UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

VIDEO TOLLING INTEGRATED SOLUTION

Ana Cláudia Bernardes Martins

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Engenharia de Software

Trabalho de projeto orientado por:
Prof. Doutor Carlos Eduardo Ramos dos Santos Lourenço

2020

Agradecimentos

This work proved to be a long and (somewhat needlessly) hard journey, which was nevertheless made possible due to the help and advice of many people. I would like to thank my external supervisor, Ricardo Pinto, as well as Bruno Brito and the rest of the TCoE team, for welcoming me into their midst and providing me with their guidance and resources. Moreover, I am very thankful for the expertise and input of my internal supervisor, Prof. Carlos Lourenço, who provided very useful advice throughout the course of the project. Furthermore, I would like to thank Accenture Technology Solutions for the opportunity to have this project experience within an entrepreneurial context.

Lastly, I would like to thank my family, which I shall do in my native language. Agradeço às minhas avós pelo carinho e paciência ao longo deste percurso. Finalmente, um enorme agradecimento aos meus pais e ao meu namorado pelo apoio incondicional e por todo o incentivo que me deram ao longo destes anos. Amo-vos.

À minha família.

Resumo

A indústria de cobrança de portagens foi instituída no século VII com o intuito de financiar e auxiliar na manutenção de vias públicas através do pagamento de taxas correspondentes ao seu uso. Contudo, o advento do uso massificado de veículos automóveis, e consequente aumento do tráfego, obrigou à adaptação desta indústria aos tempos modernos, tendo sido introduzida uma filosofia de livre trânsito complementar à tradicional paragem para pagamento. A adoção deste tipo de medida foi possível graças ao desenvolvimento de tecnologias de reconhecimento ótico de caracteres, que permitem a identificação da matrícula, aliados ao uso de identificadores registados para cada veículo. Porém, a ausência de paragem implica também a existência de infrações de condutores que circulem com matrículas obscurecidas ou de difícil leitura. Deste modo, é desejável o uso de métodos complementares de auxílio à identificação dos veículos, caso do reconhecimento da marca e modelo dos mesmos (MMR).

Os sistemas de reconhecimento ótico de caracteres com o objetivo de identificar matrículas são já implementados nas soluções concebidas pela Accenture para os seus diversos clientes na área, tornando estes novos métodos complementares numa adição interessante à robustez dos mesmos, de modo a reduzir custos adicionais relacionados com a identificação manual de matrículas através das imagens captadas. O presente trabalho visou então, em primeira instância, o estabelecimento de uma prova de conceito com um modelo arquitetural que permitisse a integração de um sistema de reconhecimento de marca e modelo de veículos com os sistemas informáticos previamente desenvolvidos e que se encontram atualmente em uso por parte dos clientes. Para este modelo foi também estabelecido um conjunto de requisitos, tanto funcionais como não funcionais, com o intuito de minorar, tanto quanto possível, perdas no desempenho e fiabilidade dos atuais sistemas por consequência da introdução deste novo componente de MMR. Os requisitos foram definidos fazendo uso de uma versão modificada do modelo de qualidade FURPS, segundo as boas práticas definidas pela equipa de desenvolvimento do Centro de Excelência de *Tolling* (TCoE) da Accenture Portugal. Adicionalmente, os requisitos definidos foram sujeitos ao estabelecimento de prioridades segundo as regras MoSCoW.

A captura de imagens de veículos em movimento e consequente classificação oferece desafios inerentes à sua complexidade, pelo que foram também efetuadas considerações sobre os fatores de variabilidade que devem ser tidos em conta aquando da conceção de um sistema MMR. Estes fatores foram classificados segundo três áreas principais: propriedades inerentes ao sistema de captura de imagens (RSE), propriedades do evento de captura da imagem, e propriedades do veículo.

A arquitetura proposta para um eventual sistema que possa ser passível de integração com os existentes faz uso da arquitetura dos mesmos, organizando-se em quatro camadas, a saber: acesso a dados (camada inferior), gestão e regras de negócio, avaliação de resultados e aumento da base de conhecimento disponível, e correspondência (camada superior). Para a elaboração da presente prova de conceito, foram deste modo escolhidas tecnologias que permitem a integração com os sistemas Java previamente existentes sem despender demasiado esforço adicional nessa integração. Deste modo, foram utilizadas bibliotecas Python para o uso de OpenCV, que permite o processamento de imagens, e Tensorflow para as atividades relacionadas com *machine learning*.

O desenvolvimento da prova de conceito para estes sistemas envolveu também o teste de hipóteses quanto ao modo mais vantajoso de reconhecimento da marca e modelo dos veículos propriamente dita. Para este efeito, foram equacionadas três hipóteses, que se basearam no uso de dois *datasets* distintos.

