. The types of information made available on-line by the Portuguese concessionaires were described.

Finally, studies to find alternatives to monitor traffic using non-intrusive techniques were referred.

Chapter 4

Solution Proposal

A proposal for solution is explained in this chapter. First, it analyses how and which information should be extracted. Then, the process of learning and evaluating information will be explained, including an analysis of the benefits of the suggested solution and ways to overcome the possible problems. How extensibility will be maintained is explained in the final section. This chapter finishes with a summary of all information presented.

4.1 Information Extraction

The traffic congestion level, in a simplistic approach, can be understood as a non linear function between density of cars and their average travelling speed. If there is a low density of cars, their average speed is less representative of the congestion level as moving cars have the flexibility to allow others to move faster if they desire so. In this situation, the average travelling speed is the speed at which the drivers wish to travel. However, when there is a medium to high density of cars, the average speed has more weight on the congestion level because it becomes a limitation of the link as it is constrained by its capacity. Under medium or high densities, the average travelling speed is closer to the maximum speed at which that number of cars can travel in that link. This happens because there is less room for cars to allow other cars to move faster by overtaking them. When this effect starts get noticed, cars start moving at the maximum allowed velocity which may be lower than their desired speed.

Lower travelling velocities, resulting in longer times to travel the same distance is the main effect of congested roads. This is the reason for the interest in monitoring road congestion levels: estimate how much longer a journey is going to take.

This way the proposed solution will try to extract the velocity and density of vehicles from the videos it captures.

4.2 Evaluation and Learning

The desired final result of the algorithm in one evaluation, ranging from 1 to 5, which describes the congestion level of the information visualised on the video. As explained in the previous section, the final result is a non linear function of the travelling speed and density of cars. This solution proposes the use AI methods to translate the visual information into values. More precisely, neural networks might be used as they offer a good ratio between capacity to model complex relations and fast answering time.

To maintain the initial structure of the problem, it is suggested to use a set of neural networks: two networks will evaluate the velocity and density in the images and one third will receive the results from the two networks and give the final answer.

This approach has several advantages: first, it will give simpler, more specialised problems for the neural networks to solve. This will turn the neural network training into an easier and faster task. For example, if the algorithms can successfully evaluate the velocity of cars but have low accuracy with the density estimation, it is possible to only train the neural network responsible for evaluating the density of cars. Tackling problems separately also makes it easier for an operator that wants to train the network by giving sample results. Instead of providing an abstract information which would be the final congestion level, it could provide less subjective and easier to measure information like average speed quality and density level of cars. Yet, another advantage of using different networks to estimate and work with more specialised information is that this will support the general purpose of the application. It will enable the application to extract information not directly required by it – velocity and density of cars – which can also be used with other purposes.

The learning process will be carried out in a two-step process: first, during the analysis of video the operator has the option to inform the algorithm of the expected results. This can be made for one or more of the variables extracted by the neural networks. Then, the second step is to train the neural networks with the information learnt and store the resulting brain state.

One additional feature is the ability to store information in a separate file which can be used by other data mining tools to analyse patterns, further improve the neural networks or make other studies.

If a totally autonomous algorithm is required, it is possible to train the neural networks with a wide range of samples which would allow the networks to model the majority of situations they could face and load that network state when the algorithm is initiated.

In order to address the problem of not having control over the cameras and its positioning, robust algorithms will be applied, favouring the extraction of general attributes, treating the vehicles as a whole, instead of tracking and extracting information from each vehicle individually.

4.3 Extensibility

To promote the extensibility of the solution, there will be a careful design that minimises dependencies and assumptions between the several entities of the system.

To facilitate the addition of new features, tasks performed by different parties will be defined outside them so the tasks can later be invoked by other parties, without requiring to define those tasks more than once.

There are two types of tasks performed by the application. These will be named as *modules* and *algorithms*. Although *modules* regularly will apply several known algorithms, the difference is that *modules* provide some function or utility that could be used by several *algorithms*. One *algorithm* is a set of tasks – most of them invoked through *modules* – that have a more complete meaning or utility. For example, passing the last captured frame to an *algorithm* is expected to produce some kind of useful behaviour while passing it to a *module* is expected to transform or produce new information. Typically, one *algorithm* is constituted by calling several *modules* and conjugating their answers.

Finally, the user interface and the actions performed will be totally independent, allowing to have different interfaces for the same algorithms. This can be useful to allow skilled operators to train one algorithm while others, for example the general population, just access the results of the algorithm without controls to interfere with the algorithm.

4.4 Summary

This chapter explained that the two key features to be extracted from videos are velocity and car density and how their dynamics affects the congestion levels. With these two features it is expected that neural networks be trained to evaluate congestion level of the road under observation.

The use of neural networks is justified by its capacity to solve complex problems while giving fast answers after initial train. The hierarchical organisation of the networks was explained and it has been seen how it could help the training process of the networks and provide extra information for other uses.

The learning process will happen in a two-step process: first, sample answers are stored while the algorithm is running. In a second step, the learnt samples are used to train the new brain which is stored in the end of the process.

To cope with all the possible camera locations, robust algorithms will be used. It will be preferable to use information from the traffic as a whole than extracting information from particular vehicles.

To increase the extensibility of the application, several actions will be made: minimise the number of dependencies between the entities of the system, invoking common

tasks will be facilitated by the architecture of the system (avoiding repetition of task definitions), the actions performed will be grouped into two types: modules and algorithms and finally there will be a strict separation between the user interface and the actions performed.

Chapter 5

Prototypical Development

This chapter will explain how was this project developed and the reasons for those choices. The program's architecture is covered in here. Thie first section deals with the most generic classes that model the behaviour of the rest of the program. Also, a detailed list of the implemented modules is presented in the following section. Which algorithms were implemented, what do they do and how do they use the modules is the subject of the last section.

5.1 Program architecture

The complexity of the solution does not permit to present the class diagram of all classes at once. However, given the modular approach during development, the classes can be presented in a modular fashion, grouping related classes together. To further improve the comprehension of the subject under evaluation – program architecture – some auxiliary methods and attributes were disregarded and only the most generic classes are explained.

The objective in this section is to analyse the several classes implemented, their relations and how they interact but not going too much into details about the exact implementation.

Modules Overview

A module is a class that offers a set of methods that will change a given image or produce some type of new information. As explained in section 4.3, modules differ from algorithms – referring to the classes named as Module and Algorithm in this project – mainly in two ways: modules are a sequence of steps that can be reused throughout several algorithms and algorithms are expected to produce a more meaningful result than modules. It

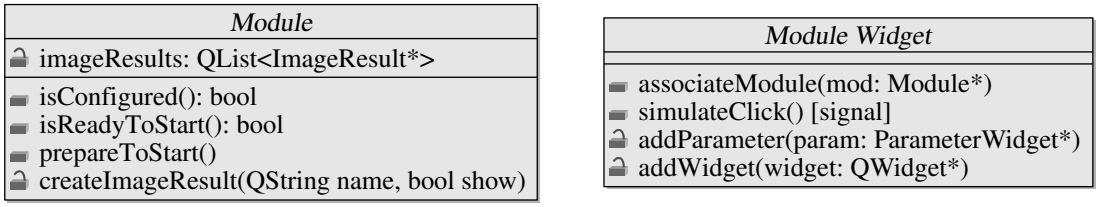


Figure 5.1: UML class diagram for module-related classes

is possible for one module to include other modules. Fig. 5.1 depicts the modules class diagram using UML notation.

Module Class

This is an abstract class, `Module`, which is inherited by all of the implemented modules. Inheriting this class ensures that a considerable amount of features do not need to be implemented by the subclasses as they were implemented in the base class, allowing the developer to focus on the task at hand. One example of that is automatically displaying module's results in the GUI (giving the option to hide and show the respective window).

This module defines a method, `isConfigured()` which returns if the method is configured or not. By default this function returns true (the basic modules do not need any configuration). This method should be implemented by subclasses that have different conditions to be configured.

A similar method is `isReadyToStart()`. This method follows the same principles as the one before. The reason for this module to exist is that sometimes, after configuring a module, some start-up task must be executed (for example, allocating objects with the size indicated in the parameter. This method returns whether a successful start-up has been made or not. By default this returns the same as `isConfigured()` and should be implemented in subclasses that have special requirements for the start-up process.

`prepareToStart()` is the default name for the function which does the start-up process after the module is configured but requires special tasks to be ready to start.

`createImageResult()` is a method which creates an image result with the given parameters, appends it to the class attribute `imageResults` and returns a pointer to the created object. This allows automatising of tasks performed with those results like displaying them on the GUI.

Module Widget Class

The `Module` class has a corresponding widget class named `ModuleWidget`. This class provides parameter configuration capabilities over the GUI. It should be inherited by all classes which provide the same capabilities to their corresponding modules.

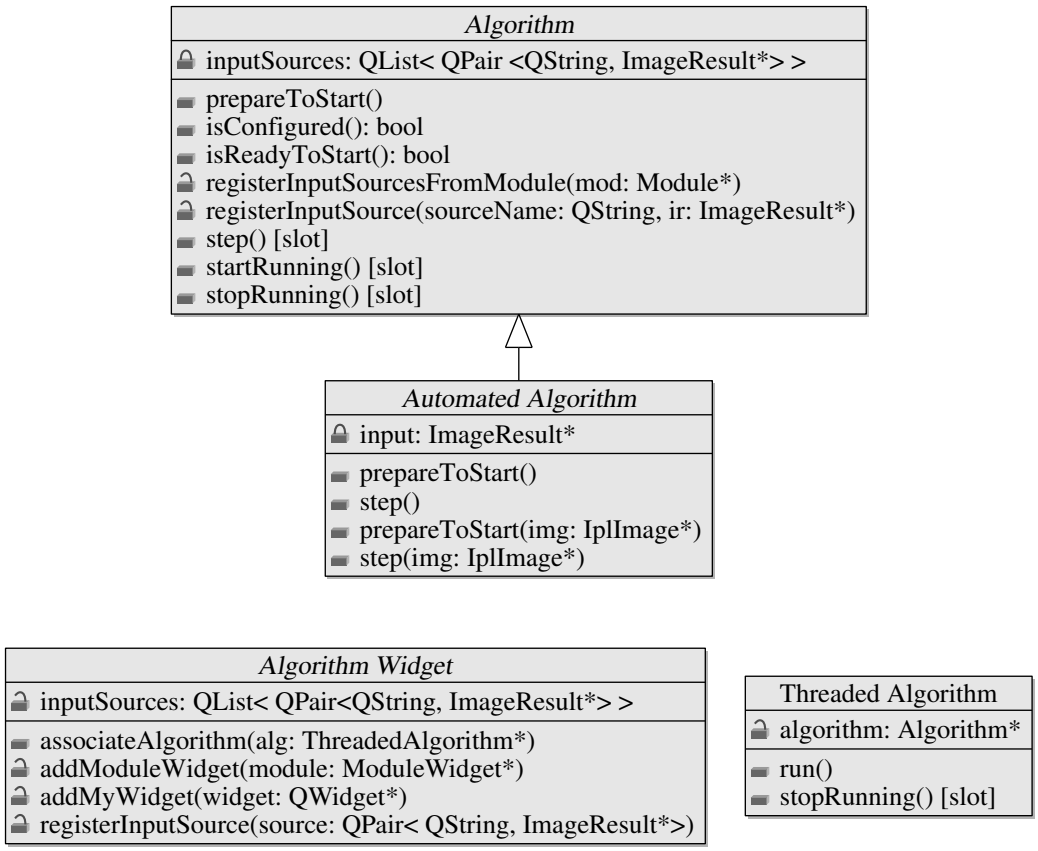


Figure 5.2: UML class diagram for algorithm-related classes

The method `associateModule()` is the responsible for connecting controls in the GUI to the module's attributes.

Algorithms overview

One algorithm is responsible for performing a set of tasks, using modules as a tool to achieve some goal. However, this is a simplistic perspective. Algorithms are also responsible for retrieving the frames from the capture device (which may be a camera or a video file) and each algorithm runs in a different thread.

Algorithm Class

This class represents one algorithm. Every algorithm should be a descendant of this class (eventually one indirect descendant). This will give the algorithms an important set of methods which allows them to work properly and with some automatising facilities like automatically handling output windows, opening capture devices and timing each step iteration.

