FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Title of the Dissertation

João Neto

WORKING VERSION

U. PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Name of the Supervisor

June 27, 2017

Title of the Dissertation

João Neto

Mestrado Integrado em Engenharia Informática e Computação

June 27, 2017

Abstract

Here goes the abstract written in English.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed vehicula lorem commodo dui. Fusce mollis feugiat elit. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec eu quam. Aenean consectetuer odio quis nisi. Fusce molestie metus sed neque. Praesent nulla. Donec quis urna. Pellentesque hendrerit vulputate nunc. Donec id eros et leo ullamcorper placerat. Curabitur aliquam tellus et diam.

Ut tortor. Morbi eget elit. Maecenas nec risus. Sed ultricies. Sed scelerisque libero faucibus sem. Nullam molestie leo quis tellus. Donec ipsum. Nulla lobortis purus pharetra turpis. Nulla laoreet, arcu nec hendrerit vulputate, tortor elit eleifend turpis, et aliquam leo metus in dolor. Praesent sed nulla. Mauris ac augue. Cras ac orci. Etiam sed urna eget nulla sodales venenatis. Donec faucibus ante eget dui. Nam magna. Suspendisse sollicitudin est et mi.

Fusce sed ipsum vel velit imperdiet dictum. Sed nisi purus, dapibus ut, iaculis ac, placerat id, purus. Integer aliquet elementum libero. Phasellus facilisis leo eget elit. Nullam nisi magna, ornare at, aliquet et, porta id, odio. Sed volutpat tellus consectetuer ligula. Phasellus turpis augue, malesuada et, placerat fringilla, ornare nec, eros. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Vivamus ornare quam nec sem mattis vulputate. Nullam porta, diam nec porta mollis, orci leo condimentum sapien, quis venenatis mi dolor a metus. Nullam mollis. Aenean metus massa, pellentesque sit amet, sagittis eget, tincidunt in, arcu. Vestibulum porta laoreet tortor. Nullam mollis elit nec justo. In nulla ligula, pellentesque sit amet, consequat sed, faucibus id, velit. Fusce purus. Quisque sagittis urna at quam. Ut eu lacus. Maecenas tortor nibh, ultricies nec, vestibulum varius, egestas id, sapien.

Phasellus ullamcorper justo id risus. Nunc in leo. Mauris auctor lectus vitae est lacinia egestas. Nulla faucibus erat sit amet lectus varius semper. Praesent ultrices vehicula orci. Nam at metus. Aenean eget lorem nec purus feugiat molestie. Phasellus fringilla nulla ac risus. Aliquam elementum aliquam velit. Aenean nunc odio, lobortis id, dictum et, rutrum ac, ipsum.

Ut tortor. Morbi eget elit. Maecenas nec risus. Sed ultricies. Sed scelerisque libero faucibus sem. Nullam molestie leo quis tellus. Donec ipsum. Nulla lobortis purus pharetra turpis. Nulla laoreet, arcu nec hendrerit vulputate, tortor elit eleifend turpis, et aliquam leo metus in dolor. Praesent sed nulla. Mauris ac augue. Cras ac orci. Etiam sed urna eget nulla sodales venenatis. Donec faucibus ante eget dui. Nam magna. Suspendisse sollicitudin est et mi.

Phasellus ullamcorper justo id risus. Nunc in leo. Mauris auctor lectus vitae est lacinia egestas. Nulla faucibus erat sit amet lectus varius semper. Praesent ultrices vehicula orci. Nam at metus. Aenean eget lorem nec purus feugiat molestie. Phasellus fringilla nulla ac risus. Aliquam elementum aliquam velit. Aenean nunc odio, lobortis id, dictum et, rutrum ac, ipsum.

Ut tortor. Morbi eget elit. Maecenas nec risus. Sed ultricies. Sed scelerisque libero faucibus sem. Nullam molestie leo quis tellus. Donec ipsum.

Resumo

O Resumo fornece ao leitor um sumário do conteúdo da dissertação. Deverá ser breve mas conter detalhe suficiente e, uma vez que é a porta de entrada para a dissertação, deverá dar ao leitor uma boa impressão inicial.

Este texto inicial da dissertação é escrito no fim e resume numa página, sem referências externas, o tema e o contexto do trabalho, a motivação e os objectivos, as metodologias e técnicas empregues, os principais resultados alcançados e as conclusões.

Este documento ilustra o formato a usar em dissertações na Faculdade de Engenharia da Universidade do Porto. São dados exemplos de margens, cabeçalhos, títulos, paginação, estilos de índices, etc. São ainda dados exemplos de formatação de citações, figuras e tabelas, equações, referências cruzadas, lista de referências e índices. É usado texto descartável, *Loren Ipsum*, para preencher a dissertação por forma a ilustrar os formatos.

Seguem-se umas notas breves mas muito importantes sobre a versão provisória e a versão final do documento. A versão provisória, depois de verificada pelo orientador e de corrigida em contexto pelo autor, deve ser publicada na página pessoal de cada estudante/dissertação, juntamente com os dois resumos, em português e em inglês; deve manter a marca da água, assim como a numeração de linhas conforme aqui se demonstra.

A versão definitiva, a produzir somente após a defesa, em versão impressa (dois exemplares com capas próprias FEUP) e em versão eletrónica (6 CDs com "rodela" própria FEUP), deve ser limpa da marca de água e da numeração de linhas e deve conter a identificação, na primeira página, dos elementos do júri respetivo. Deve ainda, se for o caso, ser corrigida de acordo com as instruções recebidas dos elementos júri.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed vehicula lorem commodo dui. Fusce mollis feugiat elit. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec eu quam. Aenean consectetuer odio quis nisi. Fusce molestie metus sed neque. Praesent nulla. Donec quis urna. Pellentesque hendrerit vulputate nunc. Donec id eros et leo ullamcorper placerat. Curabitur aliquam tellus et diam.

Ut tortor. Morbi eget elit. Maecenas nec risus. Sed ultricies. Sed scelerisque libero faucibus sem. Nullam molestie leo quis tellus. Donec ipsum. Nulla lobortis purus pharetra turpis. Nulla laoreet, arcu nec hendrerit vulputate, tortor elit eleifend turpis, et aliquam leo metus in dolor. Praesent sed nulla. Mauris ac augue. Cras ac orci. Etiam sed urna eget nulla sodales venenatis. Donec faucibus ante eget dui. Nam magna. Suspendisse sollicitudin est et mi.

Phasellus ullamcorper justo id risus. Nunc in leo. Mauris auctor lectus vitae est lacinia egestas. Nulla faucibus erat sit amet lectus varius semper. Praesent ultrices vehicula orci. Nam at metus. Aenean eget lorem nec purus feugiat molestie. Phasellus fringilla nulla ac risus. Aliquam elementum aliquam velit. Aenean nunc odio, lobortis id, dictum et, rutrum ac, ipsum.

Ut tortor. Morbi eget elit. Maecenas nec risus. Sed ultricies. Sed scelerisque libero faucibus sem. Nullam molestie leo quis tellus. Donec ipsum.

Acknowledgements

Aliquam id dui. Nulla facilisi. Nullam ligula nunc, viverra a, iaculis at, faucibus quis, sapien. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Curabitur magna ligula, ornare luctus, aliquam non, aliquet at, tortor. Donec iaculis nulla sed eros. Sed felis. Nam lobortis libero. Pellentesque odio. Suspendisse potenti. Morbi imperdiet rhoncus magna. Morbi vestibulum interdum turpis. Pellentesque varius. Morbi nulla urna, euismod in, molestie ac, placerat in, orci.

Ut convallis. Suspendisse luctus pharetra sem. Sed sit amet mi in diam luctus suscipit. Nulla facilisi. Integer commodo, turpis et semper auctor, nisl ligula vestibulum erat, sed tempor lacus nibh at turpis. Quisque vestibulum pulvinar justo. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Nam sed tellus vel tortor hendrerit pulvinar. Phasellus eleifend, augue at mattis tincidunt, lorem lorem sodales arcu, id volutpat risus est id neque. Phasellus egestas ante. Nam porttitor justo sit amet urna. Suspendisse ligula nunc, mollis ac, elementum non, venenatis ut, mauris. Mauris augue risus, tempus scelerisque, rutrum quis, hendrerit at, nunc. Nulla posuere porta orci. Nulla dui.

Fusce gravida placerat sem. Aenean ipsum diam, pharetra vitae, ornare et, semper sit amet, nibh. Nam id tellus. Etiam ultrices. Praesent gravida. Aliquam nec sapien. Morbi sagittis vulputate dolor. Donec sapien lorem, laoreet egestas, pellentesque euismod, porta at, sapien. Integer vitae lacus id dui convallis blandit. Mauris non sem. Integer in velit eget lorem scelerisque vehicula. Etiam tincidunt turpis ac nunc. Pellentesque a justo. Mauris faucibus quam id eros. Cras pharetra. Fusce rutrum vulputate lorem. Cras pretium magna in nisl. Integer ornare dui non pede.

The Name of the Author

"You should be glad that bridge fell down.
I was planning to build thirteen more to that same design"

Isambard Kingdom Brunel

Contents

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

2D	Two Dimensions
3D	Three Dimensions
ACM	Association for Computing Machinery
ADT	Abstract Data Type
ANDF	Architecture-Neutral Distribution Format
API	Application Programming Interface
ATON	Autonomous Transportation Agents for On-scene Networked incident manager
BGSub	Background Subtracter
CUDA	Compute Unified Device Architecture
FFmpeg	Fast Forward Moving Pictures Expert Group
HSV	Hue, Saturation and Value
IEEE	Institute of Electrical and Electronics Engineers
ITS	Intelligent Transport Systems
LIACC	Laboratório de Inteligência Artificial e Ciência de Computadores
MIT	Massachusetts Institute of Technology
MRI	Magnetic resonance imaging
OpenCV	Open Source Computer Vision
RGB	Red Green Blue
SAKBOT	Statistical and Knowledge-BasedObject Tracker
YACCLAB	Yet Another Connected Components Labelling Benchmark
CAGR	Compound Annual Growth Rate
OpenCV	Open Source Computer Vision Library

Todo list

ABBREVIATIONS

Chapter 1

Introdução

O primeiro capítulo da dissertação deve servir para apresentar o enquadramento e a motivação do trabalho e para identificar e definir os problemas que a dissertação aborda. Deve resumir as metodologias utilizadas no trabalho e termina apresentando um breve resumo de cada um dos capítulos posteriores.