O primeiro conceito abordado consistiu em *fingerprinting* de imagens associadas a um *dataset* desenvolvido na Universidade de Stanford, contendo 16185 imagens de veículos automóveis ligeiros em variadas poses, que podem ser divididas segundo 49 marcas e 196 modelos distintos, se for considerada a distinção dos anos de comercialização dos mesmos. Para o efeito, foi usado o modelo de características AKAZE e testados três métodos distintos para efetuar as correspondências: força bruta com teste de rácio descrito na literatura (para dois rácios distintos, 0,4 e 0,7), força bruta com recurso a função de *cross-check* nativa das bibliotecas usadas, e FLANN. A pertença de uma imagem a determinada categoria foi então ditada pelo estabelecimento de correspondências entre os seus pontos-chave e os pontos-chave das imagens do *dataset*, testando vários algoritmos de ordenação para aumentar as probabilidades de correspondência com uma imagem pertencente à mesma classe. Os resultados obtidos demonstraram, no geral, precisões relativamente baixas, sendo que nenhuma ultrapassou os 20% para o reconhecimento da marca ou modelo dos veículos. Contudo, dos ensaios efetuados, dois destacaram-se ao conseguir atingir 16,8% de precisão para a marca e 11,2% para o modelo. Estes ensaios tiveram, de resto, características em comum, sendo que, em ambos os casos, foi utilizado o método de força bruta com rácio de 0,4. Os métodos de ordenação de resultados foram, todavia, diferentes, sendo que num dos casos foi usado o valor máximo de pontos-chave em comum (MV) e no segundo um rácio entre este número de pontos em comum e o número de pontos-chave existentes (MR). De entre ambos, o ensaio que recorreu ao método MR foi considerado estatisticamente mais significativo, dado possuir um valor do coeficiente de correlação k de Cohen mais elevado em relação a MV.

Os parcos resultados obtidos através deste método levaram à tentativa de adoção de uma abordagem diferente, nomeadamente no que tocava à seleção das imagens que deviam ser comparadas, uma vez que os fatores de variabilidade identificados na análise se encontravam demasiado presentes nas imagens do *dataset* de Stanford. Deste modo, a grelha do veículo foi identificada como região de interesse (ROI), dados os padrões distintivos inerentes à mesma e a presença do logotipo identificador da marca à qual pertence o veículo. O objetivo desta nova abordagem residia na identificação desta ROI de modo a proceder à sua extração a partir da imagem original, aplicando-se

depois os algoritmos de *fingerprinting* anteriormente abordados.

A deteção da ROI foi efetuada com recurso a classificadores em cascata, os quais foram testados com dois tipos de características diferentes: LBP, mais rápidas, mas menos precisas, e Haar, mais complexas, mas também mais fiáveis. As imagens obtidas através da identificação e subsequente recorte foram depois analisadas segundo a presença de grelha, deteção da mesma ou de outros objetos, bem como o grau de perfeição da deteção efetuada. A determinação da ROI a recortar foi também avaliada segundo dois algoritmos: número total de interseções entre ROIs candidatas, e estabelecimento de um limiar de candidatos para uma ROI candidata ser considerada ou rejeitada (apelidado de *min-neighbours*). As cascatas foram treinadas com recurso a imagens não pertencentes ao *dataset* de Stanford, de modo a evitar classificações tendenciosas face a imagens previamente apresentadas ao modelo, e para cada tipo de característica foram apresentados dois conjuntos de imagens não correspondentes a grelhas (amostras negativas), que diferiam na sua dimensão e foram consequentemente apelidadas de N_{small} e N_{big}.

Os melhores resultados foram obtidos com o *dataset* N_{small}, estabelecimento de limiar, e com recurso a características Haar, sendo a grelha detetada em 81,1% dos casos em que se encontrava efetivamente presente na imagem. Contudo, esta deteção não era completamente a que seria desejável, uma vez que, considerando deteção perfeita e sem elementos externos, a precisão baixava para 32,3%. Deste modo, apesar das variadas vertentes em que esta deteção e extração de ROI foi estudada, foi decidido não avançar para o uso de *fingerprinting*, devido a constrangimentos de tempo e à baixa precisão que o sistema como um todo conseguiria alcançar.

A última técnica a ser testada neste trabalho foi o uso de redes neuronais de convolução (CNN). Para o efeito, e de modo a obter resultados mais fiáveis para o tipo de imagem comumente capturado pelos RSE em contexto de *open road tolling*, foi usado um novo *dataset*, consistindo de imagens captadas em contexto real e cedidas por um dos clientes do TCoE. Dentro deste novo conjunto de imagens, foi feita a opção de testar apenas a marca do veículo, com essa classificação a ser feita de forma binária (pertence ou não pertence a determinada marca), ao invés de classificação multi-classe. Para o efeito, foram consideradas as marcas mais prevalentes no conjunto fornecido, Opel e Peugeot.