It has a set of `inputSources` which represent images that are automatically assigned to output windows to display their results. It has a set of methods similar to the ones described in Module's class, namely `isConfigured()`, `isReadyToStart()` and `prepateToStart()`.

The method `registerInputSourcesFromModule()` will invoke `registerInputSource()` for each of the module's input sources, passing them to the algorithm.

`step()` is a purely virtual method which must be implemented by subclasses. The objective in this method it to do the algorithm's main task: retrieve the image to process and perform the sequence of steps to get the final result.

`startRunning()` and `stopRunning()` are the two methods responsible for doing the first tasks when an algorithm is commanded to start running and clean-up tasks when the algorithm is commanded to stop and terminate.

Automated Algorithm Class

This class is an utility class which provides some extra features to the Algorithm class which may be unintended for some particular algorithms. For an algorithm to benefit from it, it must inherit this class instead of `Algorithm` class. As this class itself is inheriting `Algorithm`'s class, the algorithms inheriting it, will actually inherit both classes.

This class automatically handles the opening of the capture device upon the call of `prepareToStart()`. Additionally, it will implement a `step()` method which retrieves the latest frame from the capture device. In case it fails (which happens when a video as come to an end) it will also stop processing the video). Additionally, `step()` will call a method with the same name but that it takes an argument which is the latest retrieved frame. It also notifies the parent class when it starts and finishes processing the iteration so it can effectively calculate the time taken for processing. This value is used to present in the GUI the algorithm running time.

Threaded Algorithm Class

This class is responsible for providing a multi-threaded behaviour to each algorithm. Algorithms are implemented without considering the multi-thread environment where they run. This class, when `run()` is invoked, will create a new `Algorithm` instance in a new thread where it will be running. The instance's class is indicated in one parameter given to the constructor of `Threaded Algorithm`.

Algorithm Widget Class

This is the corresponding class to the `ModuleWidget` but which is used in algorithms. It also has a method named `registerInputSource()` which receives the `inputSources`

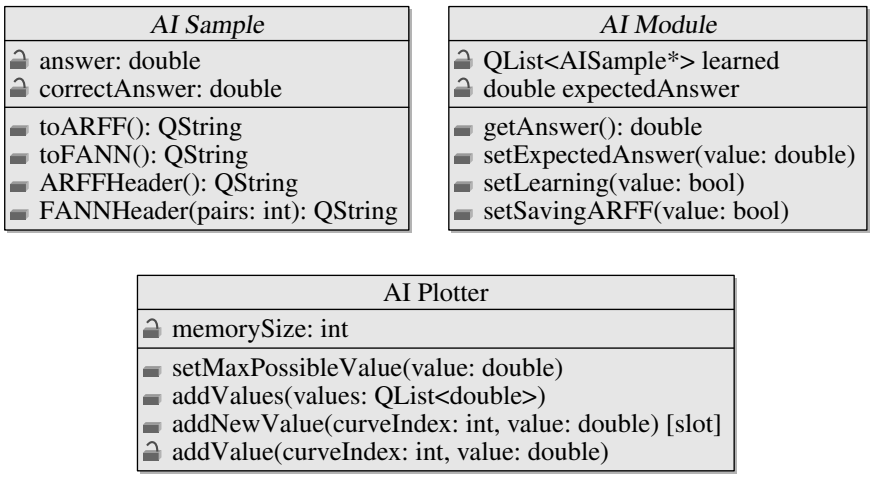


Figure 5.3: UML class diagram for artificial-intelligence-related classes

from the algorithm and creates the corresponding windows and controls which provides additional functions like presenting image results and showing or hidding windows.

Artificial Intelligence Overview

The group of classes described in this section is related with the artificial intelligence (AI) of the application. These are the classes that construct objects that can interact with the neural networks to translate them to evaluations.

Both `AIModule` and `AISample` classes are inherited by the corresponding classes that instantiate these classes to handle optical flow and background estimation results. A third pair of classes also inherits these classes to handle the results of the former results.

AI Module Class

An AI Module is a class responsible for loading neural networks, present them the latest inputs and return the corresponding outputs. It can also prepare files for training.

The attribute `learned` contains all the examples that were learned during the execution of the algorithm and which are going to be saved when it finishes. `expectedAnswer` is the correct answer for the current conditions, manually indicated by a person.

`getAnswer()` returns the answer of the network to the latest input. `setExpectedAnswer()` indicates the module which is the new value for `expectedAnswer`. `setLearning()` and `setSavingARFF()` set if the inputs and outputs are to be saved. The former will save the results in a format understandable by the neural networks for training and the latter stores them in a different format which is compatible with a wide range of data-mining software.

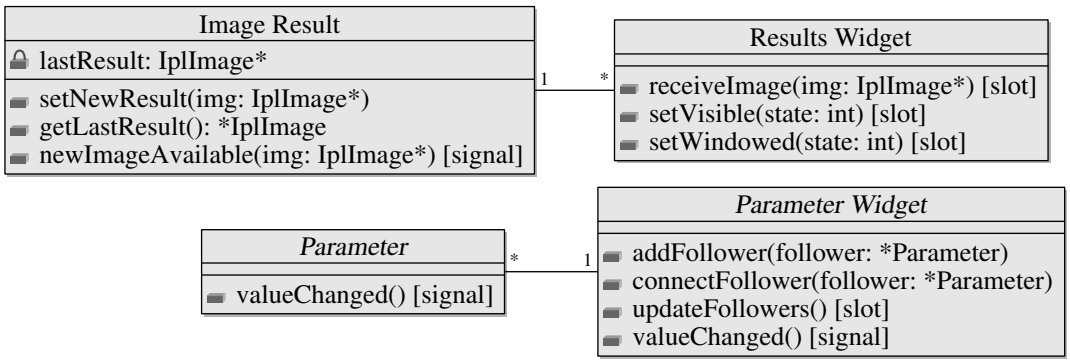


Figure 5.4: UML class diagram for UI-related classes

AI Sample Class

This class represents one instance of information which can be presented to the neural networks in order to get an expected result. It can also store the expected results and the answer given by the network.

`toARFF()` and `toFANN()` will return a `QString` of the information contained in the object, compatible with the respective format. This is used when saving the results of an algorithm to a file. `ARFFHeader()` and `FANNHeader()` print the begin of the corresponding files which has a special syntax to make the files readable by the associated programs.

AI Plotter Class

This class provides graphical analysis of the inputs given to the neural networks. `memorySize` sets how many values are kept in the memory of the object, thus setting the number of samples presented in the charts. Each chart may have several curves where typically each curve represents a different variable throughout time.

The method `setMaxPossibleValue()` will set the maximum value indicated by the plots. This will affect the scale of the *y* axis. If this value is not set, the axis will be automatically scaled to fit the currently displayed values. `addValues()` receives a list of values to be inserted into the plot. The first element is added to the first curve, the next element to the next curve and so on. In the end of this process, the plot is replotted. `addNewValue()` Adds the value to plot and re-plots it. The actual task of adding the value to the curve's information and maintain it internally is executed by the method `addValue()`.

Interface Overview

On this set of classes is used to present to the user the interface of some component that exists in the modules or algorithms being executed.

Image Result Class

This class represents one image which should be presented to the user. It may contain the inputs of modules or algorithms to allow a better understanding of what they are receiving, or can present intermediary or final results.

When `setNewResult()` is invoked, the attribute `lastResult` is updated with the provided image and this image is presented to the user though the GUI.

Results Widget Class

This class is the one that actually displays to the user the current results of the modules or algorithms in the form of images. It can present the images using OpenCV windows or widgets developed in the context of this project.

`receiveImage()` receives the image and displays it. `setVisible()` will show or hide the corresponding window / widget. If the internal widgets are being used, `setWindowed()` sets whether the corresponding widget is detached from the window that contains all results (hence having its own window). If the internal widgets are not being used (OpenCV windows are used instead) this is ignored.

Parameter Class

This class is an abstract class that represents some parameter of the algorithm which may be configured through the GUI. It has been implemented to handle int, double, boolean and colour (RGB) parameters. Each of these subclasses have the corresponding set and get methods. Additionally, this class has the signal `valueChanged()` that is emitted every time this parameter changes its value. This can be used to perform some special task (for example, allocating enough space for the new size of some array).

Each parameter object can be updated when the corresponding widget emits the `updateValue()` signal. This is handled automatically without any need of special attention from the developer.

Parameter Widget Class

This is the widget that corresponds to one parameter of modules or algorithms. It presents a controller to the user where he can choose the values for these parameters. This class is also implemented by classes to handle int, double, boolean and colour (RGB) parameters.

`addFollower()` will assign one `Parameter` to the values indicated in this widget. After this call, when the widget emits a new value, the parameter will receive and possibly handle it.

A `Parameter Widget` can have a live update behaviour or not. If it has, as soon as the user changes its value, the follower parameters are notified. Otherwise, they are only notified when the apply button is pressed.

Each parameter widget contains two controllers, one of them is always disabled. This control informs the user what was the latest value sent to the parameters that are following that widget. The first contains the chosen value to be sent.

5.2 Thread synchronisation

The developed application uses one thread per running algorithm plus one more for the GUI. This allows to run two or more algorithms at the same time without blocking the UI for instance. Due to advantages and disadvantages of several alternatives that are impossible to decide which is the best option for most situations, there is a set of variables in the file `main.cpp` which permits the configuration of the thread synchronisation method.

These variables are:

useMutex: this boolean states whether mutexes should be used to enforce thread synchronisation or not. If `true`, mutexes are used. If `false`, it defaults to use blocking queued connections to send QT's signals. This means that the execution of the step will block when it sets a new result and will wait for the program to copy the result to the corresponding `resultWidget`.

forceNoQueuedConnections: if using blocking connections when not using mutexes is not the pretended solution, this boolean should be set to `true`.

useCvWindows this boolean sets which type of window should the results (images) be presented. If set to `true`, the results are shown in native OpenCV windows. Otherwise, special widgets developed in this project can be resizing capability.

In section 6.3 there is a comparison of the different possible combinations in terms of efficiency and user experience.

5.3 Modules

This section describes the several modules that were implemented. Each module has a general description of what it does and how it achieves it. Each module may show the user several windows which are updated with the results of the last calculation. The meaning of each of these (possibly intermediary) results is also explained. Additionally, it is explained how to invoke the module and how to retrieve results.

5.3.1 Background Estimation

This module is responsible for creating and maintaining an estimation of the background. Based on the assumption that the background of a video will only change gradually and slowly over time (i.e. due to illumination changes), this module can detect moving parts in the given image by thresholding the difference between the image and the background estimation that has been constructed.

The implementation uses the Gaussian mixture algorithm [KB01] to produce the estimation. When the module is presented with one image through the call of `update()`, it will automatically update the background estimation with that image and subsequent calls which retrieve foreground information will consider this image as the one to compare against the background estimation.

After invoking `update()` with some frame, the following methods can be used to return information about that frame:

getBackground() This method returns an `IplImage` pointer to the background estimation.

getForeground() This method returns a pointer to an `IplImage` which contains the foreground image. As explained before, this is a threshold difference between the latest image provided in `update()`, with the background estimation. The returned image will have white pixels at the positions where the difference was bigger than a threshold and black pixels in the remaining positions. This means that white pixels are pixels considered to be moving and black ones are considered to belong to the background.

getImprovedForeground() This method returns the image returned by `getForeground()` after improving the quality of the moving area (smoothness) by applying processing to it. The processing that is made involves a set of closing operations.

getForegroundArea() This method will analyse the image returned by `getForeground()`, count the number of white pixels on it and return that value.

This module has four results that can be displayed to the user through the UI: the input image, the background estimation, the foreground and the improved foreground estimation. All of the above results are updated upon `update()` invocation but the improved foreground which is only updated if the method that calculates it is invoked. The reason behind this is to avoid using processor time whenever these results are not necessary.

5.3.2 Contour Retrieval

This module receives one image through the method `findContours()` and will calculate the existent contours. A contour is a line that connects points with the same colour.

This implementation was expected to work on binary images which means that it connects white pixels being a pixel considered white if its value is not zero. The result of the contour look-up could be drawn in another image through the method `drawContours()`. Optionally, both tasks can be executed at the same time by calling `findAndDrawContours()`.