1.1 Contexto/Enquadramento

Esta secção descreve a área em que o trabalho se insere, podendo referir um eventual projeto de que faz parte e apresentar uma breve descrição da empresa onde o trabalho decorreu.

Traffic management in cities is a vast area as it studies the planning and control of the road network. As the cities tend to evolve following the concept of *smart cities*, this is also a field where the information technologies are being applied to improve the effectiveness of these tasks.

With the increase in the number of vehicles using the roads [Nav00], a need to improve the methods of traffic management rose as well. The development of multiple approaches to this issue represents both it's importance and the pertinence of the work being developed at LIACC in regards to this topic.

Evolving the existing solutions involves a process of reformulation of the methods being currently used. With the decreasing costs of cameras for video surveillance, the number of units installed around the world is rising, passing the 245 million mark in 2014 [Jen15], over 65% of that number being from Asia.

With this number of cameras placed globally the amount of data being collected every day is too large to be humanly processed, and thus the need to create autonomous processors arises. The market for automatic analysis of video is expected to be worth 11.17 Billion USD by 2022 [Rep17], with the facial recognition area expected to have the highest CAGR (Compound Annual Growth Rate) due to the potential related to the security applications, even going as far as replacing airport checks in Australia by 2020 [Koz17].

The usage of computer vision to analyse traffic conditions has been around for some time now, with earlier works dating to 1990, such as the work by Rourke et. al [RBH90] where the advantages of using video analysis in this area are listed, as well as some issues that the community is still facing. Since then the field has been explored by many authors, applied to applications ranging from vehicle counting [CBMM98] [BMCM97] to incidents detection.

Armis-Group is a company based in Porto that builds software in the areas of Information Management, Digital Sports and Intelligent Transport Systems.

1.2 Project

Na continuação da secção anterior, e apenas no caso de ser um Projeto e não uma Dissertação, esta secção apresenta resumidamente o projeto.

The project for this dissertation was part of a collaboration between LIACC and Armis-Group. This collaboration aims to develop solutions in the field of Traffic Management, taking advantage of the laboratory's resources and the company's expertise and experience in the area. The objective of the project was to design and develop a software module capable of analysing video and extracting relevant events on the roads as well as performing counting and basic classification of vehicles.

In the scope of the mentioned collaboration, LIACC developed a set of applications that form the "Video Server", a software package that

1.3 Motivation and Goals

Apresenta a motivação e enumera os objetivos do trabalho terminando com um resumo das metodologias para a prossecução dos objetivos.

The main goal of the dissertation is then to implement a solution that fits the needs of the project as stated above, detecting events in a short window of time to allow for fast response from the authorities, extracting

This goal can be broke down into more specific tasks to allow for a better understanding of the work ahead:

- Explore state of the art techniques that address the multiple obstacles to be faced along the process;
- Work in close relationship with the collaborators of the project;
- Implement solutions to perform the following tasks:

  Clear the image of noise and other unwanted features;

  Find objects of interest;

  Classify detected objects;

  Detect relevant events.

- Testing of the implementation in a real scenario.

1.4 Estrutura da Dissertação

Chapter 2

Computer Vision and Traffic Analysis - A literature review

2.1 History of Computer Vision

Computer vision appeared as an area of investigation around the mid 60's at the MIT by Professor Larry Roberts, whose doctorate thesis focused on methods to extract 3D information from a 2D image [Hua96] in order to reconstruct entire scenes from the geometry gathered. This area is now considered by the ACM as a branch of artificial intelligence according to the 2012 ACM Computing Classification System [ACM12]. From this point onward scientists began applying the techniques developed in multiple fields.

The use of computer vision in manufacturing industry was among the first to be explored, as there were already multiple robots working in the assembly lines of the factories that needed to be improved, as noted by Stout in 1980 [Sto80]. These robots were becoming outdated due to the lack of interaction with the environment. This meant that for a robot to work with a part in an assembly line it needed to be placed in a pre-determined position, and any deviation could mean that the line needed to be stopped. Michael Baird relates in [LB78] a computer vision system developed to inspect circuit chips rotation deviation in a welding base, indicating to the welder that the chip needed to be adjusted. Computer vision was also used as a tool to automatically analyse weather satellite images [BT73] and multidimensional medical images [Aya98], with applications being developed in these areas being now common in our daily life.

2.1.1 Computer Vision in Intelligent Transport Systems

Regarding intelligent transport systems (ITSs), the use of computer vision to aid in the analysis of traffic has been increasing in the last years due to the decreasing costs of hardware, both cameras, storage and processing power, as well as the growing knowledge to extract useful data from the video gathered. In contrast to the high installation and upkeep costs of other traffic control tools such as Inductive Loop Detectors and Microwave Vehicle Detectors, applying computer vision to handle these tasks is a profitable option for entities in charge of the analysis of this data.

One of the first works applying computer vision to analyse traffic was published in 1984 [DW84] and detected and measured movement in a sequence of frames. Since then, multiple fields of study have been created, not only for the measuring of traffic, as explored in [LKR^{+}16], [?] and [HK12] where the authors evaluate methods to analyse traffic in urban environments, but also for the analysis of the environment around a self-driven vehicle, reading traffic signs using multiple approaches using convolutional neural networks [SS11] or using bag-of-visual-words [SLZ16], or to automatize the parking process as discussed in [HBJ^{+}16]. Recently some studies have surfaced where the authors discuss the analysis of passenger numbers and behaviour inside vehicles, in order to enforce traffic laws [LBT17].

For our application however we need to focus on the aspect of traffic analysis, and in order to understand what traffic analysis software needs to accomplish, it is convenient to study what composes a typical traffic scene. Usually the scenes are viewed from a top-down perspective that places the cars against the road as background, as seen in 2.1. The vehicles have a roughly regular rectangular shape when viewed from the top, with little variation when the camera is rotated, however their textures vary heavily, making it difficult for a detector to work based on the vehicles image representation [BP98]. However, depending on the camera position there can be object occlusion by other objects or scene components, which makes it difficult to detect and track vehicles relying solely on subtraction based segmentation techniques.

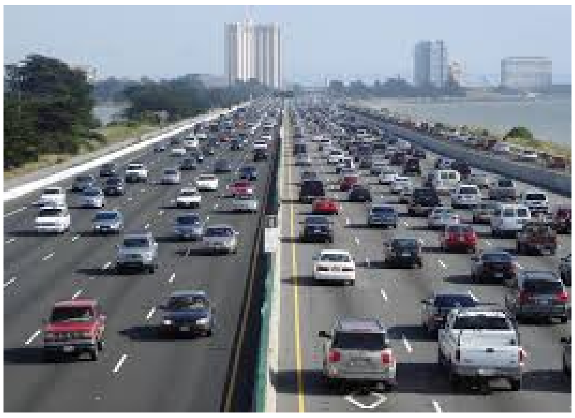


Figure 2.1: Typical High-Way Traffic Scene

The scenes also have varying light according to the time of the day, and proposed solutions need to adapt to these changes as fast as possible, otherwise data might be lost. Other weather conditions can also interfere with the analysis, such as fog and rain, and need to be addressed when designing solutions.

2.1.2 Our Challenge

The "Analytics Server" module of the "Video Server" project had a list of requirements that needed to be addressed:

- Detect relevant events:

Wrong way driving;

Fallen objects on the road;

Prohibited zone entering;

Suspicious camera approximation

- Count and classify vehicles into light and heavy categories

The following sections of this document will explore the theoretical basis on which our solution was built, aimed at tackling these specific points.

2.2 Image Treatment

This section will review some of the proposed methods to improve image quality before or during processing by removing noise, extracting unwanted features or enhancing relevant features for the processes involved.

2.2.1 Blur

In order to blur an image one needs to convolve it using a special kind of filter. The process of convolution in image processing consists in the application of a filter to every pixel of an image, returning a new image where each pixel is the result of the function to the pixel in the same position in the previous image. As we can see in 2.2 the application of a 3*3 mask, where all 9 values have the value 1/9, results in a image where each pixel value equals the mean of itself and surrounding neighbours in the original image.

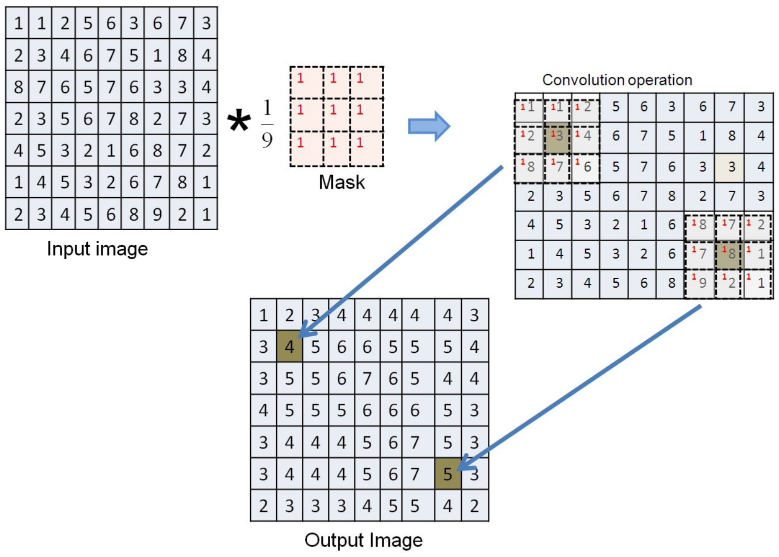


Figure 2.2: Convolution Example [Lab17]

While this is the simplest blur processing we can apply to an image, there are more complex ones that yield better results. The one most often used in image processing is the Gaussian Blur, which uses a square filter with values calculated with the equation in 2.1 from [SS01].

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.1}$$

This equation returns a filter consisting of concentric circles that results in a smooth blurring of the images due to the application of the same power of blurring from equidistant pixels. The function can be tuned to the needs of the process through the σ value, returning a sharper or softer blur filter as seen in figure 2.3.