Os primeiros resultados para o uso de CNN revelaram-se promissores, com precisão de 88,9% para a marca Opel e 95,3% para a Peugeot. Todavia, ao serem efetuados testes de validação cruzada para aferir o poder de generalização dos modelos, verificou-se um decréscimo significativo, tanto para Opel (79,3%) como para Peugeot (84,9%), deixando antever a possibilidade de ter ocorrido *overfitting* na computação dos modelos. Por este motivo, foram efetuados novos ensaios com imagens completamente novas para cada modelo, sendo obtidos resultados de 55,7% para a marca Opel e 57,4% para a marca Peugeot. Assim, embora longe de serem resultados ideais, as CNN aparentam ser a melhor via para um sistema integrado de reconhecimento de veículos, tornando o seu refinamento e estudo numa solução viável para a continuação de um possível trabalho nesta área.

Palavras-chave: Portagens, MMR, redes neuronais

Abstract

For a long time, tolling has served as a way to finance and maintain publicly used roads. In recent years, however, due to generalised vehicle use and consequent traffic demand, there has been a call for open-road tolling solutions, which make use of automatic vehicle identification systems which operate through the use of transponders and automatic license plate recognition. In this context, recognising the make and model of a vehicle (MMR) may prove useful, especially when dealing with infractions.

Intelligent automated license plate recognition systems have already been adopted by several Accenture clients, with this new feature being a potential point of interest for future developments. Therefore, the current project aimed to establish a potential means of integrating such a system with the already existing architecture, with requirements being designed to ensure its current reliability and performance would suffer as little an impact as possible.

Furthermore, several options were considered as candidates for the future development of an integrated MMR solution, namely, image fingerprinting of a whole image, grille selection followed by localised fingerprinting, and the use of convolutional neural networks (CNN) for image classification. Among these, CNN showed the most promising results, albeit making use of images in limited angle ranges, therefore mimicking those exhibited in captured tolling vehicle images, as well as performing binary classification instead of a multi-class one. Consequently, further work in this area should take these results into account and expand upon them, refining these models and introducing more complexity in the process.

Keywords: MMR, open road tolling, convolutional neural networks

Contents

List of Figures

List of Tables

List of Equations

Abbreviations

AKAZE	Accelerated KAZE
ALPR	Automatic license plate recognition
ANPR	Automatic number plate recognition (see ALPR)
API	Application programming interface
ASIFT	Affine SIFT (see SIFT)
AUC	Area under the curve
AVA	Availability requirement
avg	Average
AVI	Automatic vehicle identification
BI	Bias index
BRIEF	Binary robust independent elementary features
CBO	TCoE's commercial back-office (see TCoE)
CED	Canny ED (see ED)
CNN	Convolutional neural network
CSIFT	Colour-invariant SIFT (see SIFT)
DB	Database
DIIPC	Design, implementation, interface and physical constraints
DOF	Degrees of freedom
Dr.	Doctor
e.g.	For example, from Latin *exempli gratia*
ED	Edge detection
EH	Edge histogram
et al.	And others, from Latin neuter form *et alia*, masculine *et alii* or feminine *et aliae*
ETC	Electronic toll collection
etc.	And other similar things, from Latin *et cetera*
ETCC	Electronic Transaction Consultants Corporation (company)
ETL	Extraction, transformation, loading
EU	European Union
FAST	Features from accelerated segment test
FCUL	Faculty of Sciences of the University of Lisbon, from Portuguese *Faculdade de Ciências da Universidade de Lisboa*
FLANN	Fast library for approximate nearest neighbours
FP	Fingerprinting
FPR	False positive rate
FR	Functional requirement (business requirement)
FURPS	Functionality, Usability, Reliability, Performance, Supportability
FURPS+	FURPS with DIIPC (see FURPS, DIIPC)
GSIFT	Global SIFT (see SIFT)
GUI	Graphical user interface
HLF	Haar-like features
i.e.	That is, from Latin *id est*
IALPR	Intelligent ALPR (see ALPR)