5.3.3 Edge Detection

This module detects edges through the Canny edge detector algorithm[Can86]. In this algorithm, edges are found by analysing the first derivative of the image. First, it smooths the image using a simple 2D Gaussian filter. Then it will calculate the image derivative using the Sobel operator[Bov05]. The third step is to iterate throughout the filtered image derivative to detect edges. For this step, there is one of two thresholds to be used: one, smaller, is used if the current pixel is next to an edge and another, bigger, if the pixel is not adjacent to an edge. If the pixel has a bigger derivative than its threshold, it is considered to belong to an edge.

The existence of two thresholds is based on the idea that edges tend to be continuous, therefore if there is an edge in the neighbourhood of one pixel it is more likely to be in an edge as well. This allows to track a continuous edge which has some parts less sharp without increasing the number of false-edges present in the image results.

This module receives one image through the method `detectEdges()` and will update two results: the edges found and the filtered image used to detect edges.

5.3.4 Evaluator

This module is the module responsible for handling AI. It receives through the method `addResults()` pointers to objects that represent the latest sources of information to be processed or learned by the AI modules that process optical flow results and background estimation. It uses the parameters to inform the corresponding AI modules about the new data, querying them for the answer that is given by the neural networks. If both modules give an answer, it will provide the answers to the module that makes the final decision about the congestion level.

This module has in its interface several plots that represent the values received by each module over time. These plots are automatically updated when `addResults()` is invoked.

5.3.5 Morphological Operations

Morphological operations are a number of functions that can be applied to images that will change it in a particular way. These operations are used to improve the quality of the image to better suit the needs of other processing modules. This module has implemented

the following operations: image opening and closing, gradient, top-hat and black-hat calculation.

The functions can be invoked by `openImage()`, `closeImage()`, `applyGradient()`, `applyTopHat()` or `applyBlackHat()`. Alternatively, `doAllOperations()` can be invoked and all of the above methods will be executed.

As these operations are applicable in single channel images only, if coloured images are provided, the tasks are performed to each channel individually. This module provides results widgets for each of the operations which are updated when the corresponding method is invoked.

5.3.6 Optical Flow

Optical flow is the module that is used to estimate the velocity of objects in the video stream. When `calculate()` is invoked, proving the latest retrieved frame from the video, it will calculate the optical flow. Optical flow is calculated using the provided image and the previous image (which is internally stored). First, it finds key points in the initial image. A key point is a pixel which has characteristics that allow it to be distinguished among its neighbours. After all key points are detected in the first image, they are tracked in the next frame and the dislocation is measured.

This module produces three results: the latest frame, the previous frame and an image with arrows. This image is constituted by the previous frame with arrows that represent the movement direction of the starting point. If the drawing scale is set to 1, then the terminal point of the arrow is the position of the same point in the next frame.

To produce data for the neural networks, the function `getStatistics()` should be used. This function will take all the velocity vectors found in the last invocation of optical flow, calculate their magnitude and sort them by their magnitude. Then, it returns a vector with 8 percentiles of vector magnitude equally distributed. This allows a better understanding of how the velocity vector's magnitude is distributed and reasoning how many cars are moving at what speed.

5.3.7 Projection Correction

This module is responsible for correcting the perspective of one received image. Its objective is to minimise the problems that arise from projection, which makes the more distance objects seem smaller than the objects at a closer distance. This is a major problem when analysing car's movement as the dislocation's magnitude is dependent on the distance of the car to the camera.

The perspective is corrected by applying one transformation matrix to all the points of the image which makes the further objects have the same size as if they were closer to the camera. This way, the key point for this module is to find the correct transformation

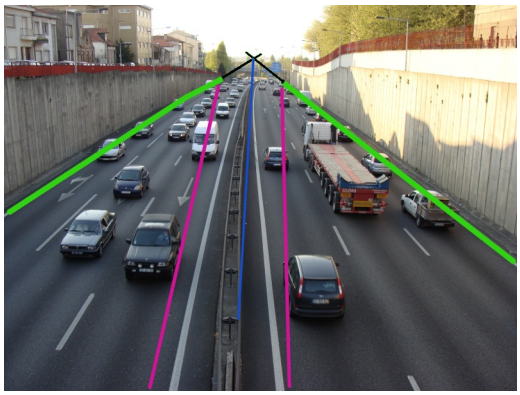


Figure 5.5: The parallel lines calculated in the projection correction module.

matrix for the video perspective. OpenCV has a function that receives 2 sets of 4 points. The first set defines two vectors in the original image and the second set defines the desired positioning in the corrected image. Therefore, this module will try to find these sets of points and use this function.

These points can be found if the vanishing point in the picture is found. As soon as the vanishing point is known, two lines can be drawn that are heading to it. This way, the 4 initial points are known. In order to find the 4 desirable points (the points in the image with the projection corrected) the two lines must be made parallel while not making them overlap (otherwise it would be impossible to find the transformation matrix).

In other words, what is made is outlining two parallel lines in the scenario, directed to the vanishing point. Considering the perspective in the original image, these lines will converge to a point – the vanishing point. As it was stated, these lines are expected to be parallel so in the corrected image these lines should have the same distance at all points. In order to achieve this, the end point of each line remains the same but the starting points are changed so the resulting lines are made parallel. Fig 5.5 shows the two sets of lines that are used in this process. In this image, two sets of lines are drawn. The green lines denote the parallel lines heading to the vanishing point. They were extended with the black lines until they intercept each other, identifying the vanishing point. From this point, a blue line with the average inclination of the initial two was drawn. The final step was drawing the purple lines which are parallel and equidistant to the blue line and have the same end point as the green lines.

Having explained the objectives of this algorithm, there is still one information that must be calculated: the position of the vanishing point. To estimate this point, all lines in the image are calculated. For each of the retrieved lines, the intersection point with the remaining points is calculated. The median of x and y coordinates of all interception points is calculated and assigned to be the vanishing point. As it will be seen in section 6.1.6, this is a very reasonable assumption.

The exact implementation of the explained algorithm is made with the following actions:

1. the first step is to find the edges in the given image. This step is explained in the module of edge detection.

2. using the image produced in the first step, a line detection algorithm is applied. This is done through the Hough line detection algorithm which is implemented in OpenCV.

3. after detecting lines, some of them are excluded as their direction is considered to be too much horizontal. This happens because a considerable number of objects – cars, road signs, buildings – create several horizontal lines which can be considered noise in the task of finding the vanishing point.

4. considering only the remaining lines from the previous step, for each of the lines, calculate the intersection point with each of the other lines. Calculation the median of the x and y component of the intersection points, the retrieved values constitute a good approximation for the vanishing point position.

5. to create the four points in the current image, it starts by placing two points some distance apart, whose centre is also the image centre. The direction in which they are separated is perpendicular to the direction from the centre of the image to the vanishing point. These will be the starting points of the two lines. The end points will be points in the line that connect the starting point previously calculated to the vanishing point and will have a given distance to the initial point.

   To calculate the points in the corrected image, it uses the same end points as the ones used in the initial image. The starting points are placed the same distance apart from the initial point as they were in the original image, but they are placed in a line which is parallel to the line that goes from the centre of the image to the vanishing point.

6. With the eight points calculated in the last step, the warping function of OpenCV calculates the transformation matrix which represents the intended correction.

This module can display the edge image, filtered image, an image with several lines that were used during processing when `calculateCorrection()` is executed and the result of applying the correction matrix to the input image. After this call, the module will internally store the correction matrix used and subsequent frames can be corrected by issuing `correctProjection()`. This method will simply use the same matrix calculated before and does not update any correction estimation.

Unlike most of the other modules, this module's main method – `calculateCorrection()` – is not supposed to be executed in every iteration or retrieved image otherwise the comparison between consecutive frames would be seriously compromised by the small differences from the two correction estimations. After the initial call of `calculateCorrection()` (which can be repeated whenever desired), the method `correctProjection()` should be used to apply the same transformation to the image.

5.4 Algorithms

5.4.1 Background Estimation

The purpose of this algorithm is to compare different parameter setups and its effects in the background estimation. It is composed by one single module, namely the background estimation module and has one additional output result which is the foreground pixels, detected by the background estimation module and applied with the colour red in a copy of the original image.

5.4.2 Projection Estimation

This algorithm provides a way to deeply analyse the projection correction module. Instead of calculating the corrected projection once, this algorithm corrects it in every frame. This has the advantage of testing the correction algorithm for a number of images. The differences of the corrected projection between different images of the same scene gives the notion of how robust the parameter set currently employed is.

5.4.3 Decision Algorithm

The evaluation of the road conditions is presented in this algorithm. It can be used both to evaluate a video and to produce samples to train a neural network. Although this module uses the background estimation, projection correction and evaluator modules, only the configuration interface of the latest is shown.

Upon algorithm initiation, it will get the first image from the video and use it to correct the perspective, using this correction throughout the rest of the video. In each step, the background estimation module is updated and the optical flow module computes the velocity vectors. AI samples are created from the results of the two modules and are provided to the evaluation module which will evaluate them and, if desired, save the results.

5.5 Summary

This chapter has explained how this system was implemented. It has given an overview of the program architecture, explaining the structure of the modules, algorithms, artificial-intelligence-related classes and the interface. In this section, the most generic classes which model the way the rest of the classes interact were explained.

The possibilities to deal with thread synchronisation were also analysed. It has explained the several implemented modules and algorithms, as well as their approaches to perform the designated tasks.

Chapter 6

Results Assessment

In this chapter the achieved results are presented and analysed. The next section will report on the achieved results in the several implemented modules. For each module, the results under different conditions are analysed and the effect of changing module parameters is demonstrated.

Measurements of the time that each module's functions take to execute with different parameters and images is analysed. The effects of the different synchronisation methods are also analysed.

The last section of this chapter summarises all the information herein presented.

Throughout this chapter several examples are presented using a collection of videos which were recorded during the first weeks of this thesis. In every example, the source video and the used parameters are presented to allow the reader to better understand the results. Appendix A describes the conditions in each video, allowing the reader to have a better comprehension of time-based results.

6.1 Modules

6.1.1 Background Estimation

This module receives images and uses them to model the background of the image. With a successfully constructed background estimation, the parts of the image that have a difference greater than a threshold, can be considered as not belonging to the background. In scenes where the real background is static, like most of the scenes filmed by traffic cameras, the regions not belonging to the background are areas in movement.

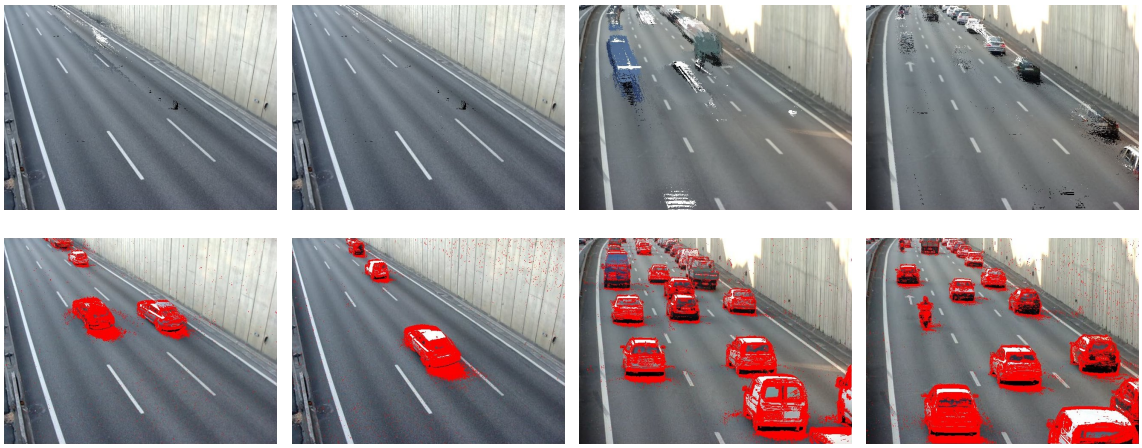


Figure 6.1: Background estimation under different density of cars

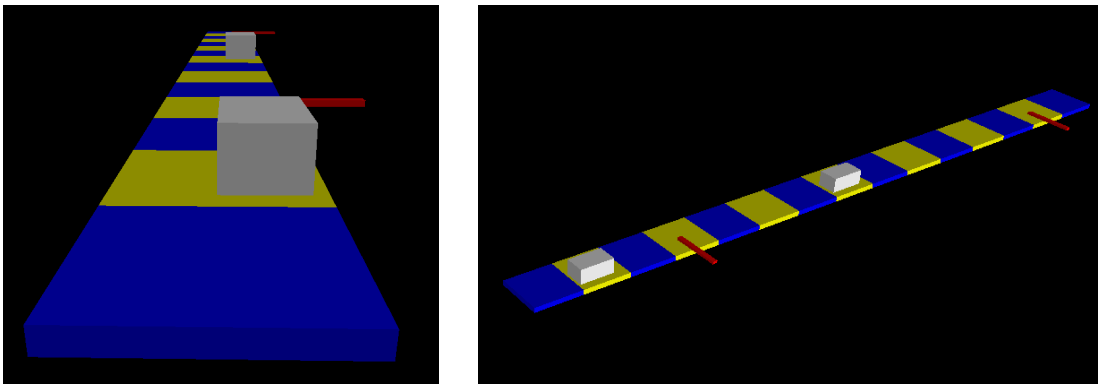
The oftener this algorithm has information about the background, the better its background estimation is. This means that in situations with dense traffic, the road (background) is seen less often, hence the background estimation is more difficult. For this reason, both situations are demonstrated here.