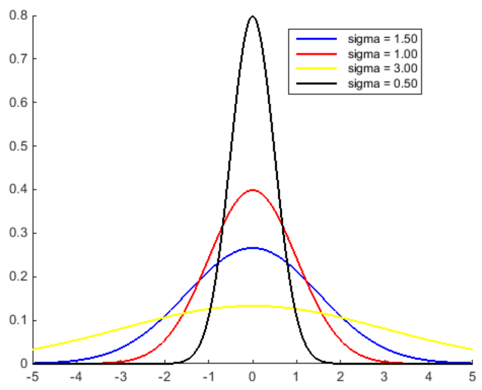


Figure 2.3: Gaussian Function Variation with Sigma Value

This kind of filters have multiple uses in computer vision. Blurring an image removes the grain noise it contains, creating more even surfaces that ease the process of image processing, as well as making edges easier to detect with edge detection algorithms.

2.2.2 Mathematical Morphology

Mathematical Morphology is a set theory, offering solutions to multiple image processing problems, where the sets are composed of tuples corresponding to the 2D coordinates of each pixel in the image. These sets identify the white or the black pixels in a binary image, depending on the convention adopted, or the intensity of the pixel in a grey-scale image [Dou92]. During this chapter we will consider that all images are binary and that the pixels to process are black in a white background. This theory provides a set of operators that can be used to detect convexity and connectivity of shapes in an image, enhancing features, detecting edges and removing noise.

The operators are defined by a structuring element, also known as kernel, a set of coordinates represented in a binary image, usually much smaller than the image to process. The centre of the kernel is not in the top left corner as in most images but rather in the geometrical centre, meaning that there some of the pixels have negative coordinates. The kernel shapes in figure 2.4 are the ones typically used for most applications, but the process allows the user to design their own if needed.

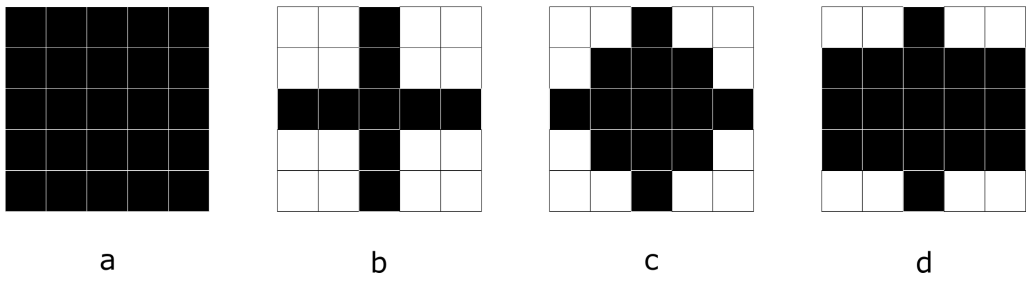


Figure 2.4: Morphology Operators typical structuring elements (5x5)
a) Rectangular b) Cross c) Diamond d) Elliptical

According to Gonzalez and Woods [GW92] an operator is defined by an image and a structuring element, and to run it the origin point of the element needs to be placed at every pixel of the image. At each of these locations a verification is done depending on the operation being performed and an output image is created using the independent output of these operations.

The two most basic operations are erosion and dilation, essential to morphological processing. The erosion operator translates the structuring element to every black pixel of the image and in each position checks if all the active pixels of the kernel match with black pixels in the image. If it is true then the returned pixel is black, otherwise it is white. An example of the usage of such an operator can be seen in figure 2.5. These operators are mainly used to remove noise from binary images or to separate objects to enable individual labelling and counting.

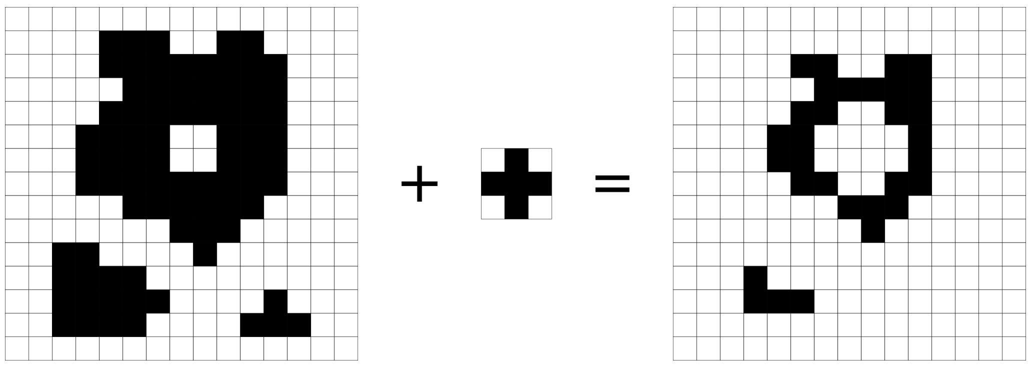


Figure 2.5: Result of the Erosion operator using a cross 3x3 kernel

The dilation operator works by setting to black all the pixels that match the position of the translated kernel's pixels, resulting in the closing of holes in the image and the enhancing of small features. Results can be seen in figure 2.6. Using a non-symmetrical kernel the dilation will be directional, enabling the user to fine tune the process to their needs.

The dilation and erosion operators can be combined to form the opening and closing operators. An opening operator is formed by an erosion followed by a dilations using the same structuring

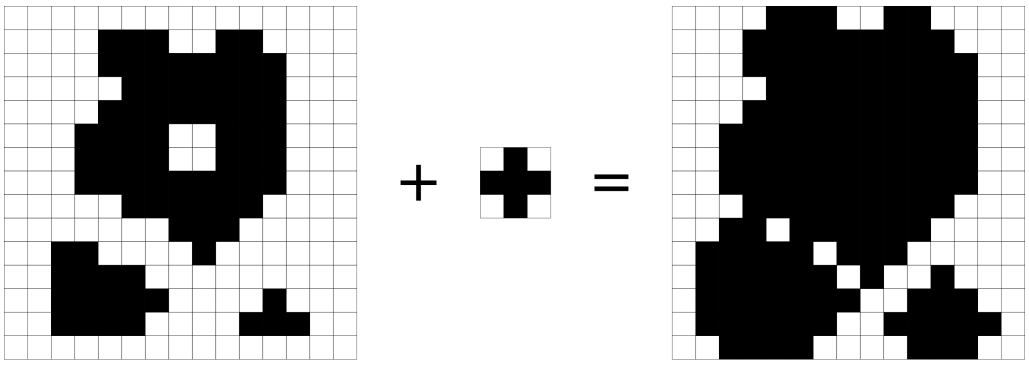


Figure 2.6: Result of the Dilation operator using a cross 3x3 kernel

element. This results in a less destructive erosion, preserving some of the image details as seen in figure 2.7. The closing operator is formed by a dilation followed by an erosion and the results can be seen in figure 2.8. It is commonly used to remove salt noise and to extract objects of a particular shape as long as they all share the same orientation [FPWW03].

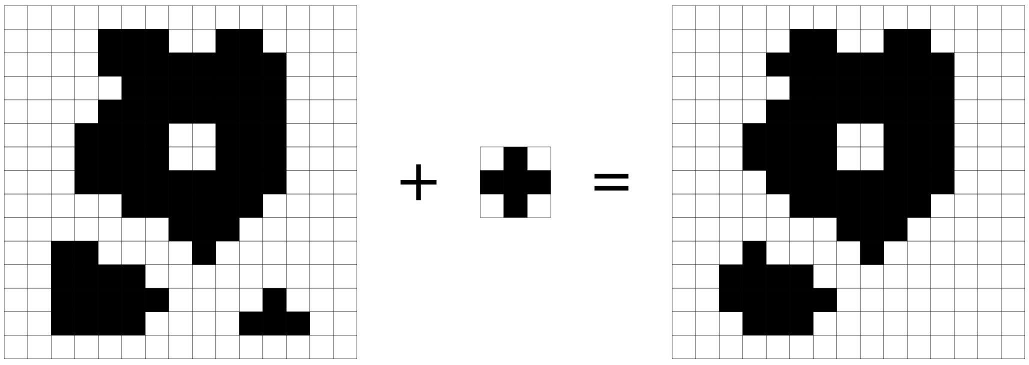


Figure 2.7: Result of the Opening operator using a cross 3x3 kernel

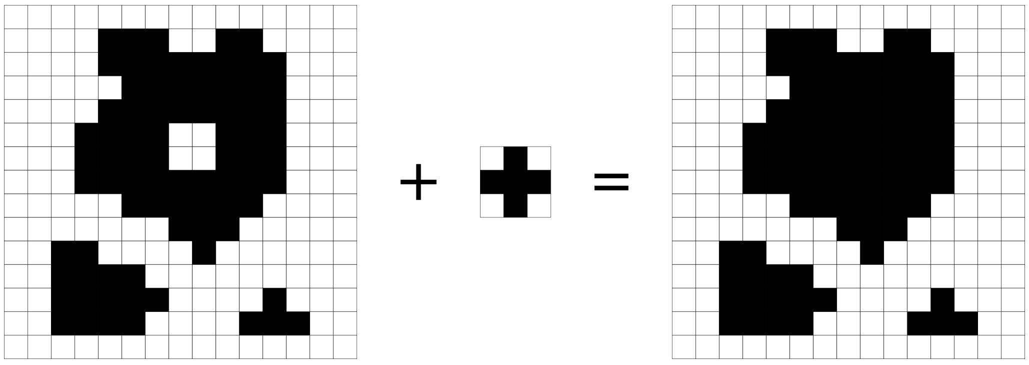


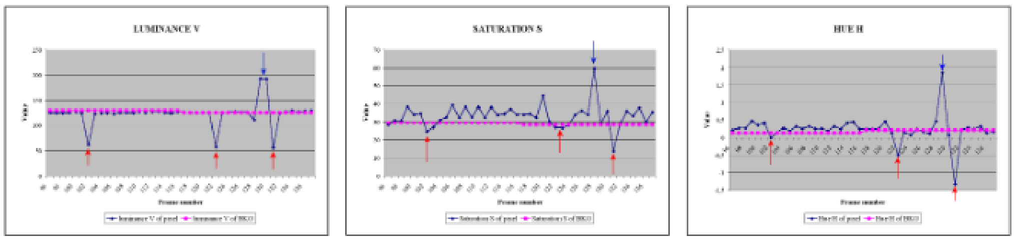
Figure 2.8: Result of the Closing operator using a cross 3x3 kernel

2.2.3 Shadow Removal

Since shadows can be detected as foreground during the segmentation process, a process to remove them is important in certain scenarios. We will now describe some of the more interesting proposed methods to perform this task in this section.