IBM	International Business Machines (company)
IDE	Integrated development environment
IR	Image review
ISO	International Organization for Standardisation
ITS	Intelligent transport systems
KB	Knowledge base
kNN	k-nearest neighbours
LBP	Local binary patterns
LOG	Laplacian of Gaussian
LP	License plate
LPN	License plate number
LPR	LP recognition (see LP)
LSH	Locality-sensitive hashing
MAI	Maintainability requirement
MCC	Matthews correlation coefficient
MEI	FCUL's Master's programme in Informatics Engineering, from Portuguese *Mestrado em Engenharia Informática* (see FCUL)
ML	Machine learning
MMR	Make/manufacturer and model recognition
MPEG	Moving picture experts group (standard set)
MSc	Master of Science
MTBF	Mean time between failures
N/A	Not available
NFR	Non-functional requirement (operational requirement)
NRT	Non-real-time
n-SIFT	n-dimensional SIFT (see SIFT)
OBO	TCoE's operational back-office (see TCoE)
OBU	On-board unit
OCR	Optical character recognition
ORT	Open-road tolling
PABAK	Prevalence-adjusted bias-adjusted kappa
PCA	Principal component analysis
PCA-SIFT	SIFT using PCA (see SIFT, PCA)
PEI	MEI's Informatics Engineering project, from Portuguese *Projeto de Engenharia Informática* (see MEI)
PER	Performance requirement
PGM	Portable grey map
PI	Prevalence index
PNN	Probabilistic neural network
PPV	Positive predictive value (also called precision)
PRC	Precision
Prof.	Professor
PW	TCoE's public website (see TCoE)
px	Pixel(s)

RCV	Recoverability requirement
REC	Recall
REL	Reliability requirement
RES	Resilience requirement
RGB	Red, green, blue
RMSE	Root-mean-square error
ROC	Receiver operating characteristic
ROI	Region of interest
RSE	Road-side equipment
RT	Real-time
SaaS	Software as a service
SAN	Storage area network
SAP	Systems, applications and products
SCA	Scalability requirement
SCW	Sliding concentric windows
SEC	Security requirement
SIFT	Scale-invariant feature transform
SLAM	Simultaneous localisation and matching
SML	Supervised ML (see ML)
SOA	Service-oriented architecture
SPC	Specificity
SSW	TCoE's self-service website (see TCoE)
SUP	Supportability requirement
SURF	Speeded-up robust features
SVM	Support vector machine
SVOOC	Sparse vector of occurrence counts
TCoE	Tolling Centre of Excellence
TNR	True negative rate
TPR	True positive rate
URPS	NFR characteristics of the FURPS quality model (see FURPS, NFR)
US	United States of America
USA	Usability requirement
USD	US dollar (see US)
UspML	Unsupervised ML (see ML)
VCC	Visual content classification
VLPR	Vehicle LPR (see LPR)
vs.	Against or as compared to, from Latin *versus*
x-NN	kNN with x neighbours (see kNN)

Chapter 1

Introduction

1.1 Motivation

Traffic congestion constitutes a major problem in urban and suburban areas in both developed and developing countries, leading to the differentiated application of taxation of road use as a means of attempting to redirect traffic [1]. This taxation, usually called tolling, can be done manually or automatically. Manual tolling requires the existence of a toll plaza with dedicated lanes and booths were either there is an operator handling the transaction or there exists an automated way of payment, such as debit/credit card or weight-based payment control, usually with no change available. However, since manual tolling requires that the vehicles stop in order to process each payment, queuing becomes a problem [2]. Moreover, due to financial, environmental and land use constraints, building new highways to accommodate for the increasing number of commuting vehicles has ceased to be a viable option [3], and inserting toll plazas into existing highways is a costly and impractical endeavour [2]. These circumstances lead to the emergence of automated tolling and intelligent transport systems (ITS), which include electronic toll collection (ETC). A 2014 report by the European Commission stated that 60% of the European charged road infrastructure is equipped with ETC, counting more than 20 million subscribed road users. The infrastructure used is different from country to country and, in most cases, non-interoperable technologies coexist within the European Union (EU) [4].

ETC uses automatic vehicle identification (AVI) to improve efficiency when collecting tolls from road users, by use of a transponder or on-board unit (OBU), which communicates with a road-side sensor or equipment (RSE) in order to identify the vehicle with no need for its driver to provide further authentication [3]. Initially, its implementation relied on adapting existing toll lanes into mixed lanes (allowing for the use of both ETC and manual toll collection) or converting them into ETC-only lanes, which brought some advantage in terms of traffic flow regulation. However, in roads with no taxation implemented, the constraints associated with constructing toll plazas and dedicated lanes impeded the application of ETC methods as described [5].

In order to bypass these difficulties, the concept of Open-Road Tolling (ORT) arose. ORT eliminates the need for constructing toll plazas altogether, as well as the need for stopping or slowing down too much in order for the OBU to be correctly read. The more strict version of ORT eliminates all vertical structure associated with toll plazas, replacing it with toll gantries from which hang cameras and devices to read OBU from vehicles in transit [5–7]. ORT was adopted by the Illinois Tollway in 2004 as part of their Congestion Relief Plan, a response to the increase in the number of I-PASS (Tollway's ETC system) transponder users, who were by then responsible for 40-45% of the total of transactions performed [8].