In order to facilitate the evaluation of the results, two images for each example of the algorithm are shown: the background estimation and the foreground mask applied in red to the input image.

Figure 6.1 shows the results of this module executed with two different videos. The first column has a scenario with low density of cars, after 10 seconds of processing. In the second column, the same scenario is shown but after a longer processing time. The third and fourth columns have another scenario, with a higher traffic density, and show the results after 10 seconds of processing and around one minute of processing. The top row contains the background estimation and the bottom row contains the results of applying the foreground mask on red to the last image received from the video. The parameters used are listed in table 6.1

Scenario	low density	high density
Video	1322	1292
Window size	200	200
Background threshold	2	2
Standard threshold	2.7	2.7
Number of gaussians	2	2
Minimum area	1	1
Initial weight	0.05	0.05
Initial Variance	40	40

Table 6.1: Parameters used in background estimation module



(a) Frontal perspective (b) Lateral perspective

Figure 6.2: Comparison of the portion of road hidden from cars at different distances from the camera.

At first glance, it can be seen that the algorithm successfully estimated the background with a reasonable accuracy and has detected moving pixels belonging to cars.

From the comparison of the images under good (low traffic density) and bad conditions, it is noticeable that the background estimation has produced a less accurate background estimation. This happens because in order to produce a good estimation, it must have access to the background pixels with some frequency. In dense environments, cars slower and the speed reduction allows them to travel closer to each other. Under these circumstances, the distance between cars is not enough to allow the background module to 'see' the background as often as needed to produce a good estimation.

A more deep analysis shows that the estimation is worse in the areas of the image where traffic moves slower or where it is farther from the camera. Analysing both situations, a common denominator is found: the apparent velocity – the dislocation actually performed in the image received – is low.

In cars distant from the camera, there is also another problem which works together with the apparent slow movement: the car's height will occlude a longer extension of road. This is demonstrated in fig. 6.2. This image represents the same scene seen from two different view points. Each box simulates a car and have the same size. Each road patch, yellow or blue, has the same size. The red marks show the last point of road occluded by each car when viewing from the frontal perspective.

A last remark about background estimation under dense traffic: although the background estimation is not the most correct, the chosen parameters were selected with production of the best foreground mask in mind. Other parameters were able to produce a better background estimation but did not do the task of segmenting the foreground so well. As this module is used to count foreground pixels, the final parameter set was the one that could better detect foreground pixels in most of the conditions.

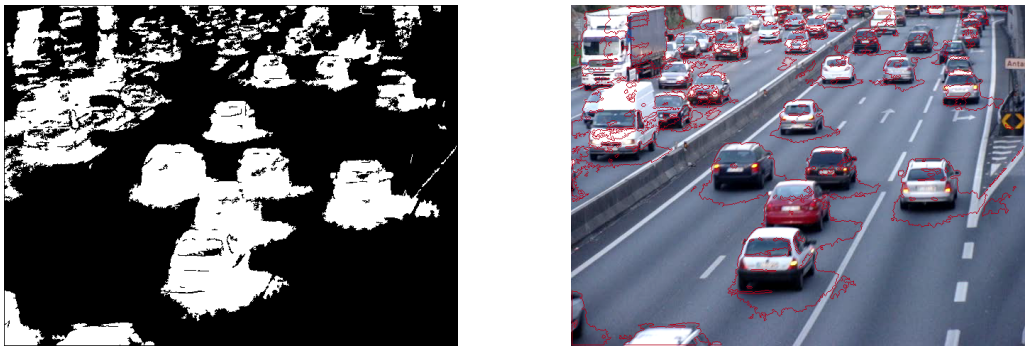


Figure 6.3: Example of contour detection (red lines).

6.1.2 Contour Retrieval

Contour retrieval module detects contours in Gray-scaled images, connecting non 0 (black) pixels together. Fig. 6.3 shows the results of this module. Although not represented in the given example, this module retrieves the information about the contours in a non-visual way, being possible to process that information with mathematical operations. It is also capable of detecting contours inside other contours.

6.1.3 Edge Detection

This module detects edges in a given image. This task is a two-step process: first the image is filtered and only then the edge detection algorithm is applied. Edge detection module has three parameters. These are the filter size and two threshold values used to detect the start of edges (the higher threshold) and line's continuation (the lower threshold).

Several experiments in different videos and with different parameter sets were made. Some of the achieved results are presented in fig. 6.4 which shows the effects of using different filter sizes. Fig. 6.5 presents the effects of changing the threshold values for the edge detection algorithm. The selected examples were chosen as they are the ones that allow to better describe the experimentation findings. The parameters used for these examples can be found in table 6.2.

Situation	Video	Filter size	Threshold 1	Threshold 2
Filter experimentation	1322	1-11	40.0	70.0
Parameter set combination #1	1322	7	172.5	220.0
Parameter set combination #2	1322	5	0.0	220.0
Parameter set combination #3	1322	3	220.0	220.0
Parameter set combination #4	1322	3	130.0	130.0

Table 6.2: Parameters used in edge detection module

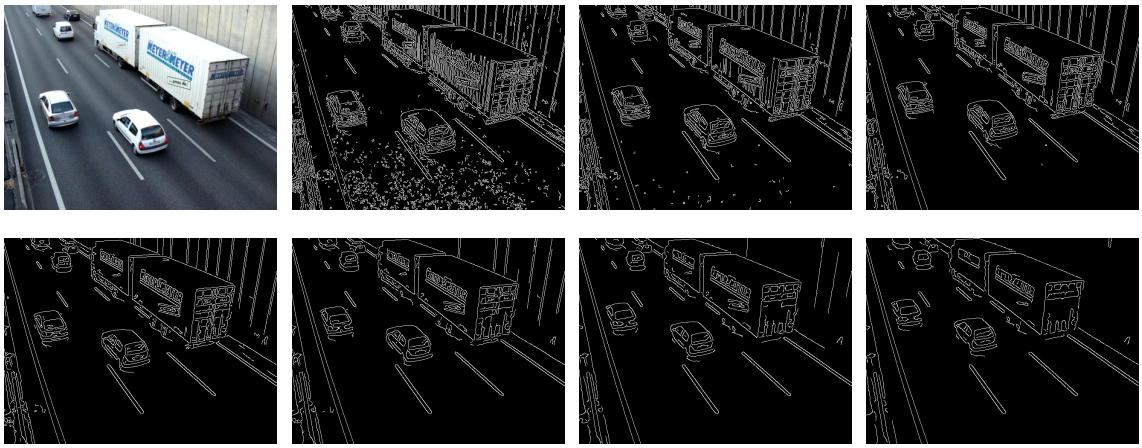


Figure 6.4: Comparison of edge detection using a different filter size.
In this is a sequence of images, the first image presents the first frame analysed from the video. The following frames show the edge detection result applied to the following frames, using different filter sizes. The values used were (in the images from left to right, from top to bottom): 1 (which means no filtering), 3, 5, 7, 9, 11 and 13.

While experimenting different filter sizes, it was noticed that the size of filter has an ample effect on the first levels of it, being an excellent tool to remove noise from the final detection. However, once medium-sized filters start to be used, increasing the filter size produces small improvements. This is highly correlated with the fact that the filtered image also does not have considerable differences when the filter size is increased at medium-sized filters.

This is explained by the filtering algorithm: for each pixel, it will sum the values of the surrounding pixels and the final value for that pixel is an weighted average where a pixel has more weight as close it is to the centre pixel. The surrounding pixels considered in the average are defined by a square whose side's dimension is the designated "filter size". Table 6.3 shows the effect of increasing the filter size in terms of the growth rate produced by the addition of the new pixels to the filter. This effect is much more considerable if it is taken into account that the farthest pixels have less weight in the weighted average and that this relation is not linear. For example, in a filter with a width of 7 pixels, the central pixel may weight 250 times more than the four farthest pixels combined (the filter's corners). This exact value depends on the standard deviation used on the Gaussian filter.

As it can be seen in fig. 6.5, edges were detected in the interior of cars (and some other objects). This edges often correspond to colour changes caused by shadows, reflections or objects shape. As the results from this module can be used to detect objects, these edges may become undesirable. A way to reduce them is increasing the threshold for line detection.

However, only increasing the threshold may create some problems as edges start being not detected. To cope with the loss of edges, the filter size must be reduced. This way,

Figure 6.5: Edge detection using different parameter combinations (described in table 6.2).

edges will be less blurred at the filter stage. When both parameters are changed (filter size and threshold value) it was noticed that some edges disappear while the most bold edges have remained.

The strategy explained in the last paragraph is represented in fig. 6.5. The first image shows the example of a situation in which the filter is too strong and the edges start fading out. In a different situation, the fourth image represents a case in which the applied threshold is lower than the ideal one. Both situations were addressed in the second and third images. Although the second image has a slightly better result, the third image has other advantages which are explained later on in this chapter. It could be worth to noticing that in the third situation, applying a filter with the size of 5 instead of 3 produced a result with less noise, more similar to the second image.

Another finding is related to the difference between the two thresholds. As explained in chapter 5, the smallest of the two is used to track the continuations of edges and the biggest is used to detect the existence (in other words, start) of edges. Although using two distinct values may detect edges that otherwise would remain undetected, it has been noticed that when the threshold difference is big, the results become more unstable. This means that when analysing a sequence of images, some edges whose characterisation is close to the higher threshold at some point, have a similar probability of being considered edges or not. The visual effect of this is that some of the edges appear to be blinking as they are detected in some frames but not all.

When the two thresholds have a more similar value, this effect is less outstanding

Filter size	Area	Increased Area	Growth rate
1	1	-	-
3	9	8	800%
5	25	16	178%
7	49	24	96%
9	81	32	65%
11	121	40	49%
n	n^2	$4*(n-1)$	$\frac{n^2-(n-2)^2}{(n-2)^2}$

Table 6.3: Parameters used in edge detection module

(a) Input (b) Gray scale (c) Gradient

Figure 6.6: Gradient processing steps (parameters described in table 6.4).

because the lower threshold, which now is higher than in the original situation, doesn't allow the edge (which had only a small portion above the higher threshold) to extend through the image. In this situation, what happens is that a small number of pixels appear blinking.

Blinking edges in the information extracted over time may introduce noise into a non-robust algorithm's results, reducing its accuracy. On the other hand, if the algorithm is robust enough to handle those differences (blinking edges), or does not rely on them, it may have access to better quality information. In either case, it is necessary to trade off between results quality and robustness.