In [PMGT01] Prati et al. discussed how critical the shadow removal process is to traffic analysis, and they proceed to compare two ITS management systems, SAKBOT (Statistical and Knowledge-BasedObject Tracker) and ATON(Autonomous Transportation Agents for On-scene Networked incident manager), developed in Italy and the United States of America respectively.

SAKBOT adopts a method presented in [CGPP00], which uses the chrominance information, converting pixel colours from RGB to HSV space, as it most closely relates to the human perception of colour. The process then evaluates how the scene colours components change due to the passing of vehicles and shadows, as seen in figure 2.9. From this we can extract that a shadow darkens a pixel and saturates its colour, and is the difference between an object point and a shadow point. The model then concludes that a point is considered as shadow if:



(a) Luminance (b) Saturation (c) Hue

Figure 2.9: HSV color space components change due to shadows. Red arrows indicate a shadow passing on the point. Blue arrow indicates a vehicle passing [PMGT01] ©[2001] IEEE

- The ratio between the luminance of the image and the background is between two thresholds (α and β)
- The difference between the saturation of the image and the background is below a threshold (τ_S)
- The absolute difference between the hue of the image and the background is below a threshold (τ_H)

The β value adjusts the tolerance to noise in the image to be considered as shadow and α takes into account the "power" of the shadow, how strong the light source is. τ_S and τ_H are considered by the authors to be harder to assign, and thus assigned empirically. This method main advantage is it's capability to detect moving shadows.

The method used by ATON is described in [MCKT00] and begins by identifying three separate sources of information to detect shadows and objects:

- Local information, the appearance of an individual pixel
- Spatial information, objects and shadows are compact regions
- Temporal information, the location of shadows and objects can be predicted from previous frames

The local information can be used to create a background model, both the mean and variance of the colour for each pixel when shadowed and not shadowed. In the segmentation process the values in the image are compared to the mean value of the background and if significantly different it is assigned a probability to belong to the background, foreground or shadow. The probability values are 0.3, 0.4 and 0.4 respectively. In the next iterations the neighbour pixels probabilities are considered in order to take advantage of the spatial transformation. The iterative process stops when there are no relevant changes, usually after the third iteration. Morphological operators are then computed over the output of the process to close unwanted openings in the regions. ATON does not use temporal information as of the time of the publication of the work. This process can identify with near 90% accuracy shadows and object accuracy.

2.3 Foreground Segmentation

Image segmentation is the process through which an image is separated into its different regions according to the desired output, usually objects of interest. These regions are composed by pixels that have a common characteristic [SS01] dependant on the desired result. There are several ways one can accomplish an adequate segmentation of an image, and the most relevant ones to our purpose are described below.

2.3.1 Background Subtraction

Background Subtraction is a segmentation technique based on the analysis of the difference of consecutive frames and use that information to create a background model, a representation of what the image looks like without any moving objects present. Given the nature of the process, cameras must be static and although some research has been made to overcome this issue [LCC12] [SJK09] [ZY14], this is beyond the scope of the project, due to the requirements given.

There are however a number of different ways to implement the background subtraction developed across the years with increasing segmentation accuracy and computational performance. Piccardi presented an overview of the most relevant ones in [Pic04], aiming to present each method's strengths and weaknesses. From this work we can present a list of the algorithms considered for implementation in the project.

2.3.1.1 Frame Differencing

This is the simplest approach to the problem as it only considers two frames at a time, making it only work in certain scenarios where the speed of the objects and frame rate of the camera allow it. In this model a pixel is considered as foreground if it satisfies the equation presented in 2.2, and since the only parametrizable value is the Threshold, the whole segmentation process is very sensitive to any changes in this value.

$$|frame_i - frame_{i-1}| >= Threshold \tag{2.2}$$

2.3.1.2 History Based Background Model

In order to make this segmentation method more robust, in the early 2000's Velastin [LV01] and Cucchiara [CGPP03] proposed a mathematical model to take into consideration past frames when calculating the Background Model of the scene. The most simple version of the improved algorithm used a running average of the past frames as seen in 2.3. This eliminates the need to store the frames in memory as the new average can be calculated using the previous results.

$$BackgroundModel_{i+1} = \alpha * Frame_i + (1-\alpha) * BackgroundModel_i \tag{2.3}$$

Instead of the running average one can use the median value of the last n frames, improving the method reaction to outlying values. In both however, the history can be just the n frames or a weighted average where recent frames have more weight. Both these methods cannot cope with intermittent changes on the background, such as moving leaves against a building facade, and to solve this problem a new approach was proposed, the Mixture of Gaussians which models each pixel according to a mixture of Gaussian distributions, usually 3 to 5, but in later research [Ziv04] a method was developed to calculate the number of distributions needed on a per pixel basis.

Copiar resto dos metodos

2.3.2 Feature Tracking

When the literature mentions features it is referring to interesting parts of the image that can be detected and matched in multiple different images of the same scene. The process of detecting these features is a basic step in many computer vision applications, as they allow for a broad comprehension of the scene being observed. These features can take the shape of:

- Edges - A boundary in an image (Further explored in the next section)
- Corners or Interest Points - Points in the image that have distinctive neighbourhood, making them good candidates for tracking
- Blobs or Regions of Interest - Connected areas of the image that share a property like colour or intensity

The OpenCV implementation of *goodFeaturesToTrack* is based on the work of Jianbo Shi et al. [ST94] that tackles the problem of selecting features that can be tracked consistently across the scene, usually feature points that correspond to physical points in the scene. It does so by calculating a value they named *dissimilarity* that quantifies how much the feature appearance has changed between the first and the current frame. When this value grows past a certain threshold, the point is discarded and no longer tracked.

The two main improvements presented in this paper regarding this problem are the fact that the authors allow for both linear warping and translation regarding the image motion model, and the second being a numerically superior method for calculating the *dissimilarity* value when compared to existing work. The other main contribution is a new value to measure how good a feature is, beside the already common *interest* and *cornerness*. This value is the *texturedness* and is a result of the authors findings that features with good texture are more easily tracked.

2.3.3 Edge Detection

Detecting the edges in an image is an important step into the understanding of its contents, allowing segmentation based on surfaces and a spatial perception of the scene. The work presented by Marr and Hildreth [MH80] defines edges as part visual and part physical and proceeds to show that the two are interconnected. The first step of the proposed process is to calculate the zero-crossings of equation 2.4, the Laplacian of a two dimensional Gaussian applied to different scales of the image, which returns both the points of intensity change and the slopes of the function at that point.

$$\nabla^2 G(x,y) * I(x,y) \tag{2.4}$$

What the authors noted was that if an edge was detected at a certain scale it appears also at adjacent scales, which they called "spatial coincidence assumption". This theory can be reversed, if there is a convergence of edges on multiple scales there is a real physical edge, and based on this, if we find zero-crossings on neighbour scales, we can assume there is a real edge in that region. From here multiple implementations of the algorithm appeared, the most used one being presented below, the Canny Edge Detector using a Sobel filter. Although this was among the first implementations of an edge detector it is the most rigorously defined one and is still considered to be state of the art.

The Sobel filter is used to create an image that accentuates the edges of an original image. This process works by convolving two kernels with the image, one for the horizontal and the other for the vertical direction, that return the intesity change in that pixel for each one. The filters can be seen in 2.5 as they were presented by Irwin Sobel [Sob89].

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}, G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.5}$$

Using G_x and G_y it is possible to retrieve the regions of the image that contain edges by calculating the magnitude of the gradient vector as the square root of the sum of the squares of both components and thresholding these values. One of the features of this process is the ability to retrieve the direction of the edge from these values, using the inverse tangent operation.

The Canny Edge Detector uses the result of the Sobel filter as input and filters the most relevant edges as well as thinning them to 1-pixel wide lines. To accomplish this John Canny [Can86] used an optimization process comparing the value of the gradient magnitude to the one of the neighbour pixels along the direction returned by the Sobel filter, and keeping only the one with the largest value along the edge.

2.3.4 Object Based

Object Based segmentation relies on the a priori knowledge of the geometry of the objects we want to extract from the image. It is mainly used in medical applications such as in [SMG+93] where it is used to extract a model of the brain surface from MRI scans. To be able to use this approach to solve our issue of vehicle detection we would need detailed models of all the vehicles that can possibly pass through the scene we are analysing, which is not feasible, and thus this method was not further explored.

2.4 Object Detection

In this section we will analyse processes to detect objects of interest in images, a necessary step of any traffic surveillance system.

2.4.1 Connected Component Labelling

Connected component labelling is a process by witch the subsets of connected components are labelled according to the user needs. We will explore some of the most common implementations and how they evolved over time.

The one-pass solution presented by Abubaker [AQIS07] is a fast and simple method to implement. It begins by labelling the first pixel of the image to 1, if it is a background pixel, skip to the next one, otherwise add it to a queue. While there are elements in the queue remove the one at the top and analyse its neighbours, if they are in the foreground add them to the queue with the current label. When there are no more elements in the queue increase the label number by one and proceed to the next unexplored pixel.

The two-pass solution presented by Shapiro et al. [SS01] uses the bi-dimensionality of the image data to transverse the pixels. The process, as the name implies, iterates over the image two times, the first time to assign temporary labels and the second to reassign these labels to the smallest ones available. It uses a neighbouring table to know what labels are adjacent to each other.

- First pass

```
    Perform a Raster Scan to analyse all the pixels
    For each pixel, if it belongs to the foreground
        Get all its neighbours (four or eight depending on the connectivity assumed)
        Assign the smallest label from the neighbours
        Add the the previous and the new label to the neighbouring table
        If there are no neighbour pixels add a new label
```

- Second pass

```
    Perform a Raster Scan to analyse all the pixels
    For each pixel assign the lowest label from the neighbouring table
```

The process is illustrated in figure 2.10.