The non-existence of barriers creates, however, another problem for tolling authorities, since users who do not have a registered account can freely drive in these highways, incurring in tolling

violations [9]. In these cases, the standard mode of recognition is through the use of automatic license plate recognition (ALPR), in which cameras capture images of each vehicle that passes through the toll gantry. If the vehicle does not carry a valid transponder, the tolling gantries then run an optical character recognition (OCR) algorithm to detect its license plate number (LPN) and proceed to file for payment. ALPR is usually dealt with using computer vision techniques, sometimes aided by machine learning [10,11]. However, even a sophisticated technique like ALPR can be avoided by using a diverse array of diversionary tactics, such as obscuring the license plate or failing to notify authorities of a change of address [9, 10]. Another main issue arising from the use of ALPR is the need for manual image review. Such techniques are indispensable, since, even if drivers do not intentionally obscure or otherwise tamper with their vehicles' license plates, ALPR may not provide an answer with the required degree of confidence. Should such a situation arise, the dubious image would then be forwarded to a human operator, who would be responsible for identifying the offending vehicle's LP whenever possible. This is, therefore, a costly and time-consuming task, which reduces the profitability of tolling companies using ALPR solutions [12, 13].

Alternative methods of identifying offending vehicles are, therefore, needed. Among these, special attention must be paid to the yet unsolved problem of vehicle make and model recognition (MMR), which can act complementary to ALPR in identifying transgressors [14]. This work aims to create the basis for a model that can reasonably predict a vehicle's make and model based on footage collected from existing cameras in toll gantries.

1.2 Host institution

Accenture LLP is a global company founded in 1951 in the United States of America (US) by Arthur E. Andersen and Clarence DeLany, under the name Andersen Consulting. Having been rebranded in 1989 and formally adopting the name "Accenture" in 1991, the company boasted a headcount of 425,000 collaborators and a revenue of USD 34.9 billion as of November 2017. Accenture's headquarters have been established in Dublin, Ireland, since 2009 [15, 16].

Providing services to clients in more than 120 countries and being established in more than 40, Accenture is constituted by five business units, each with its defined purpose, whose identifying logos are depicted in Figure 1.1 [17].

- Accenture **Strategy**: *shapes the future at the intersection of business of technology*. Responsible for the definition and reformulation of the clients' business models in order to stay competitive.
- Accenture **Consulting**: *transforms businesses through industry expertise and insights*. Responsible for designing and implementing transformative business solutions to improve the clients' success in their respective target markets.
- Accenture **Digital**: *creates value through new experiences, new intelligence and new connections*. Responsible for enhancing the clients' digital enterprise capabilities and connections through the use of digital marketing, mobility and analytics.
- Accenture **Technology**: *powers businesses with cutting-edge solutions using established and emerging technologies*. Responsible for the implementation of technological solutions that improve the clients' businesses.
- Accenture **Operations**: *delivers outcomes through infrastructure, security, cloud and business process services*. Responsible for business improvement by introducing scaling capabilities and improving the clients' processes, making them more efficient and flexible.

Figure 1.1: Accenture's business units. (Adapted from [17].)

In Portugal, Accenture has offices in Lisbon, Oporto and Braga, with its headquarters being located on the 16th floor of Amoreiras's Tower 1, situated in Av. Eng. Duarte Pacheco, Lisbon. Among many other teams and projects also located in Lisbon is Accenture's Tolling Centre of Excellence (TCoE, Lisbon Delivery Centre, Amoreiras Plaza, room 7B), where video-based tolling solutions and platforms are developed for client businesses based both in Portugal and overseas (USA, Australia). The tolling project's scope falls in the technology and consulting business units, with particular emphasis on the former.

The current project was performed in TCoE's headquarters, under the supervision of its Delivery Lead and Technology Consulting Senior Manager, Dr. Ricardo Pinto, with the aid of Bruno Brito, Application Development Specialist, and with Faculty of Sciences's (FCUL) mentorship being provided by Prof. Dr. Carlos Lourenço. This work was developed within the scope of FCUL's Master's programme in Informatics Engineering (MEI)'s final project (PEI) course, lasting a total of nine months, with its practical implementation spanning from September 2017 to June 2018. Although not directly integrated in any of the formal teams the tolling project is composed by, this project aims to benefit all of them in the long term, by setting the basis for an additional level of certainty in vehicle identification for tolling transgressions, which may then be implemented by each team as they see fit.