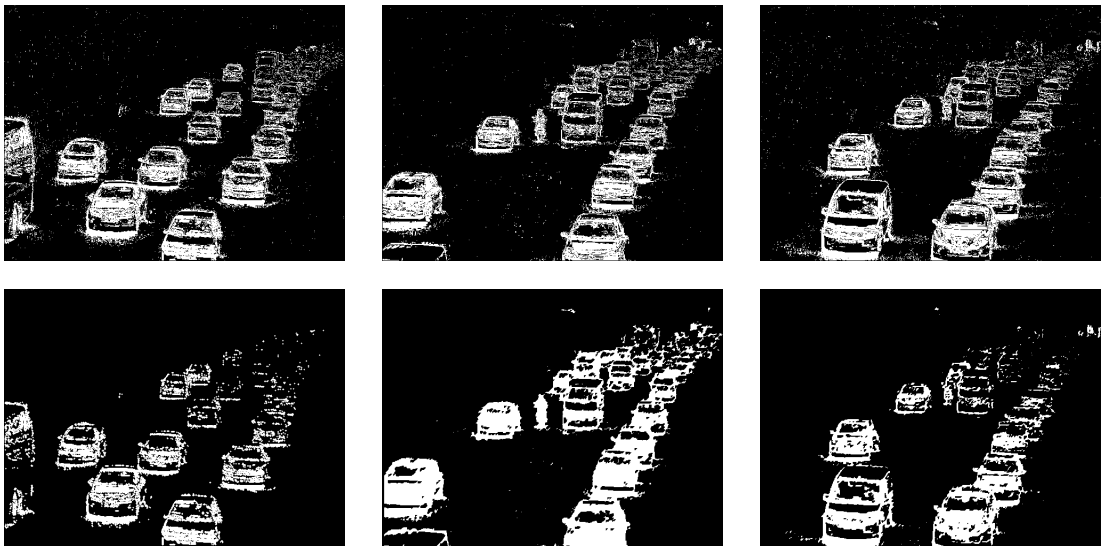
6.1.4 Morphological Operations

This module has five different modules. Each of them is described on its own section. All the different operations were subject to processing with different image types: RGB, Gray-scale and binary images. The following sections describe the experimentation results for each component.

Gradient

This operation creates an image in which each pixel has a value that indicates the contrast intensity in its neighbourhood. Experiments under several types showed good capacity for edge detection, specially when using Gray-scale images. Fig. 6.6 shows an example of these results.

When applied to 3-channel images, such as RGB or HSV images, it is able to detect the contrast in each channel individually. Depending on the contrast of each channel, if the result image is interpreted as an RGB image the result image will have the colours constituted by the contrast of each channel. This means that if a pixel has a similar change in all components, it will be have a white or Gray colour, being as bright as the amount of contrast in that area. On the other hand, if one pixel only has contrast in one or two components, it will have the colour constituted by the mixture of both components'



(d) Open (e) Close (f) Open and close twice

Figure 6.7: Open and close examples (parameters in table 6.4).

colour. If a pixel has only one contrast in the hue channel, and assuming that the hue channel is interpreted as red in the result, that pixel would have a red colour.

This feature can be very interesting in particular applications, where changes in particular components of the image are sought. Although the example of interpreting results as an RGB image was presented, this is not required. However, this is probably the most interesting method to analyse 3-channel (hence three different variables) results using one single representation, as one person is capable of understanding the colour components of the results and assigning them to the original components. Besides allowing compact information presentation, it also may enable the discovery of other relations when particular colours appear in the final result: primary colours represent pixels where mainly one of the components has contrast; intermediary colours represent pixels where two or three components have contrast, eventually with different amounts in each channel; if Gray or white colours are found, then all the components had a similar contrast.

Experiments on the parameters' values concluded that the best possible combinations are doing one iteration with an element size of 2 or 3, or two iterations, with a component size of 2. Element size is the side's size of the block of pixels used in those calculations at each step. Increasing these values produced a better defined and clear result image while bigger values rendered an unfocused image.

Open and Close Image

This two operations are very similar and perform complementary operations. Although they can be applied to coloured images, affecting each channel individually, the results

(a) Input for top hat (b) Top hat (c) Input for black hat (d) Black hat

Figure 6.8: Top hat and black hat results (parameters described in table 6.4).

did not return any interesting results. On the other hand, these functions are very useful when operated in binary images (black and white only).

Opening an image removes white pixels (changing them to black) in areas where there is at least one black pixel. Closing an image will remove black pixels (changing them to white) where there is at least one white pixel.

The two operations are very useful to improve the quality of the foreground mask detected by the background estimation module. Fig. 6.7 shows several examples of these methods applied to foreground masks. The first row presents the input images and the second row presents the result of applying the corresponding operation(s) to the given input. Improvements in all images were achieved, either by removing noise (which is not very perceptible on the images because they were scaled down) and by grouping together white pixels, which have better defined moving objects.

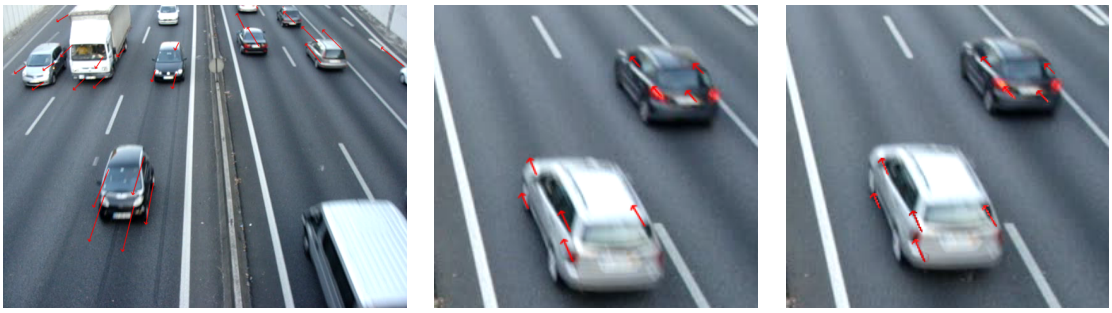
It was noticed that increasing the element's size allows it to affect a bigger area, which means that bigger areas my be transformed into black or white (depending on the operation). Images produced using a bigger element exhibited less similarities with the initial shape, thus becoming a less reliable enhancement of the original image. Using more iterations allows the effect of the algorithm to spread more. This means that openings will spread more black pixels around the already existent black regions and closing operations will spread more white pixels around white areas.

One sequence of open operations (with less iterations) and closing operations may effectively improve the quality of the foreground mask estimated by the background estimation module and other binary images that were subject to noise.

Top Hat and Black Hat

These two transformations are similar although they return the opposite results: top hat returns bright areas based on the brightness of the element's area and black hat returns dark areas. Both transformations can be used for feature extraction and detection.

Top hat has successfully highlighted car lights during daylight conditions. Although there were no videos available recorded at night, it is strongly believed that this can be used to improve traffic monitoring in low lighting conditions and possibly at night. One



(a) General scenario (b) Detail with unscaled arrows (c) Detail in the next frame with arrows from the last frame

Figure 6.9: Optical flow example (parameters in table 6.5).

example of this is presented in fig. 6.8b where the closest car was breaking (thus had the breaking light on and one car in the other direction also has the lights on). Another feature highlighted by top hat is road marks. The use of black hat was able to suppress some objects such as road marks.

6.1.5 Optical Flow

Optical flow is a module used to estimate the velocity of the objects in the scene. It compares two consecutive frames and tracks the motion of key points in the two frames. After tracking all key points, the module extracts statistics about the motion vectors which are provided to the AI module which will evaluate the data.

Changing the module parameters caused differences in the results of the module. The number of pyramids (*nPyramids*) is used in filtering operations. This way, reducing its value may introduce some noise in the form of false movement vectors being found.

The default values for the parameters which may cause the processing to stop (maximum features, maximum iterations and maximum epsilon), were relaxed but that produced no rise in the number of detected movement vectors. This means that these parameters weren't being reached. As having these parameters over-relaxed does not force the algorithm to process for a longer period, it was decided not to change them.

Image	Operation	Source video	Iterations	Element Size
6.6	gradient	1346 (Gray scale)	2	2
6.8	top hat	1346 (RGB)	1	12
6.8	black hat	1346 (Gray scale)	3	7
6.7	open	1293 (Foreground mask - binary)	1	3
6.7	close	1293 (Foreground mask - binary)	3	2
6.7	open and close twice	1293 (Foreground mask - binary)	1	2

Table 6.4: Parameters used in morphological operations

The minimum distance affects mainly the number of motion vectors detected in each car. As the used algorithm does not have any type of information about the objects in the image, several points from one moving object are tracked. The most often tracked points in cars are mirrors, tires, wind shield corners, bumpers and vehicle edges. In order to reduce the number of motion vectors per object the minimum distance parameter can be used. However, this is not recommended because smaller objects like motorcycles moving close to other objects may lose all motion vectors assigned to its pixels. A better strategy would be to group motion vectors by detected objects in the image (which are detected on a parallel process).

The parameter that limits minimum quality of the key points, if too low, may allow false motion vectors to be detected due to the noise in the input image. If this value is too restrictive, movement stops being detect. The experiments conducted have found that this value should be no more than 0.25 or the movement is badly detected. However, for values above 0.05 faster cars, if blurred, may not be detected as well. For values below 0.01 the results start incorporating noise. Despite the presented numbers, it is important to be aware that these values are severely affected by video characteristics like noise, frame rate, distance to the road, etc.

The evaluation of fig. 6.9 shows the accuracy of this module in the two images on the right. The first of the two shows the previous frame and the motions vectors pointing to the position of the destination of the corresponding pixel in the next frame. The second image shows the current frame, and kept the arrows exactly in the same position to provide a comparison basis. It is clear the precision with which the motion of the tail lights, tire and several points of the windshield were detected.

6.1.6 Projection Correction

This module corrects the projection of one image in several steps. First, it detects edges using the method explained in the edge detection module. Using the results from the first

Parameter	General scenario	Detailed scenario
Video	1388	1388
Max features	400	400
nPyramids	5	5
Window size	5	5
Min quality	0.1	0.1
Min distance	5.0	5.0
Max iterations	200	200
Max epsilon	0.25	0.25
Min hypotenuse	2.0	2.0
Length factor	1.0	6.0

Table 6.5: Parameters used in optical flow

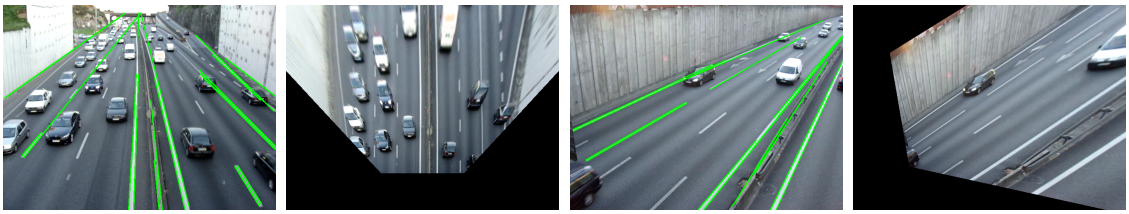


Figure 6.10: Projection correction examples (parameters in table 6.6).

step, it detects lines by applying the Hough line detection algorithm. Then, for each of the found lines, it will calculate the intersection point with the other lines. Calculating the median of each of the coordinates of the intersection points, it estimates the position of the vanishing point.

Using this point, it creates two lines which are directed towards it. For each of these two lines, it selects two points which are going to be the start point and end point of a line segment. Then, it creates two other line segments, with the same length and ending at the same point as the previous lines. This lines are made parallel, by averaging both initial lines' inclination. From the points of the defined line segments, it detects the transformation matrix required to transform the two initial lines into the other two. After the initial calculation of this matrix, its values are internally stored so the same matrix can be applied to subsequent frames.

As the first step, edge detection, is the task of a particular module of this project, the analysis of this part of the process is presented in the section 6.1.3.

The second step is line detection from the edge results. This process can be seen in fig. 6.10, more specifically in the first and third images. The results demonstrated very good capabilities in line detection. The lines have been successfully detected, including under dense traffic where the conditions make it more difficult to spot road marks and other reference lines.

The objective of line detection is to provide an estimation of where the vanishing point should be. In order to accomplish that, it is expected that the majority of the lines

Parameter	First example	Second example
Video	1387	1387
Threshold 1	40	120
Threshold 2	70	120
Filter size	15	5
Maximum line gap	50	50
Minimum line length50	50	
Line threshold	100	150
Exclusion margin	25	25

Table 6.6: Parameters used in projection correction

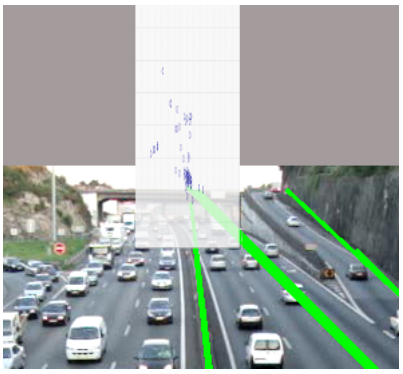


Figure 6.11: Scatter plot scaled and overlaid in the frame.

are heading to the vanishing point. Experimentation and the presented results demonstrate that this assumption was correctly stated.