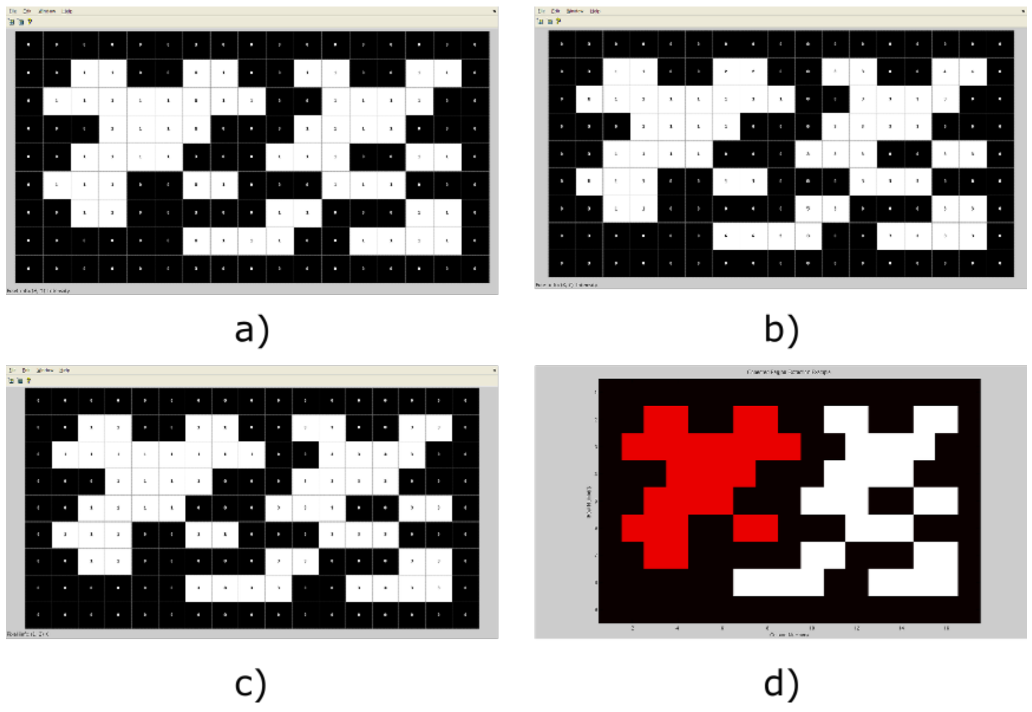


Figure 2.10: Graphical example of the two-pass algorithm [Wik17]
a) Original Image b) First Pass c) Second Pass d) Coloured Result

While these are the most basic solutions for the problem, a number of authors proposed enhancements to improve their performance, the most prominent ones being the Contour Tracing Labelling by F. Chang et al. [CCL04], a method that uses contour tracing to identify internal and external contours of each component. These contours are then used to label the connected components when the image is traversed horizontally by counting the number of contours passed. The Hybrid Object Labelling presented by Herrero [MH07] presents a method that combines recursive and iterative analysis, finding unlabelled pixels and recursively finding their neighbours until a labelled one is found, then proceeding to the assignment in the adjacent rows.

```
Input: Set of points in the plane
Output: List of vertices of the convex hull in clockwise order

Sort the points by their x-coordinate, creating a sequence p_{1}, p_{2}, ..., p_{n}

Put point p_{1} and p_{2} in a list called L_{upper}
for i=3 to n
  Append p_{i} to L_{upper}
    While L_{upper} has more than two points and the last three points don't form a
         right turn
      Delete the middle of the last three points from L_{upper}

Put points p_{n} and p_{n-1} in a list called L_{lower}
for i=n-2 down to n
  Append p_{i} to L_{lower}
    While L_{lower} has more than two points and the last three points don't form a
         right turn
      Delete the middle of the last three points from L_{lower}

Remove first and last point from L_{lower} to avoid duplication of the points where
     the upper and the lower hull meet
Append L_{lower} and L_{upper} and return that list
```

Listing 2.1: Convex Hull calculation

The implementation of OpenCV is based on the work of Grana et. al [GBC10]. It uses a new concept in this area that the authors describe as a single entry decision table and enhancing them to create the OR-decision tables that allow multiple equivalent decisions for the same conditions. The most convenient decision is selected by creating a decision tree, branching it whenever multiple decisions are possible. To improve the method performance the access to the image is made block-wise instead of the more usual pixel-wise manner. This is the chosen method for implementation in OpenCV due to it's performance compared to other state of the art algorithms.

It is interesting to note the YACCLAB (Yet Another Connected Components Labelling Benchmark) [GBBV16] project, a benchmark platform that allows researchers to compare the performance of their connected components labelling methods to the methods of other researchers. The novelty of this platform is the fact that it uses the authors implementations instead of their own, allowing the algorithms to perform exactly as intended, with all the tweaks and tricks.

2.4.2 Convex Hull

A convex hull is the smallest convex shape that encompasses all the points of the figure. This problem is usually solved in two or three dimensions, although some solutions can solve it for higher number of dimensions [Cha93]. In the field of computer vision however only the second and third dimension are needed.

An algorithm to calculate a convex hull from a set of points [dB08] can be seen in listing 2.1.

2.5 Fuzzy Sets

Fuzzy sets were introduced by L.A. Zadeh [Zad65] and Dieter Klaua [Kla67] in the mid 60's. Although the work from Klaua is written in german, a very detailed analysis of his work can be found in a paper by Siegfried Gottwald [Got10].

Fuzzy sets are a class of sets where instead of each member belonging to a class like in regular sets, they have a degree of membership to each class ranging from zero to one. This degree is determined by the membership or characteristic function of the class it represents, and can also be seen as a truth value in certain applications.

Since their formulation, the use of these sets as spread to multiple areas: text processing [CBK00] where they are used to model linguistic modifiers taking into account relationships between objects; vehicle safety [DSM12] in which they are used to build a fuzzy deterministic non controller type system that is able to indicate if a battery is on fire given a broad range of inputs; quality of life analysis [PB12] where they are used to analyse the effect of noise pollution caused by road traffic on the efficiency of human workers.

2.6 Traffic Analysis

The topic of traffic analysis using computer vision has been studied by multiple authors who tackled multiple problems. Since our solution is to be able to run on both highway and urban environments, both areas were studied.

2.6.1 Highway Scenarios

Highway scenes are more easily analysed due to the following attributes contributing to a controlled environment:

- Known object types in the scene, as we only have vehicles to analyse, with no pedestrians or other actors present;
- Cameras positioning is studied towards this type of application, meaning there is little to no occlusion of objects;
- The trajectory of the objects has little to no deviation.

In their work "*Detection and classification of vehicles*" [GMMP02] S. Gupte et al. present a method to replace the magnetic loop detectors with a vision-based video monitoring system. Their approach needs minimum scene-specific knowledge, an advantage that speeds up the set-up process. The vehicle tracking method proposed is based on a state machine where a region can belong to one of five different states:

- Update, when a region from the current frame matches exactly to a region in the previous frame;

- Merge, when multiple regions from the previous frame merge into a single one in the current frame;
- Split, when a single region from the previous frame split into multiple regions in the current frame;
- Disappear, when a region from the previous frame is not matched to any region from the current frame;
- Appear, when a region appears and is not near a recently disappeared region;

Using these five states they are able to track vehicles crossing the scene even if they are partially occluded behind other vehicles or scene features. They also present a classification technique that distinguishes cars from non-cars (vans, SUVs, semis and buses) based on the size of the region using a fix threshold value. This was able to achieve a 70% classification success rate in a 20 minutes video.

Goo Jun et al. [JAG08] present a method for segmenting different vehicles inside the same region retrieved by the background subtracter. To do so they calculate a motion vector for each vehicle by tracking feature points across multiple frames and then clustering them based on their velocity.

2.6.2 Urban Scenarios

Urban scenes are harder to analyse due to the following attributes:

- Unknown object types in the scene, making it difficult to segment and classify them;
- Cameras are positioned in low locations, creating a lot of occlusion between objects and other objects, and objects and features in the scene;
- Unexpected trajectories of the objects present in the scene;
- Objects may stop at any point and be classified as background by the segmentation process.

Buch et. al [BOV08] present a method to detect and classify vehicles in urban scenes that use 3D models whose projections are compared to the segmented regions. This method has a reported accuracy of 100% under sunny weather but this value declines when faced with foggy or rainy weather.

Hashmi et. al [HK12] propose an approach to gather statistics of intersection usage, analysing how many vehicles pass through the scene and also from where they come and where they exit. This evaluation is only possible using video, as the traditional methods of counting vehicles cannot establish a relationship between detections at different points of the intersection. This paper however can only deal with free-flowing traffic and has problems with stopped vehicles.

2.7 Summary

This chapter covered the beginning of the field of computer vision and its multiple usages, reviewed basic concepts of image treatment, focusing on the ones that would be used during the development of the solution.

When reading the works related to traffic analysis we detected a largely unsolved issue, related to the segmentation of stopped vehicles in urban scenes.

This review was written taking into consideration that some of the readers may not be familiarized with the area, and thus contains overviews of some basic methodologies that are common in computer vision.

Chapter 3

Implementation

Este capítulo pode ser dedicado à apresentação de detalhes de nível mais baixo relacionados com o enquadramento e implementação das soluções preconizadas no capítulo anterior. Note-se no entanto que detalhes desnecessários à compreensão do trabalho devem ser remetidos para anexos. Dependendo do volume, a avaliação do trabalho pode ser incluída neste capítulo ou pode constituir um capítulo separado.

Capitulo implementação Como resolver os desafios Estabilizaçao BG Sub

As previously stated, the work done for this project consists of a module for the "Video Server" being developed by LIACC, that will perform all the required analytics of the video received. This chapter provides the details of the implementation of this module, developed during the dissertation semester between February and June 2017. Tasks performed involved designing an appropriate architecture, able to scale with the project, treating the images received from the video streams to detect events and extract information.