1.3 Summary and contributions

This work aimed at the establishment of a proof of concept for the integration of a make and model recognition into an existing open-road tolling system, through the use of computer vision and machine learning techniques. To this purpose, the following main objectives were defined for contributions to the tolling teams:

- Definition of functional and non-functional requirements for the new MMR system, along with their relative priorities
- Establishment of the quality model to be followed
- Analysis of the degrees of variability to be taken into account when performing MMR
- Proposal for a possible architecture for the system, as well as possible frameworks and patterns that could be used in its implementation
- Comparison of possible computer vision and machine learning algorithms, and combinations thereof

1.4 Structure of the document

This document is organised as follows:

- Chapter 1 - **Introduction**: provides information about the scope and problems addressed by this work, its context, motivation and main goals.
- Chapter 2 - **Contextualisation and methodologies**: addresses the methodologies involved in the course of this work and similar others.

- Chapter 3 - **Related work**: details works performed by other authors on the topic.
- Chapter 4 - **Analysis**: details the work planning, quality parameters and requirements.
- Chapter 5 - **Design**: presents the system architecture and performance indicators.
- Chapter 6 - **Implementation and results**: describes how the system was implemented and presents the obtained results.
- Chapter 7 - **Discussion and conclusion**: provides commentary on the work and its results, and concludes the work by providing an overview, critical discussion of the work and future work to be done.

Chapter 2

Contextualisation and methodologies

2.1 The tolling industry

Stretching as far back as the 7th century BC, levies have been applied to road use with the aim of generating funds to help pay off and maintain the taxed roads [18]. Over the years, with the development of modern civilisation, the demand for ownership of a private, personal vehicle has increased to the point of becoming the norm, and therefore travelling demands have also increased with time. However, since financing new road construction and improvements is increasingly difficult, the rate of growth of travel demand outstrips the growth of road capacity [1]. Modern tolling systems continue to leverage the principle of users paying for the construction and maintenance of the roads they use. However, nowadays tolling is also employed as a means to moderate traffic congestion or air pollution [18].

Toll roads can be divided into three general systems [19]:

- **Open**: possessing mainline barrier toll plazas with payment lanes, this system serves as a single-point access restriction site where all vehicles that intend to use the road are taxed once. Also known as "barrier toll system".
- **Closed**: in which users go through entry and exit toll gantries, with both entry lanes and payment lanes, paying for the distance they travelled on the taxed road. Also known as "ticket system".
- **Open road**: with no toll booths, this method relies on electronic collection of levies when a vehicle passes a strategic point in the road being taxed.

A study published by KPMG International in 2015, which collected data from 43 different public and private entities in the Americas, Europe and Asia, detailed the relative prevalence of each payment method used in the surveyed tolling agencies. [20]. These results are shown in Figure 2.1.

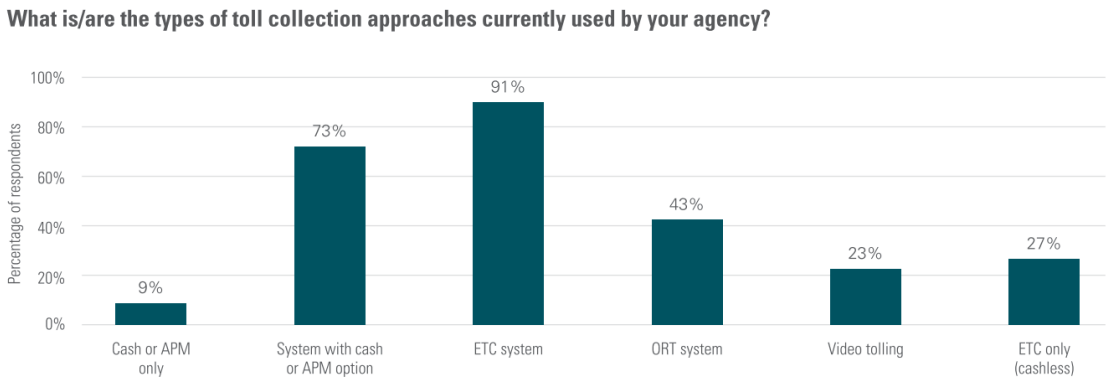


Figure 2.1: Payment methods used by various tolling agencies, according to KPMG International (2015). (Retrieved from [20]).

As demonstrated by Figure 2.1, the collection of tolling fees can be done in several different ways. For instance, payment may be manual, via cash or automatic payment methods (APM). However, this approach is time-consuming, since it requires the customer to stop in order to make the payment. Therefore, another possibility lies in the use of on-board units (OBUs) and other intelligent systems, which allow the registered vehicles carrying them to be detected through a signal emitted by the unit. The fee is assigned as a transaction to the user the vehicle is registered to. Although this approach allows for free transit, it also increases the possibility of violations, since a transgressing vehicle may not be equipped with an OBU [21].