Another assumption was that after calculating the intersection of each of the lines, the median of the intersection coordinates would approximate to the position of the vanishing point. Fig. 6.11 shows one frame and the corresponding estimation of intersection points in the form of a scattering plot. The plot was scaled and positioned to match the exact points that the plotted values refer in the original frame. As some of the intersection points included were located outside the original image, the Gray area was added.

From the analysis of this figure it is clear that the intersection points concentrate in one small area where the vanishing point is present, proving the correctness of the assumption. Then second and fourth images of fig. 6.10 show the result of applying the calculated transformation matrix to the input image. One measure to evaluate the quality of the approximation is comparing the road marks to check if they are parallel instead of converging to one point like in the original image. Table 6.7 presents more details about this.

A test for the robustness of this algorithm is calculating the transformation matrix for each frame in the video, applying it to the input image and comparing the differences in the projection. This experiment obtained good results. Similar information is presented in the fig. 6.12.

This figure is the scatter plot of three non-consecutive frames from the same video. Each colour (red, green, blue) represents one different frame. The lines point at the final position for the vanishing point for the frame with the same colour. Each plotted circle represents one intersection between two lines.

This plot shows that this algorithm returns similar results with different images from the same video sequence. The vanishing points were close to each other in all estimations. In fact, two frames shared exactly the same vanishing point.

Parameter experimentation has concluded that lowering the filter size and increasing thresholds (both for edge and line detection) has returned better results, with more lines

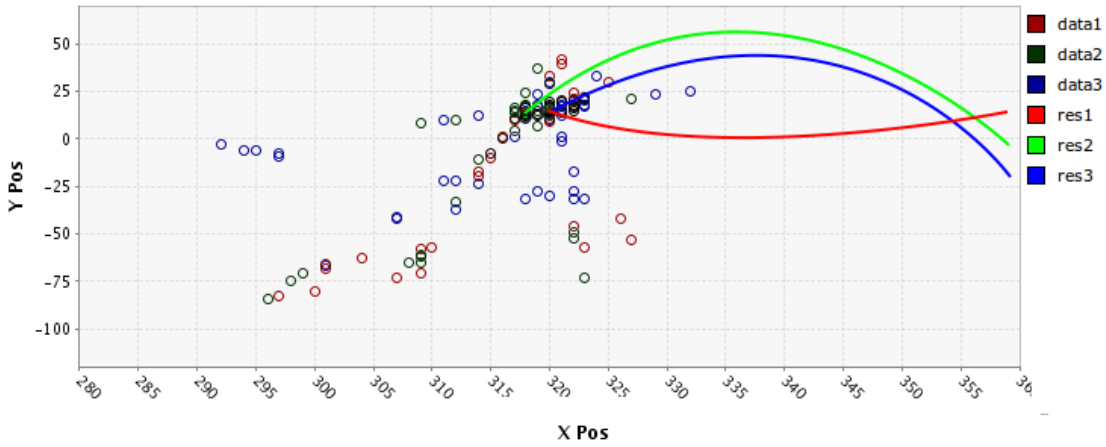


Figure 6.12: Scatter plot of lines' interception points.

being detected and selecting the most significant ones.

Line detection under heavy traffic conditions is more difficult because some of the best indicators for line detection – road marks – become occluded. However, the algorithm was able to detect enough lines to calculate a reasonable approximation for the vanishing point.

In the presence of heavy traffic, using a higher threshold for line detection improves the results because edges on the cars may be grouped together and considered a line. Increasing the line threshold reduces the possibilities of this to happen. Increasing edge threshold also reduces the number of edges on cars which also reduces the chances of having false lines detected in cars.

It has been noticed that in scenarios in which the vanishing point is not near to the central vertical axis, the farthest lanes are smaller than the closest ones. This is an identified problem in the algorithm. The two initial lines should have a different size, dependent on the angle that the vanishing direction makes with a horizontal line. However, this correction could not be implemented during the development period.

Another undesired result which is quite difficult to solve is that the most distant cars seem stretched along the image's vertical axis. This is directly related with cars' height, as described earlier in the section 6.2 and there is no possible solution for this problem. In fact, the measured results present in table 6.7 show that moving object's size difference, when comparing the size in the top of the image and in the bottom, is smaller in the corrected images than in the original. This means that one object moving from a distant point to the camera (or vice-versa) remains more similar in size in the corrected video than in the normal one.

The two problems described here can be reduced if the algorithms use information about the original frame and the corrected one, as one favours cars at a short distance and the other favours the most distant cars.

Object	Corrected projection	Normal projection
Third lane's width	53px	43px
	54px	143px
difference	1	100
First lane's width (left-most)	53px	94px
	50px	48px
difference	3	46
Last lane's width (right-most)	46px	30px
	53px	63px
difference	7	33
Car's vertical span length	36px	84px
	78px	21px
difference	42	63
Van's vertical span length	42px	107px
	96px	27px
difference	54	80

Table 6.7: Projection correction evaluation

6.2 Profiling

This project is intended to work with real-time traffic videos. This way, the processing time of each module is an important characteristic and it must be reasoned whether the benefits from using one particular function are worth the time spent processing it.

This way, this section presents an empirical study of the temporal complexity of the many modules implemented in this project. When the processing time is correlated with some of the parameters, that relation is also stated. When several parameters are experimented with different results, the parameter values and corresponding processing times are listed in tables. All of the experiments were carried out using videos with 640*480 resolution.

Background estimation is at the same time one of the most interesting implemented modules and the one that takes longer to process. The exact processing time is related with

Parameter	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5
Window Size	200				
Background threshold	2.7				
Standard threshold	2			0.5	3.5
Number of gaussians	5	3	2		
Minimum area	1				
Initial weight	0.05				
Initial Variance	30				
Processor time	80ms	60ms	50ms	150ms	75ms

Table 6.8: Background estimation module's benchmarks

Blank cells have the same value of the first experiment.

the number of Gaussians distributions used to model the background and the standard threshold. It has been noticed that the processing time is also somewhat related to the number of foreground pixels detected which might explain why the standard threshold affects the processing time. Processing times ranged from 50ms to 150ms. Table 6.8 presents the extracted information.

Edge detection's processing time is only affected by the size of the filter. Edge detection can take 7ms, when the filter size is 5 and go up to 14ms, when the filter size is 15.

The projection correction module is also a very interesting module developed in this project. It performs several steps. In this analysis they were studied separately because some of them may be optional after initial calculation.

The first task is edge detection which was analysed two paragraphs earlier. Then, line detection is made. This task has a processing time which depends on the number of lines found, so the parameters for edge detection and its parameters may affect it indirectly. The processing times usually ranged from 17 to 21ms.

The tasks responsible for detecting the vanishing point and calculating the correction matrix were very quick, taking less than 1ms. Correcting one image (applying the correction matrix to one image) in average took 11ms. This was done on a 3-channel image. Doing the same task to a single channel image should take one third of the time. Applying the task to a set of points (coordinates) should be very fast.

Optical flow also extracts important information from the image sequences. It is constituted by two steps: key points detection and comparison of them in two consecutive frames. The first step is the one which takes longer (18ms) but it is also the one that remains constant when other parameters are changed. The second step has different running times which, in some situations, seems to be related with the number of key points and motion vectors detected.

While the window size seems to increase processing time alone (i.e., without necessarily increasing the number of motion vectors detected) the other parameters also increased the number of motion vectors detected, which could explain the increased processing time. Table 6.9 lists the profiling results for the optical flow calculation step.

Morphological operations are a set of useful operations which can boost the quality of other modules. The experiments were conducted using 3-channel images. As each channel is treated individually, if Gray scaled images were used, the times would be one third of the registered values. One parameter for this module is the number of iterations to perform. This value multiplies the processing time by its value when compared to performing one single iteration. For this reason, all the conducted experiments were made using only one iteration.

Morphological operations' running times are relatively fast and they are severally affected by the element's size. Opening and closing images is also slightly faster than the

Parameter	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
Max features	400	100	1000			
nPyramids	5			1	7	
Window size	3					
Min quality	0.01					
Min distance	30.0					
Max iterations	200					475
Max epsilon	0.1					
Processor time	3ms	2ms	3ms	1.5ms	3ms	5ms

Parameter	Exp 1	Exp 7	Exp 8	Exp 9	Exp 10	Exp 11
Max features	400					
nPyramids	5					
Window size	3				7	2
Min quality	0.01					
Min distance	30.0					
Max iterations	200	475	475	475		
Max epsilon	0.1	0.35	0.01	0.86		
Processor time	3ms	4ms	5ms	2ms	7ms	2ms

Table 6.9: Optical flow module's benchmarks

Blank cells have the same value of the first experiment.

other operations. With an element's size of 2, the processing time is 7 or 8ms, but when the size is increased to 6, the processing time increases to 17 or 18ms. Table 6.10 lists the measured processing times.

Contour detection was performed within 2ms. The remaining functions like copying or converting images were very fast (under 1ms). The time used to present windows in the GUI is impossible to calculate because that is made in another thread handled by the OpenCV library and does not permit profiling operations on it, but it is believed to be very fast.

The tests were conducted using a laptop with processor Intel Core 2 Duo T9400, 4GB Ram DDR3 and a NVIDIA GeForce 9600M GT 512Mb GDDR3 VRAM. The experiments (and development) were made in a linux environment, using OpenCV 1.1.0. It is

Element's size	2	3	6	11
Open	6ms	9ms	17ms	31ms
Close	6ms	9ms	17ms	31ms
Gradient	7ms	11ms	18ms	31ms
Tophat	7ms	10ms	18ms	31ms
Blackhat	7ms	11ms	18ms	31ms

Table 6.10: Morphological operations module's benchmarks

Blank cells have the same value of the first experiment.

possible that the same implementation running in a windows environment has better performance as hardware support under windows is generally better than under linux. Also, Intel Integrated Performance Primitives[Coo] were not used which could provide better optimisations, improving processing times.

6.3 Synchronisation

There must be always some type of thread synchronisation or else access violations may cause the program to quit. The next list will state the possible combinations of the parameters, how these render to the synchronisation method, how efficient they are and how they affect the user experience:

useMutex thread synchronisation using mutexes has been considered very slow. It provides a bad user experience and should be avoided.

forceNoQueuedConnections the default alternative to using mutexes is using queued connections. When invoking methods that may require synchronisation, the execution in the caller thread is stopped until the other thread finishes. This option is not very efficient although generally offers a better user experience than mutexes. If this parameter is set to true, the program won't use mutexes neither queued connections. When this happens, some kind of synchronisation must be added. OpenCV windows include synchronisation by which they should be used.

useCvWindows using CV windows is much more efficient but some of the extra utilities of the widgets developed within this project are not available. This widgets are less efficient than OpenCV's windows. For that reason it is only recommended to use it in less demanding applications, with a small number of outputs.

The best results are achieved while not using neither mutexes nor using queued connections, and using OpenCV windows for the UI. If the internal widgets are to be used, not using mutexes and using the subsequent queued connections is the best available option.

6.4 Summary

This chapter started by performing a deep analysis of the module's results, capabilities and weaknesses. They were tested under different scenarios and using different parameter combinations.

It has demonstrated how background estimation module can detect moving objects within the videos and explained why distant objects are more difficult to detect. It also explained how the car's height is related to this problem.

In this chapter, it was explained the several steps performed by the edge detection module. It has analysed relation between the parameters and the trade-off between the robustness of the results and their quality. It also analysed why increasing the filter size has little effect beyond middle sized filters.

Morphological operations module and its several operations were analysed on this chapter. Different parameters, different types of images (coloured, Gray scale and binary images) were used and the different types of use which can be made were explained. It has been established how the gradient operation can be used to perform edge detection, top hat and black hat to enhance and extract features from the images and openings and closings to remove the noise present and improve the quality of binary images.

This chapter also analysed optical flow and has analysed the details of the motion detection and explained how that can be used to estimate the velocity of cars.

The projection correction module was analysed. Several estimators were measured to analyse its performance, with the corrected images having demonstrated very good results when comparing object's dimensions at short and long distances. The stability of the results were also analysed and the initial assumptions were proved to be correct.

Detailed benchmarks of the processing time of each module's methods were presented, considering different parameters, image types and conditions which may affect the processing time. The results demonstrated fast operations in most of the modules but also demonstrated that the use of some modules or some values of parameters must be used with caution.

The effects of the different options for thread synchronisation were analysed and the best possible combinations were recommended either for the developed widgets are to be used or not.