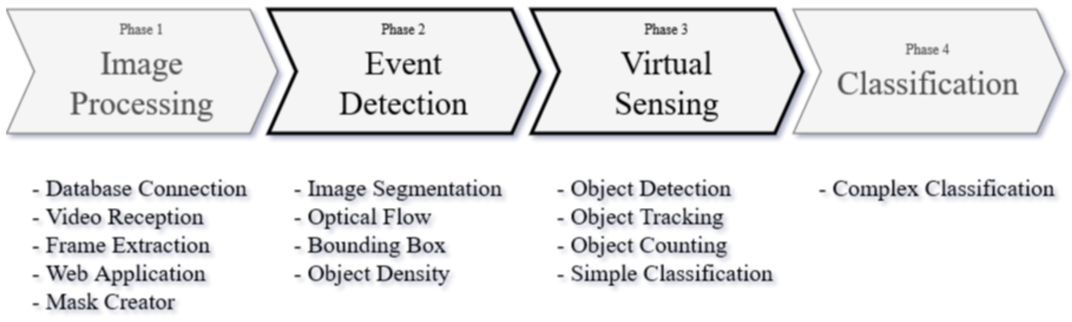


Figure 3.1: Project Planning

It is expected that this chapter provides some insight on how the theoretical backgrounds presented before were used during the project development, namely phase 2 and 3 of the planning presented in 3.1. Phase 2 focused on the detection of relevant events to the traffic controller such as suspicious approximation to the camera, intrusion in prohibited areas, fallen objects on the road and wrong way travelling, while phase 3 aimed to count and classify vehicles in both urban and high-way scenarios.

3.1 Technology

In order to build this project we needed both a library of already implemented computer vision algorithms as well as a simple way to retrieve frames from both video streams and files. This section describes what were the chosen technologies including a brief description and why it was chosen.

3.1.1 OpenCV

OpenCV is a library composed of implementations of useful computer vision algorithms implementation, widely used across the industry and academy. It has interfaces for multiple programming languages, like C++, Java and Python, but is natively written in C++ in order to take advantage of low level performance enhancements, as performance is an important factor in real time computer vision applications [Ope17a].

The library contains over 2500 algorithms ranging from the more basic image processing, such as filtering, morphology operators and geometric transformations, to more complex ones that are able to compare images, track features, follow camera movements and recognize faces, among others. Along with the image processing capabilities, OpenCV also ships with interfaces to stereo cameras such as Kinect that allow users to retrieve a cloud of 3D points and a depth map from the captured image. This was the chosen library as there existed already previous work at LIACC using it, which could be leveraged for this project.

3.1.2 JavaCV

JavaCV is a wrapper for OpenCV written in Java that works on top of the JavaCPP Presets, a project that provides Java interfaces for commonly used C++ libraries, such as OpenCV and FFmpeg, the ones we are using, as well as CUDA, ARToolKitPlus and others. It provides access to all the functionalities of OpenCV inside a Java environment, and was the chosen solution as there was already experience inside LIACC working with this technology.

Even though the code is written in Java and runs inside a Java Virtual Machine, the code from OpenCV is compiled from C/C++ and the memory of the objects created there is allocated in a separate thread. This made it impossible to rely on the garbage collector to do the memory management, and necessary to manually delete the native objects. Failure to address this issue causes the system to run out of memory and a subsequent program crash.

3.1.3 FFmpeg

FFmpeg is a framework that performs encoding, decoding, transcoding, streaming and filter operations in all the well known video formats, across a large number of platforms. It is used in the

```java
// videopath can be either a stream path or a file path
FFmpegFrameGrabber _frameGrabber = FFmpegFrameGrabber.createDefault(videopath);
Frame currentFrame = null;

while( (currentFrame = _frameGrabber.grabImage()) != null) {
  // Use grabbed frame
}
```

Listing 3.1: FFmpegFrameGrabber usage example

"Video Server" to read streams from any format and restream them all to the same format. In the Analytic module it is used to grab the frames from both the video files and the streams using the same code, as shown in 3.1.

3.2 Architecture

3.2.1 Implementation Architecture

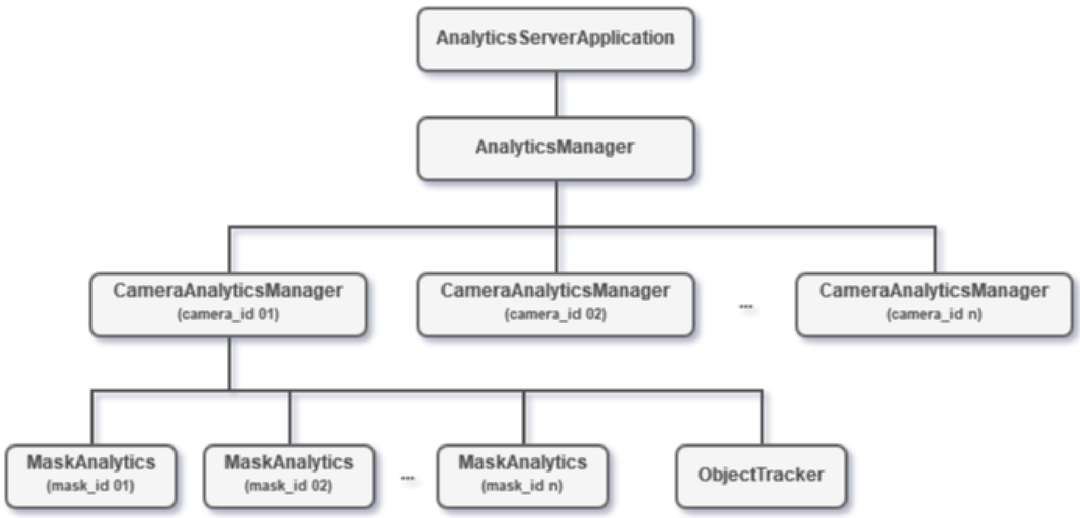

AnalyticsServerApplication
AnalyticsManager
CameraAnalyticsManager (camera_id 01)
CameraAnalyticsManager (camera_id 02)
...
CameraAnalyticsManager (camera_id n)
MaskAnalytics (mask_id 01)
MaskAnalytics (mask_id 02)
...
MaskAnalytics (mask_id n)
ObjectTracker

Figure 3.2: Manager Architecture

The Analytics module was designed to comply with the following specifications, imposed by both our partner and to comply with the already developed modules.

- Run uninterruptedly waiting for requests to be made;
- Allow for video analysis with no significant delay;
- Process a large number of video inputs at the same time;
- Receive input from streams or files;

- Specify which analysis are to be run on a specific video;
- Use free-to-use or open-source tools;
- Allow for deployment scalability.

The application main thread runs an instance of the AnalyticsManager class that implements the Java Runnable interface. This thread will launch one CameraAnalyticsManager instance for each video to be analysed, be it from stream or from file, and keep track of each worker status, disposing of them when they finish, and launching new ones when a request is received. The CameraAnalyticsManager is then responsible to query the database in order to find out what are the analytics the user wants to retrieve from the video through information that is stored in the *camera* table of the database. This process then starts a frame grabber that will convert each frame of the video into its representation in OpenCV and feed it to each Analyser through a queue. This enables each one of these workers to run at his own pace and if one of them runs slower than the pace at which the frame grabber reads the images, it will simply queue them up and not slow down the remaining workers. The drawback of this solution is that it is theoretically possible to run out of memory to keep these frames, although this limit was not reached during the testing phase.

3.3 Segmentation

This section describes how the segmentation of the received images is performed, and how it returns a foreground mask representing the moving areas of the scene.

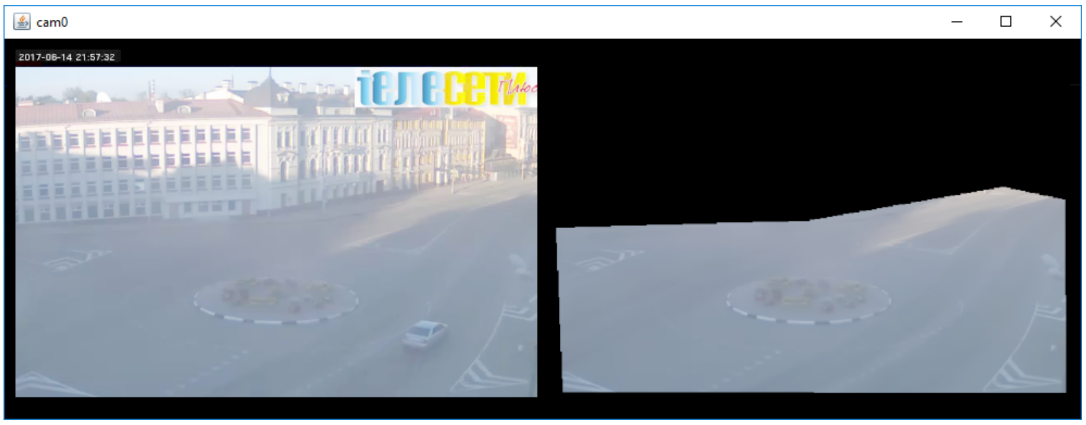


Figure 3.3: Background Subtraction - Background Model

In figure 3.3 we can see the original frame on the left and the calculated background model of the scene on the right. To achieve this result, the first step is to mask the obtained frame in order to prevent uninteresting regions of the image from being processed by the Background Subtracter. The masking process consists in placing a binary image, called a mask, over the original image and removing all the information where the mask has false values.

The application of the mask solves an issue where moving or changing regions of the image outside our area of interest would create unwanted artefacts, for example moving trees due to the wind blowing, or the issue that occurred in this scene, where a car passing would be reflected in the windows of the building.

The next step of the Segmentation process is to feed the masked frames into a Background Subtraction algorithm that will use them to update its internal representation of the scene. This project uses the OpenCV implementation of the Mixture of Gaussians algorithm that allows the user to tune:

- The number of past frames considered on the background calculation
- The threshold value from which a pixel is considered to be foreground, compared to the difference between its current value and the one from the background model
- The learning rate of the algorithm, how much each new frame influences the model

After updating the background model we can retrieve from the Background Subtracter its foreground mask, a rough representation that is calculated by subtracting the background model from the current image and thresholding it, thus returning only the pixels where the difference is significant enough.