Electronic toll collection (ETC) systems are not, however, limited to the use of on-board units, relying on a complex set of systems depicted in Figure 2.2 [19].

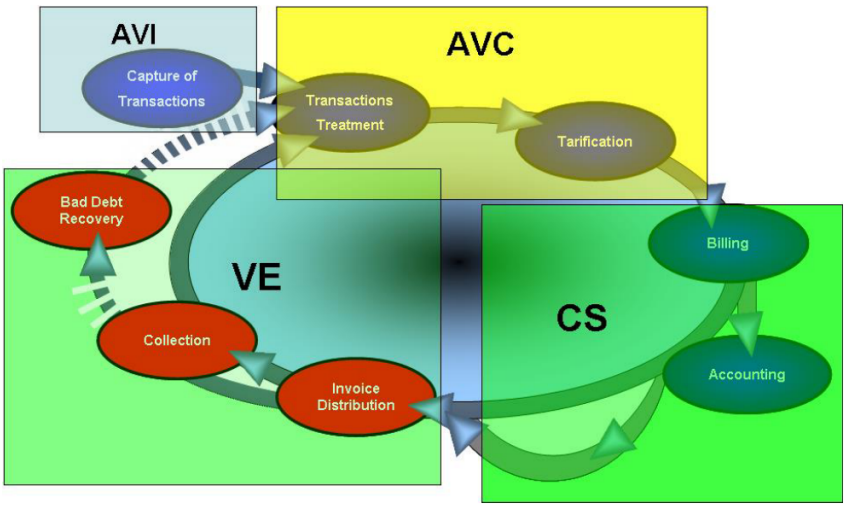


Figure 2.2: Components of an ETC system according to 4icom and Steer Davis Gleave (2015): automatic vehicle identification (AVI), automated vehicle classification (AVC), customer service (CS) and violation enforcement (VE). (Retrieved from [19]).

According to an 4icom and Steer Davis Gleave report published in 2015, from which Figure 2.2 was retrieved, ETC components can be summarised as follows [19]:

- Automatic vehicle identification (AVI): transmission of an identification code between an in-vehicle device, such as an OBU, and some sort of road-side equipment, or RSE.

- Automated vehicle classification (AVC): determination of the appropriate vehicle classification for toll collection, through examination of factors such as the number of axles, vehicle length or the height above the front axle.
- Customer service (CS): includes tasks such as the creation and maintenance of customer accounts, issuing transponders, or accounting.
- Violation Enforcement (VE): enforcing an acceptable level of compliance through the following key elements:
  - License plate image capture
  - Acquisition of the name and address of the violating party
  - Violator payment
  - Legal system interfaces (in countries where toll violations are a citable offence)

Video tolling involves the capture of images of vehicle license plates and their recognition, being also dubbed Automatic Number Place Recognition (ANPR) or Automatic License Plate Recognition (ALPR). ALPR is used in violation enforcement, both in ensuring a vehicle has paid their toll and in pursuing non-payers. This system makes use of optical character recognition (OCR) software, which scans the image and identifies the characters present in the plate. The plate number is then run against existing databases in order to charge the corresponding vehicle or identify the offender. This approach allows for the implementation of open-road tolling (ORT) strategies [4]. However, this approach is dependent on the capacity of the system to accurately identify the license plate number. As of 2015, the recognition rate of multiple European operators was of about 85% [19].

2.2 Machine Learning

Machine learning (ML) is a field of study that involves adaptive mechanisms that enable computers or other machines to learn from experience, by example and by analogy. This kind of system can, by design and with the right algorithms, improve its learning capabilities over time [22]. ML is organised around three main foci: task-oriented studies, cognitive simulation and theoretical analysis [23]. Every instance in a dataset used by a ML algorithm consists in a same set of features, which can be continuous, categorical or binary [24].

ML can be divided mainly into supervised and unsupervised learning. Supervised learning requires the existence of a "teacher", normally in the form of data labels corresponding to what is considered the correct and desirable output. On the other hand, unsupervised data emulates human brain activity more closely, finding patterns in the data by itself with no external help [22]. One other form of ML is reinforcement learning, in which an agent must learn its behaviour through trial and error interactions with a dynamic environment. This environment has a set of pre-defined states it can assume, similarly to the agent, which can perform an action belonging to a set of predefined actions. Besides these two sets that constrain and define the system's behaviour, there is also a reinforcement signal, commonly discrete and either binary or integer, that represents a reward (positive) or penalty (non-positive) for an action taken [25]. This third type of ML falls outside the scope of the present work and will therefore not be further detailed.