Chapter 7

Conclusions

This project was created to answer the needs of NDrive to automate its traffic evaluation method to address the increasing need of monitoring a higher number of traffic cameras.

A modular application has been developed with a set of tools capable of evaluating videos and helping the evaluation task. It is extremely easy to add new modules or algorithms, and to connect them to the UI using already existent controls.

7.1 Solution Success

Comparing the results with the objectives stated in the section 1.2, it has successfully created a possible new use to the infrastructure of traffic cameras installed in Portugal which would promote it and increase its return to the community.

The application runs without making any assumption about the video it receives although it can benefit from the setup of a small number of parameters. It is capable of learning from examples that could be extracted from different videos at different locations.

The only costs for this system are the computer where it is running and the INTERNET connection which may provide the information. This way, an extremely low cost solution was achieved.

During this project, the main modules for this application were developed. It is capable of detecting moving areas, the motion between consecutive frames, detect and correct the projection in one video and learn the evaluation process from examples. This way it can evaluate the speed and density of the road and from these values estimate the congestion level of the road.

7.2 Final Remarks

A complete solution was developed. It was designed with a modular architecture making it easy to add new features as they are required. There are also widgets that make developing an UI to the algorithms and modules very simple and fast.

The background estimation module is a good indicator of the road's traffic density by maintaining a background estimation and comparing it to the received frame. The areas where the difference is bigger than a threshold, are considered to be moving. If these areas area applied in the original image, it can identify the areas where the moving objects are.

The optical flow module is capable of detecting the motion of key points between two frames. This information can be used to track the movement and can be interpreted as an indicator of the moving objects' speed.

The projection correction module, using the edge detection module, is capable of detecting the vanishing point and correcting the projection with good accuracy.

The morphological operations module offers a wide range of operations which can be applied to images in order to improve them or enhance some particular features.

All these modules can be used in algorithms to perform a sequence of tasks and achieve a final estimation for that road's congestion estimation. Actually, as the modules were developed to be used with any purpose, those could be used in different types of tasks.

7.3 Solution Innovations

This thesis has innovated in several aspects.

The starting point of this project is it self very innovative: using an existing infrastructure – traffic cameras throughout Portugal – as a source of information to traffic monitoring applications. This is expected to help performing another innovative task: traffic route guidance with real-time traffic information.

Throughout this project, a methodology was developed to detect the vanishing point and correct the projection of the video which is robust and reduces the size differences of moving objects between the closest and furthest positions captured in the video.

This project used optical flow to estimate the velocity of cars and the foreground mask detected from the comparison of the current frame and the background estimation to estimate the density of the road. With that information, it evaluates the congestion level without ever analysed specific objects or making assumptions about them. This produced a set of robust set of techniques which are expected to be applied in a variety of scenarios.

Additionally, neural networks were used to learn estimating the velocity, density and congestion level of roads using generic information which can be extracted in any environment with any camera setup.

7.4 Solution Limitations

Obviously in an application developed in such a short time span, there are some limitations.

Although the algorithms and techniques were developed making no assumptions about the scenarios, there was a limited range of videos for experimentation. This way, some particular weather conditions may affect negatively the performance of the application. The application is expected to work on night scenes – although there were no experiments – if the road has some level of illumination and if the car lights does not obfuscate the cameras, making the retrieved frames glared. With fog, the used parameters probably need tuning and for rainy conditions, probably different methods should be applied.

There is an identified problem with the projection correction algorithm related with the angle of the line defined by the vanishing point and the center of the image with a vertical line. The problem is more noticeable as this angle increases. The problems arising from this problem may affect the road density evaluations but are not expected to harm the motion detection algorithms. Nevertheless, using this corrected projection in the most problematic scenes still can improve the quality of the background estimation, specially if the original perspective is also considered. One final remark about this problem is that as most of the cameras are aligned with the direction of traffic, this problem is in most of situations unnoticed.

Background estimation has two issues: first, it has a processing time longer than the desired to allow fast processing of video frames. The other problem is related with the foreground mask which could be optimised. The next section includes suggestions to address this problem.

7.5 Future Work

At the point where this project has reached, the next task is creating a sample database to effectively train the neural networks. This should be made throughout several days in order to catch differences in the traffic arrangement which may be unnoticeable to the human operator and which can remain similar throughout a small number of hours.

After having a good set of labelled examples, these can be used to train the neural networks. There are different possibilities: one network can be trained for a particular camera, which will probably have the best results in that camera but has poor results when extrapolating to other cameras. Another possibility is training the neural network with all the learnt samples. This may be become more difficult for the neural network to model all examples. Intermediary solutions are also possible. In these approaches, a set of cameras with similar characteristics (for example, cameras that are at highways have one neural network and cameras at urban areas have another).

For the training to be effective, some kind of definition must be made in terms of how to evaluate a given image and how to represent the evaluation result. Neural networks have the capacity to learn that the best possible value is either 1 or 5, but these metrics should be used all the time in the same way. It is important that similar conditions in different cameras are labelled with the same result (if they are trained in the same neural network) so the learning algorithms are able to learn from it.

Throughout this project, some notes were taken about future development tips which are stated in the following sections.

General Architecture

Each module may create an output by calling the method `createImageResult()` which is defined in the module class. This method will store internally the created object and returns a pointer to it so it can be normally used. The fact of having it stored internally, will automate several things like the creation of windows for each of those results in the UI. If this application is to be continued, similar procedures could be used in the parameter declaration which would automatically create controls in the UI for them. Other behaviours like this could be made in order the allow a faster development.

Background Estimation

This module uses functions in an obscure part of OpenCV, where the code is not so well documented. In the last stages of this project it was found one function that could optimise the foreground mask with the segmentation image of the input. This function is not used and should be experimented. Another discovery was that there is present one improvement of this algorithm which could be used in a very similar way. These suggestions weren't implemented because they were found too late, when the developing process had already stopped and there was not enough time to implement them.

The time taken by this module to process information is more than the ideal so it should be avoided to use it in every frame. One possible solution is just updating the background estimation once every few seconds and manually finding the foreground mask by thresholding the difference between the current frame and the latest background estimation (without updating it).

Projection Correction

As already stated, this module has an identified problem which could be solved by geometry. However it still achieves good results even in the most unfavourable situations.

New Features

A great tool to help developer analyse what should be done next would be to have a set of functions that could be applied to one of the currently displayed image results. These functions could run morphological operations on the images, apply thresholds, filters, calculate histograms, zoom parts of the image and many others.

Another functionality would be the capacity to store parameter configurations in XML format and loading the desired configuration when the program starts.

References

[Ale97] Luís Filipe Barbosa de Almeida Alexandre. Detecção de movimento e sua aplicação à monitorização de tráfego rodoviário. Master's thesis, Engineering Faculty – University of Porto, September 1997.

[AS07] B. Alefs and D. Schreiber. Accurate speed measurement from vehicle trajectories using adaboost detection and robust template tracking. In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 405–412, Seattle, WA,, September/October 2007.

[ASB08] P. F. Alcantarilla, M. A. Sotelo, and L. M. Bergasa. Automatic daytime road traffic control and monitoring system. In *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on*, pages 944–949, Beijing,, October 2008.

[AZXB07] D. Acunzo, Ying Zhu, Binglong Xie, and G. Baratoff. Context-adaptive approach for vehicle detection under varying lighting conditions. In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 654–660, Seattle, WA,, September/October 2007.

[BB95] S.S. Beauchemin and J.L. Barron. The computation of optical flow, 1995.

[Bov05] Alan C. Bovik. *Handbook of Image and Video Processing*. Academic Press, Inc., 2 edition, 2005.

[Bri] Brisa – Auto-estradas de Portugal S. A. Em Viagem – Brisa Website. http://www.brisa.pt/PresentationLayer/emviagem.aspx?menuid=262\&exmenuid=0 – When visualizing the map, click on "Serviços Brisa" and choose the desired ones. Visited by the last time on the 3rd June 2009.

[BST99] Alex Berson, Stephen Smith, and Kurt Thearling. *Building Data Mining Applications for CRM*. McGraw-Hill Companies, 1999.

[Can86] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, November 1986.

[CD07] Pádraig Cunningham and Sarah Jane Delany. k-Nearest Neighbour Classifiers. Technical report, UCD School of Computer Science and Informatics, 2007.

[CHFH07] Yi-Ming Chan, Shih-Shinh Huang, Li-Chen Fu, and Pei-Yung Hsiao. Vehicle detection under various lighting conditions by incorporating particle filter. In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 534–539, Seattle, WA,, September/October 2007.

[Coo] Intel Coorporation. Intel® software network. http://software.intel.com/en-us/intel-ipp/, visited by the last time on the 27th June 2009.

[Cor95] Miguel Fernando Paiva Velhote Correia. Análise do movimento em sequencias de imagens. Master's thesis, Engineering Faculty – University of Porto, 1995.

[CSY07] Jae-Young Choi, Kyung-Sang Sung, and Young-Kyu Yang. Multiple vehicles detection and tracking based on scale-invariant feature transform. In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 528–533, Seattle, WA,, September/October 2007.

[Câ] Câmara Municipal do Porto. Mapa Interactivo da Cidade do Porto. http://194.79.88.139:8081/mapainteractivo/viewer.htm?transito=true, visited by the last time on the 1st June 2009.

[DPP97] Pedro Domingos, Michael Pazzani, and Gregory Provan. On the optimality of the simple bayesian classifier under zero-one loss. In *Machine Learning*, pages 103–130, 1997.

[Ede] Herbert A. Edelstein. Building Profitable Customer Relationships With Data Mining.

[Ede99] Herbert A. Edelstein. *Introduction to Data Mining and Knowledge Discovery, Third Edition*. Two Crows Corporation, 1999.

[Est] Estradas de Portugal. Estradas de Portugal – Trânsito em Directo. http://www.estradasdeportugal.pt, visited by the last time on the 28th June 2009.

[Fed09] Federal Highway Administration. March 2009 traffic volume trends, March 2009. www.fhwa.dot.gov/ohim/tvtw/tvtpage.cfm, visited by the last time on the 5th May 2009.

[FPSS96] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17:37–54, 1996.

[Gur97] Kevin Gurney. *An Introduction to Neural Networks*. CRC, 1997.

[Jam07] James L. Williams. Oil Price History and Analysis, 2007. http://www.wtrg.com/prices.htm, visited by the last time on the 5th May 2009.

[KB01] P. Kaewtrakulpong and R. Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In *Proceedings of 2nd European Workshop on Advanced Video Based Surveillance Systems*, volume 5308, 2001.

[KGT07] R. Khoshabeh, T. Gandhi, and M. M. Trivedi. Multi-camera based traffic flow characterization & classification. In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 259–264, Seattle, WA,, September/October 2007.

[Lea] Learning Artificial Neural Networks. Neural Networks - Training Models and algorithms. "http://www.learnartificialneuralnetworks.com", visited by the last time on the 5th June 2009.

[LHGT03] Liyuan Li, Weimin Huang, Irene Y. H. Gu, and Qi Tian. Foreground object detection from videos containing complex background. In *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, pages 2–10, New York, NY, USA, 2003. ACM.

[LRB09] P. Loureiro, R. Rossetti, and R. Braga. Video processing techniques for traffic-related information extraction using uncontrolled data sources. In *Intelligent Transportation Systems Conference, 2009. ITSC 2009. IEEE*, 2009.

[Lus] Lusoponte. Lusoponte. http://www.lusoponte.pt/livecam.asp, visited by the last time on the 3rd June 2009.

[Lut] Lutz Prechelt. Neural Network FAQ. ftp://ftp.sas.com/pub/neural/FAQ7.html#A_un, visited by the last time on the 5th June 2009.

[Mat02] Matteo Matteucci. ELeaRNT: Evolutionary Learning of Rich Neural Network Topologies. Technical Report CMU-CALD-02-103, Center for Automated Learning and Discovery School of Computer Science, Carnegie Mellon University, September 2002.

[Moo] Andrew W. Moore. Statistical data mining tutorials. http://www.autonlab.org/tutorials/, visited by the last time on the 5th June 2009.

[NDra] NDrive Navigation Systems. NDrive FAQ. http://www.ndriveweb.com/faqs/faq-index/, visited by the last time on the 8th June 2009.

[NDrb] NDrive Navigation Systems. NDrive features. http://www.ndriveweb.com/features/, visited by the last time on the 8th June 2009.