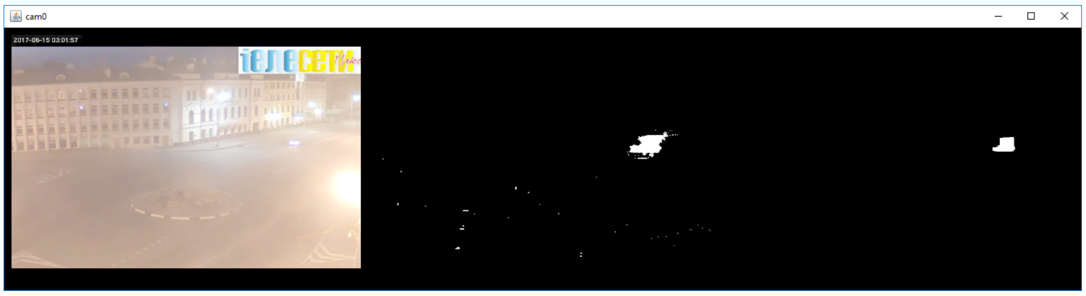


Figure 3.4: Background Subtraction - Foreground Mask

As we can see in the middle image of figure 3.4 the foreground mask from the Background Subtracter can have a lot of noise due to lighting conditions. To solve this issue two morphology filters are applied to the image: a squared erosion filter to remove the small speckles that appear in the mask; followed by a larger circular dilation filter to consolidate the positive regions of the mask, as the wind shield and windows of the vehicles are usually detected as background due to their dark colour and/or reflection of the environment.

3.4 Object Detection

This section describes how the information about the objects' size and position are retrieved from the binary mask returned from the segmentation process.

OpenCV provides BlobDetector [Ope17b], a family of functions that retrieve information of blobs in an image. It works by thresholding the input into multiple binary images using a range of threshold values, grouping the connected white pixels at each one of these images in binary blobs and then merging those that are close enough to each other into larger blobs. This method allows users to filter returned blobs based on 3 properties:

- Area - The area of the region
- Circularity - Ratio between the area of the smallest involving circle and the area of the region
- Convexity - Ratio between the are of the convex hull of the region and its area
- Inertia - Elongation of the region (0 for lines, 1 for circles)

This however does not work as intended for our input, an already binarized image where we just need to group the connected pixels. For this the project uses the OpenCV function *connectedComponentsWithStats*, that retrieves the groups of connected pixels in a binary image along with their area, width, height and both the leftmost and topmost coordinate of the group's bounding box. From this data we can create an ImageObject instance that will represent a moving object in the scene, using the regions with a size above a given threshold, in order to prevent detection of small objects or noise that got through the filtering process.

3.5 Object Tracking

In order for the solution to correctly analyse the objects of the scene it needs to track them during their lifetime, in other words, establish a relation between the objects identified on a frame with the objects identified in the next one. This is done by iterating through each one of the new objects and finding which existing object is closer to him. If the distance between them is smaller than the size of the existing object then they are matched.

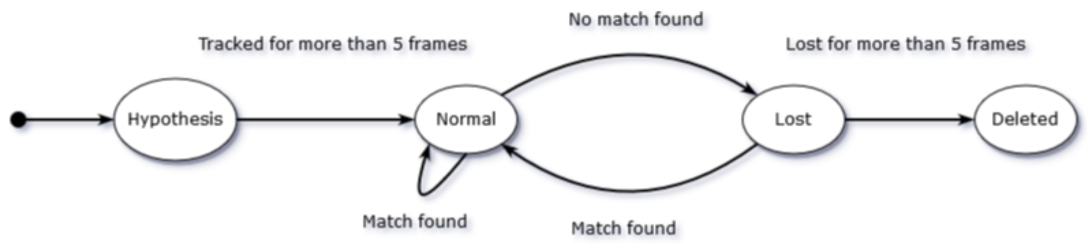


Figure 3.5: Object Life Cycle State Machine

To enhance this mechanism a sub-set of the state machine presented by Jodoin et. al [JBS14] which allows us to model the life cycle of an object. As seen in 3.5 an object starts as an Hypothesis, and cannot be used for analytics while in this state. After being tracked for 5 frames it then changes its state to Normal, where it stays until no match can be found, either from leaving the scene or due to problems with the input frame. An object in this state remains for a maximum of 5 frames, after which it is deleted or until a new object is found in the vicinities of it's last known position.

This technique relies on objects appearing close to their previous position in the next scene, and because the project was designed to run in high-way scenarios, where vehicles travel at high speeds, there was the need to improve it. The chosen solution was to follow a method used by Chris Dahms in his vehicle counting software [Dah17] that predicts an object next position by calculating an weighted average of the previous inter-position deltas and summing it to its current position as shown in 3.1.

$$NextPos = CurrentPos + \sum_{i=1}^{5} (PosHist(i) - PosHist(i-1)) * (5-i) \tag{3.1}$$

In figure 3.6 we can see the result of the tracking process on the right most image. Each detected object is circled in a colour representing its current state in the state machine, blue for normal and red for lost, and shows the trail represents its path across the scene.

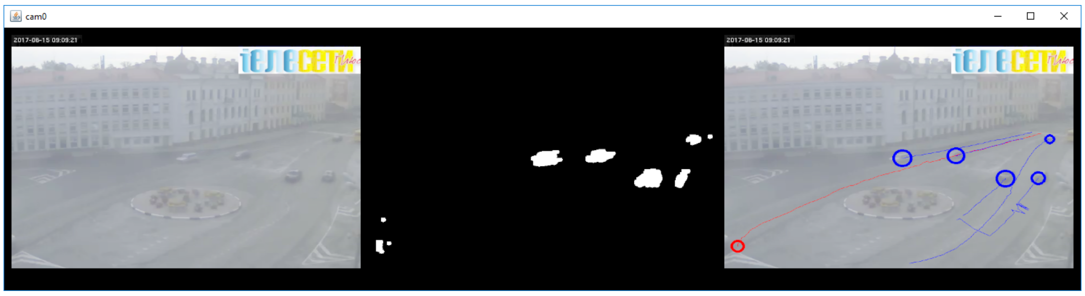


Figure 3.6: Object Tracking

3.6 Object Classification

In order to classify the objects present in the scene into light or heavy vehicles a fuzzy set is used. This set is calculated at run time based on a mask drawn by the user via the web interface that roughly approximates the size of a light vehicle. The area of that mask (A_{Light}) is used as a base value to create the fuzzy set shown in figure 3.7. This set has 2 series, one for light vehicles, drawn in blue, and one for heavy vehicles, drawn in red.

The blue series peaks at A_{Light}, where we have 100% certainty that a matched object is a light vehicle. The point immediate points are at 2/3*A_{Light}, where the trust is 80% and at 4/3*A_{Light} where it drops to 60%. Any object with area below 1/2*A_{Light} or above 2*A_{Light} are considered to have 0% chance to be light vehicles.

The red series peaks at 2*A_{Light}, and any object whose area is larger than this value is considered to be an heavy vehicle with 100% confidence. At the same time, any object whose area is

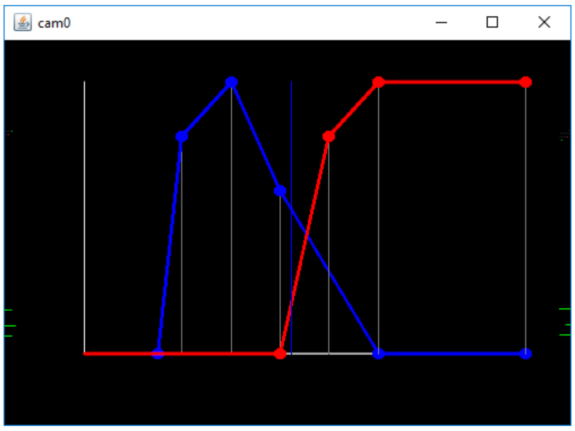


Figure 3.7: Object Classifier

smaller than 4/3*A_{Light} is never considered to be an heavy vehicle, from where the certainty rises to 80% at 5/3*A_{Light}.

When an object is counted in one of the virtual sensors its area is passed to this classifier where an interpolation is calculated for each series, returning the certainty with which it belongs to each one of the classes. Using these values the process can distinguish classifications with certainty below and above a user specified threshold to be treated separately.

3.7 Feature Point Tracking

The need to track feature points stems from the fact that without them we rely wholly on the background subtraction to process the image. Using feature points we can extract and track details of the vehicles that would otherwise be lost due to the process only working with a binary mask.

We can use these points to distinguish vehicles grouped in a single binary region, as they are characterized by This way, even if a vehicle stops for a long time and starts to be considered background we can use its tracked points to maintain the position of these stopped vehicles as they regain motion.

3.8 Utility

Over the development of the project the rise needed to implement some functionalities to enhance the code flow and to help the visualization of the implemented algorithms. This section will describe the most relevant ones for the reader to analyse and implement in its own projects.

3.9 Summary

Although dealing with a framework with no proper documentation was a difficult challenge to overcome, the performance obtained from the native code was essential for the final result, as the software is now able to track objects in a busy scene faster than the camera’s capture rate, allowing us to process a video file in a shorter period of time than it’s length.

Chapter 4

Conclusões e Trabalho Futuro

Deve ser apresentado um resumo do trabalho realizado e apreciada a satisfação dos objetivos do trabalho, uma lista de contribuições principais do trabalho e as direções para trabalho futuro. A escrita deste capítulo deve ser orientada para a total compreensão do trabalho, tendo em atenção que, depois de ler o Resumo e a Introdução, a maioria dos leitores passará à leitura deste capítulo de conclusões e recomendações para trabalho futuro.

Capitulo conclusão Continuaçao para tornar cada parte mais robustas (cada um pode ser uma tese)

This chapter presents an overview of the work done, as well as an evaluation of the state of completion achieved during the semester, followed by a list of the contributions and a foresight into the next steps of the project.

4.1 Satisfação dos Objetivos

4.2 Trabalho Futuro

Referir agrupamento com base no match n para n

Contagem tempo em video

Analise cruzamentos

References

[ACM12] ACM. The 2012 ACM Computing Classification System — Association for Computing Machinery, 2012.

[AQIS07] A. AbuBaker, R. Qahwaji, S. Ipson, and M. Saleh. One Scan Connected Component Labeling Technique. In *2007 IEEE International Conference on Signal Processing and Communications*, pages 1283–1286, November 2007.

[Aya98] N. Ayache. Medical image analysis a challenge for computer vision research. In *Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No.98EX170)*, volume 2, pages 1255–1256 vol.2, August 1998.

[BMCM97] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik. A real-time computer vision system for measuring traffic parameters. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 495–501, June 1997.

[BOV08] N. Buch, J. Orwell, and S. A. Velastin. Detection and classification of vehicles for urban traffic scenes. In *2008 5th International Conference on Visual Information Engineering (VIE 2008)*, pages 182–187, July 2008.

[BP98] Jorge Badenas and Filiberto Pla. Applying computer vision techniques to traffic monitoring tasks. In *Methodology and Tools in Knowledge-Based Systems*, pages 776–785. Springer, Berlin, Heidelberg, June 1998.

[BT73] T. O. Binford and J. M. Tenenbaum. Computer vision. *Computer*, 6(5):19–24, May 1973.

[Can86] J. Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, November 1986.

[CBK00] Martine De Cock, Ulrich Bodenhofer, and Etienne E. Kerre. Modelling Linguistic Expressions Using Fuzzy Relations. In *In 'Proceedings of the 6th International Conference on Soft Computing*, pages 353–360, 2000.

[CBMM98] Benjamin Coifman, David Beymer, Philip McLauchlan, and Jitendra Malik. A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Research Part C: Emerging Technologies*, 6(4):271–288, August 1998.

[CCL04] Fu Chang, Chun-Jen Chen, and Chi-Jen Lu. A linear-time component-labeling algorithm using contour tracing technique. *Computer Vision and Image Understanding*, 93(2):206–220, February 2004.

[CGPP00] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Statistic and knowledge-based moving object detection in traffic scenes. In *ITSC2000. 2000 IEEE Intelligent Transportation Systems. Proceedings (Cat. No.00TH8493)*, pages 27–32, 2000.

[CGPP03] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting moving objects, ghosts, and shadows in video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1337–1342, October 2003.

[Cha93] Bernard Chazelle. An optimal convex hull algorithm in any fixed dimension. *Discrete & Computational Geometry*, 10(4):377–409, December 1993.

[Dah17] Chris Dahms. OpenCV_3_Car_Counting_Cpp, May 2017.

[dB08] Mark de Berg. *Computational Geometry: Algorithms and Applications*. Springer Science & Business Media, March 2008. Google-Books-ID: tkyG8W2163YC.

[Dou92] Edward R. Dougherty. *An Introduction to Morphological Image Processing*. SPIE Optical Engineering Press, 1992. Google-Books-ID: 1kvxAAAAMAAJ.

[DSM12] Macam S. Dattathreya, Harpreet Singh, and Thomas Meitzler. Detection and Elimination of a Potential Fire in Engine and Battery Compartments of Hybrid Electric Vehicles. *Advances in Fuzzy Systems*, 2012:e687652, 2012.

[DW84] K. W. Dickinson and R. C. Waterfall. IMAGE PROCESSING APPLIED TO TRAFFIC, 1-A GENERAL REVIEW. *Traffic Engineering & Control*, 25(1), January 1984.

[FPWW03] R. Fisher, S. Perkins, A. Walker, and Wolfart. Morphology, 2003.

[GBBV16] C. Grana, F. Bolelli, L. Baraldi, and R. Vezzani. YACCLAB - Yet Another Connected Components Labeling Benchmark. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 3109–3114, December 2016.

[GBC10] C. Grana, D. Borghesani, and R. Cucchiara. Optimized Block-Based Connected Components Labeling With Decision Trees. *IEEE Transactions on Image Processing*, 19(6):1596–1609, June 2010.

[GMMP02] S. Gupte, O. Masoud, R. F. K. Martin, and N. P. Papanikolopoulos. Detection and classification of vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 3(1):37–47, March 2002.

[Got10] Siegfried Gottwald. An early approach toward graded identity and graded membership in set theory. *Fuzzy Sets and Systems*, 161(18):2369–2379, September 2010.

[GW92] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. 1992.

[HBJ+16] K. Hammoudi, H. Benhabiles, A. Jandial, F. Dornaika, and J. Mouzna. Self-driven and direct spatio-temporal mechanisms for the vision-based parking slot surveillance. In *2016 SAI Computing Conference (SAI)*, pages 1327–1329, July 2016.

[HK12] M. F. Hashmi and A. G. Keskar. Analysis and monitoring of a high density traffic flow at T-intersection using statistical computer vision based approach. In *2012 12th International Conference on Intelligent Systems Design and Applications (ISDA)*, pages 52–57, November 2012.

[Hua96] T. Huang. Computer Vision : Evolution And Promise, 1996.

[JAG08] G. Jun, J. K. Aggarwal, and M. Gokmen. Tracking and Segmentation of Highway Vehicles in Cluttered and Crowded Scenes. In *2008 IEEE Workshop on Applications of Computer Vision*, pages 1–6, January 2008.

[JBS14] J. P. Jodoin, G. A. Bilodeau, and N. Saunier. Urban Tracker: Multiple object tracking in urban mixed traffic. In *IEEE Winter Conference on Applications of Computer Vision*, pages 885–892, March 2014.

[Jen15] Niall Jenkins. 245 million video surveillance cameras installed globally in 2014 - IHS Technology, 2015.

[Kla67] Dieter Klaua. Ein Ansatz zur mehrwertigen Mengenlehre. *Mathematische Nachrichten*, 33(5-6):273–296, January 1967.

[Koz17] Michael Koziol. 'World first': Government moves to radically overhaul Australia's international airports. *The Sydney Morning Herald*, January 2017.

[Lab17] Virtual Labs. Theory - Virtual Lab in Image Processing, 2017.

[LB78] Michael L. Baird. SIGHT-I: A Computer Vision System for Automated IC Chip Manufacture. *IEEE Transactions on Systems, Man, and Cybernetics*, 8(2):133–139, February 1978.

[LBT17] Robert P. Loce, Raja Bala, and Mohan Trivedi. Detection of Passenger Compartment Violations. In *Computer Vision and Imaging in Intelligent Transportation Systems*, pages 432–. Wiley-IEEE Press, 2017.

[LCC12] X. Li, Q. Chen, and H. Chen. Detection and tracking of moving object based on PTZ camera. In *2012 IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI)*, pages 493–497, October 2012.

[LKR+16] G. Lira, Z. Kokkinogenis, R. J. F. Rossetti, D. C. Moura, and T. Rúbio. A computer-vision approach to traffic analysis over intersections. In *2016 IEEE 19th International*

Conference on Intelligent Transportation Systems (ITSC), pages 47–53, November 2016.

[LV01] B. P. L. Lo and S. A. Velastin. Automatic congestion detection system for underground platforms. In *Proceedings of 2001 International Symposium on Intelligent Multimedia, Video and Speech Processing. ISIMP 2001 (IEEE Cat. No.01EX489)*, pages 158–161, 2001.

[MCKT00] I. Mikic, P. C. Cosman, G. T. Kogut, and M. M. Trivedi. Moving shadow and object detection in traffic scenes. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 1, pages 321–324 vol.1, 2000.

[MH80] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B, Biological sciences*, 207 1167:187–217, 1980.

[MH07] J. Martín-Herrero. Hybrid object labelling in digital images. *Machine Vision and Applications*, 18(1):1–15, 2007.

[Nav00] Navigant. Transportation Forecast: Light Duty Vehicles, 2017-04-18T20:30:25+00:00.

[Ope17a] OpenCV. About - OpenCV library, 2017.

[Ope17b] OpenCV. OpenCV: Cv::SimpleBlobDetector Class Reference, 2017.

[PB12] Debasish Pal and Debasish Bhattacharya. Effect of Road Traffic Noise Pollution on Human Work Efficiency in Government Offices, Private Organizations, and Commercial Business Centres in Agartala City Using Fuzzy Expert System: A Case Study. *Advances in Fuzzy Systems*, 2012:e828593, 2012.

[Pic04] M. Piccardi. Background subtraction techniques: A review. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, volume 4, pages 3099–3104 vol.4, October 2004.

[PMGT01] A. Prati, I. Mikic, C. Grana, and M. M. Trivedi. Shadow detection algorithms for traffic flow analysis: A comparative study. In *ITSC 2001. 2001 IEEE Intelligent Transportation Systems. Proceedings (Cat. No.01TH8585)*, pages 340–345, 2001.

[RBH90] A. Rourke, M. G. H. Bell, and N. Hoose. Road traffic monitoring using image processing. In *Third International Conference on Road Traffic Control, 1990.*, pages 163–167, May 1990.

[Rep17] ReportLinker. Global Video Analytics Market 2017-2021 - market research report, 2017.

[SJK09] Y. Sheikh, O. Javed, and T. Kanade. Background Subtraction for Freely Moving Cameras. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1219–1225, September 2009.

[SLZ16] C. Supriyanto, A. Luthfiarta, and J. Zeniarja. An unsupervised approach for traffic sign recognition based on bag-of-visual-words. In *2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE)*, pages 1–4, October 2016.

[SMG$^+$93] J. W. Snell, M. B. Merickel, J. C. Goble, J. B. Brookeman, and N. F. Kassell. Model-based segmentation of the brain from 3-D MRI using active surfaces. In *1993 IEEE Annual Northeast Bioengineering Conference*, pages 164–165, March 1993.

[Sob89] Irwin Sobel. An Isotropic 3 3 Image Gradient Operator (PDF Download Available), 1989.

[SS01] Linda G. Shapiro and George C. Stockman. *Computer Vision*. Prentice Hall, Upper Saddle River, NJ, 2001.

[SS11] D. Soendoro and I. Supriana. Traffic sign recognition with Color-based Method, shape-arc estimation and SVM. In *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*, pages 1–6, July 2011.

[ST94] Jianbo Shi and C. Tomasi. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, June 1994.

[Sto80] K. J. Stout. Computer vision and robots. *Production Engineer*, 59(4):9–, April 1980.

[Wik17] Wikipedia. Connected component labeling, June 2017. Page Version ID: 783324000.

[Zad65] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, June 1965.

[Ziv04] Z. Zivkovic. Improved adaptive Gaussian mixture model for background subtraction. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 2, pages 28–31 Vol.2, August 2004.

[ZY14] Daniya Zamalieva and Alper Yilmaz. Background subtraction for the moving camera: A geometric approach. *Computer Vision and Image Understanding*, 127:73–85, October 2014.