[NDrc] NDrive Navigation Systems. NDrive Navigation Systems. http://www.ndriveweb.com/, visited by the last time on the 29th April 2009.

[NDrd] NDrive Navigation Systems. *NDrive User Guide*. NDrive Navigation Systems. http://www.ndriveweb.com/downloads/file/52/, visited by the last time on the 8th June 2009.

[oT] Illinois Department of Transportation. GCM Travel Site. http://www.gcmtravel.com, visited by the last time on the 5th May 2009.

[oT97] Minnesota Department of Transportation. Non-Intrusive Traffic Detection. Technical report, United States Department of Transportation Federal Highway Administration, May 1997.

[Pic04] M. Piccardi. Background subtraction techniques: a review. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, volume 4, 2004.

[Por05] Fatih Porikli. Multiplicative background-foreground estimation under uncontrolled illumination using intrinsic images. In *in Proc. of IEEE Motion Multi-Workshop*, 2005.

[PT03] Fatih Porikli and Oncel Tuzel. Human body tracking by adaptive background models and mean-shift analysis. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, March 2003.

[SK04] Sen and Chandrika Kamath. Robust techniques for background subtraction in urban traffic video. In Sethuraman Panchanathan and Bhaskaran Vasudev, editors, *Visual Communications and Image Processing 2004*, volume 5308, pages 881–892. SPIE, 2004.

[Sks01] Sherry L. Skszek. "State-of-the-Art" Report On Non-Traditional Traffic Counting Methods. Technical report, Arizona Department of Transportation, 2001.

[SL07] David Schrank and Tim Lomax. 2007 Annual Urban Mobility Report, September 2007. http://mobility.tamu.edu/, visited by the last time on the 6th May 2009.

[Smi97] S. M. Smith. Reviews of optic flow, motion segmentation, edge finding and corner finding. Technical Report TR97SMS1, Oxford University, 1997.

[Sof] Triangle Software. Triangle Software. http://www.trianglesoftware.com/, visited by the last time on the 5th June 2009.

[tT] Beat the Traffic. BeatTheTraffic.com: The Right Traffic at the Right Time. http://beatthetraffic.com, visited by the last time on the 5th June 2009.

[Vis] Visual Numerics, Inc. *IMSL C Stat Library*. http://www.vni.com/products/imsl/documentation/CNL700_Docs/html/cstat/default.htm?turl=naivebayesanoverview.htm, visited by the last time on the 5th June 2009.

[VTBM08] M. Vargas, S. L. Toral, F. Barrero, and J. M. Milla. An enhanced background estimation algorithm for vehicle detection in urban traffic video. In *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on*, pages 784–790, Beijing,, October 2008.

[Wika] Wikipedia. Beat the Traffic. http://en.wikipedia.org/wiki/Beat_The_Traffic, visited by the last time on the 5th June 2009.

[Wikb] Wikipedia. Overfitting. "http://en.wikipedia.org/wiki/Overfitting", visited by the last time on the 5th June 2009.

[WL08] Chris Williams and Victor Lavrenko. Nearest neighbour method , 2008. School of Informatics - University of Edinburgh.

[WYQ+07] Kunfeng Wang, Qingming Yao, Xin Qiao, Shuming Tang, and Fei-Yue Wang. Moving object refining in traffic monitoring applications. In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 540–545, Seattle, WA,, September/October 2007.

[YMW+07] Zuguang Yang, Huadong Meng, Yimin Wei, Hao Zhang, and Xiqin Wang. Tracking ground vehicles in heavy-traffic video by grouping tracks of vehicle corners. In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 396–399, Seattle, WA,, September/October 2007.

[YZMW07] Ming Yin, Hao Zhang, Huadong Meng, and Xiqin Wang. An HMM-based algorithm for vehicle detection in congested traffic situations. In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 736–741, Seattle, WA,, September/October 2007.

Appendix A

Sample Videos Description

This attachment has a description of each video that was used throughout the project to experiment the created algorithms and modules. This will enable the user to better understand the conditions under which algorithms were executed.

When left or right lanes are referred, these are in terms of the image it self. This means that in normal circumstances, in a two way road, the left lane is heading into the camera and the right lane is heading away from the camera.

Figure A.1: Sample frame from video 1279

Identification 1279

Duration 5 minutes 10 seconds

Frames per second 30

Resolution 640*480

Visualisation On this video, 3 lanes in each direction are observed. Camera is pointing to a far away point.

Conditions In general, there is a fluid traffic, with good travelling speed but the far most left lane is busy, having cars stopped for some seconds each time.

Figure A.2: Sample frame from video 1292

Identification 1292

Duration 2 minutes 6 seconds

Frames per second 30

Resolution 640*480

Visualisation On this video, 3 lanes with cars moving away from the camera are observed. Camera is pointing to a point at a medium distance.

Conditions The traffic is dense although it is able to keep good velocities at some times. Velocities range from slow to medium at different times of the video. The right line is clearly more slow and dense than the other two lanes.

Identification 1293

Duration 2 minutes 15 seconds

Frames per second 30

Resolution 640*480

Visualisation On this video, 3 lanes with cars moving into the camera are observed. Camera is pointing to a point at a medium distance.

Conditions The traffic is dense although it is able to keep good velocities.

Figure A.3: Sample frame from video 1293

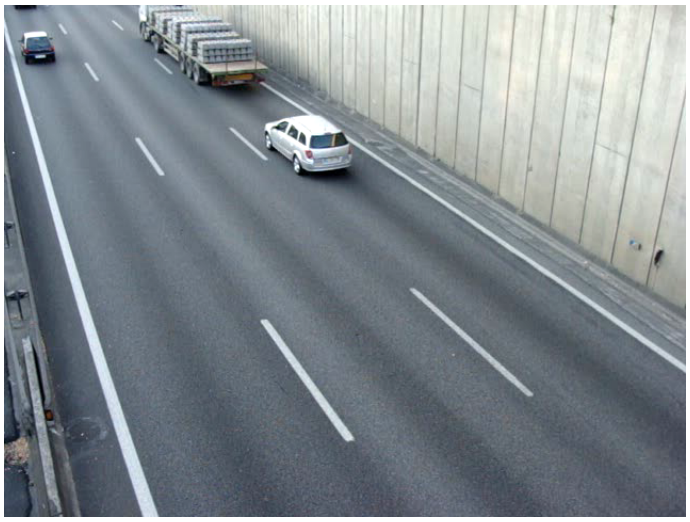


Figure A.4: Sample frame from video 1322

Identification 1322

Duration 3 minutes 31 seconds

Frames per second 30

Resolution 640*480

Visualisation On this video, 3 lanes with cars moving away from the camera are observed. Camera is pointing to a point at a close distance and laterally to the direction of movement.

Conditions The traffic is sparse, moving at a good velocity.

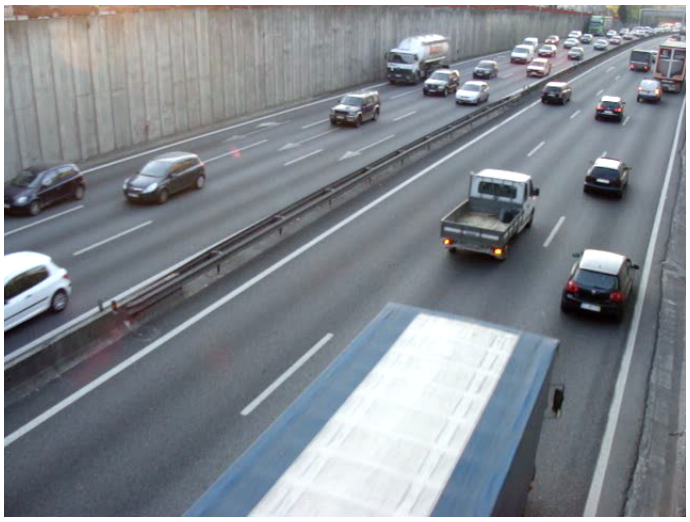


Figure A.5: Sample frame from video 1346

Identification 1346

Duration 2 minutes 18 seconds

Frames per second 30

Resolution 640*480

Visualisation On this video, 3 lanes in each direction are observed. Camera is pointing to a point at a close distance and laterally to the direction of movement.

Conditions The traffic is sparse, moving at a good velocity.

Figure A.6: Sample frame from video 1360

Identification 1360

Duration 1 minutes 12 seconds

Frames per second 30

Resolution 640*480

Visualisation On this video, 3 lanes in each direction are observed. Camera is pointing to a point at a close distance and laterally to the direction of movement.

Conditions This video starts by having a medium density level and good velocity. As the video advances, the direction that is moving away from the camera, gets more congested and slow. In the last seconds of video, some cars have stopped in the two right-most lanes.

Figure A.7: Sample frame from video 1385

Identification 1385

Duration 2 minutes 30 seconds

Frames per second 30

Resolution 640*480

Visualisation On this video, 3 lanes with cars moving into the camera are observed. Camera is pointing to a point at a medium distance, laterally to the direction of movement.

Conditions This video has dense traffic, moving at slow velocities. Some cars some their movement during the video.

Identification 1387

Figure A.8: Sample frame from video 1387

Duration 2 minutes

Frames per second 30

Resolution 640*480

Visualisation On this video, 3 lanes in each direction and some additional access ways are observed. Camera is pointing to a point at a long distance.

Conditions This video has traffic with medium density, moving at good velocities. Lanes heading to the camera are more slow and dense than the others.

Figure A.9: Sample frame from video 1388

Identification 1388

Duration 2 minutes 2 seconds

Frames per second 30

Resolution 640*480

Visualisation On this video, 3 lanes in each direction and are observed. Camera is pointing to a point at a close distance.

Conditions This video has traffic with medium density, moving at good velocities.

Figure A.10: Sample frame from video 1389

Identification 1389

Duration 1 minutes 27 seconds

Frames per second 30

Resolution 640*480

Visualisation On this video, 3 lanes in each direction and are observed. Camera is pointing to a point at a very long distance.

Conditions This video has traffic with high density, moving at medium velocities. The traffic coming to into the camera is more dense and slow than the other. Occlusion is very severe on this video.

Identification 1390

Duration 2 minutes

Frames per second 30

Figure A.11: Sample frame from video 1390

Resolution 640*480

Visualisation On this video, 3 lanes in each direction and are observed. Camera is pointing to a point at a very long distance.

Conditions This video has traffic with high density, moving at slow velocities. The traffic moving away from the camera is more less dense and has better velocities than the other. Occlusion is very severe on this video.

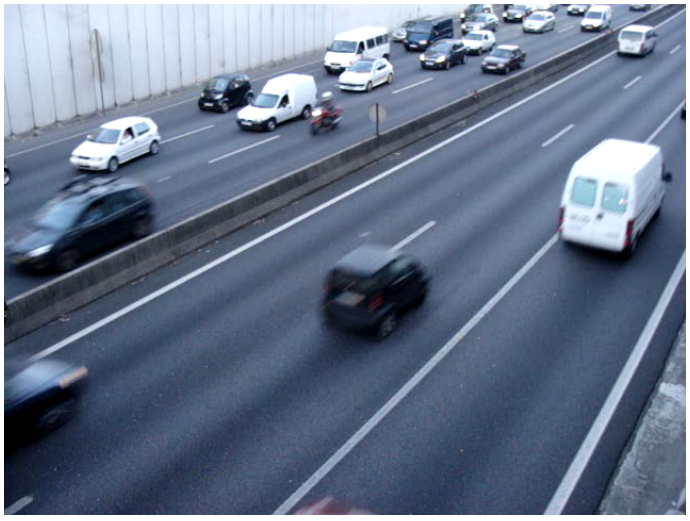


Figure A.12: Sample frame from video 1427

Identification 1427

Duration 2 minutes 16 seconds

Frames per second 30

Resolution 640*480

Visualisation On this video, 3 lanes in each direction and are observed. Camera is pointing to a point at a close distance, laterally to the direction of movement.

Conditions This video has medium density with good velocity. The fact that six lanes are shown laterally, the farther lanes appear significantly smaller than the most close lanes.

Identification 1428

Duration 2 minutes 19 seconds

Frames per second 30

Resolution 640*480

Visualisation On this video, 3 lanes in each direction and are observed. Camera is pointing to a point at a close distance, laterally to the direction of movement.

Conditions This video has high density with good velocity. The left-most lane is completely saturated, travelling at a very slow velocity.

Figure A.13: Sample frame from video 1428