2.2.1 Supervised learning

Supervised learning (SML), which may also be called inductive ML, consists in a process of learning a set of rules from instances, the examples in a training set, creating a classifier that can then generalise from new instances [24]. In this form of ML, the machine is shown an instance and outputs a vector of scores, one for each category. The class with the highest score is then considered to be the predicted one. The training process involves multiple iterations over this output, by

computing a loss function that measures the error between the desired and obtained outputs for each instance in the training set, which is optimised by the machine through the modification of a set of internal adjustable parameters [26].

SML can be subdivided into the following categories [24, 27, 28]:

- **Regression**: used when the label being predicted is a value in a continuous spectrum.
- **Classification**: applied when the data is used to predict a category, dealing, therefore, with discrete values. Can be divided into binary or two-class classification (two categories) or multi-class classification (more than two categories).
- **Anomaly detection**: utilised to detect unusual data points. Applicable in contexts such as fraud detection or security.

2.2.2 Unsupervised learning

Unsupervised learning (UspML) refers to a type of machine learning where the desired output class is unknown. Therefore, these kind of algorithms aim to analyse the data and identify similar patterns that can lead to grouping into classes [29].

UspML mainly consists of clustering and dimension reduction algorithms [24, 29]:

- **Clustering**: divides the input dataset into logical groups of related items.
- **Dimension reduction**: simplifies a large input dataset by mapping it to a lower dimensional space.

2.2.3 Deep learning

Conventional ML techniques are limited in their capability of processing data in raw form, necessitating the emergence of new forms of learning. Representation learning is one such form, in which a machine fed with raw data can automatically discover the representations needed for detection or classification [26]. Deep learning is a subcategory of ML representation learning that focuses on learning features from complex sets of input data. This field focuses on the alliance of machine learning with the general purpose of artificial intelligence through the use of algorithms called neural networks [30–32].

Convolutional neural networks (CNN) are a type of neural network model that possesses a high learning capacity and the ability to incorporate large amounts of prior knowledge about the domain being studied [33]. These structures draw inspiration from the visual cortex, which shows specialisation of cell groups in regard to specific regions of the visual field, with these neurons firing when they perceive a stimuli in the region they are mapped to [34]. To further this analogy, the basic components of a convolutional neural network are also named neurons, and these CNNs use different strategies to determine neuron activation, called activation functions. Likewise, each neuron takes a number of binary inputs, producing a single binary output [35].

CNNs constitute an approach that aims to solve the problems posed in classifying images which vary vastly. These networks model smaller pieces of information in layers, which are then combined and processed to obtain a final result. The first layer applies edge detection, with subsequent layers using this information as template to extract further details, such as shapes, or to model different scales or positions. The last layers match the input images with the templates, making the final output a weighed sum of the intermediary outputs [32].

A particular characteristic of convolutional neural networks lies in the distribution of its hidden layers, which are set as three-dimensional, with height, width and depth. Therefore, neurons belonging to a layer connect only to a subset of neurons in the next one. The purpose of these hidden layers lies in feature extraction through convolution and pooling layers. A final, fully connected layer is then responsible for the final output and classification. Convolution layers

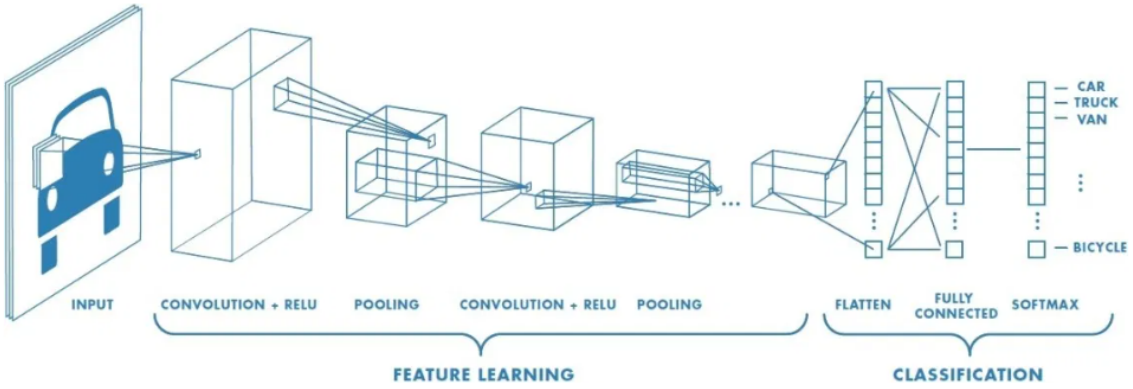


Figure 2.3: General example of the structure of a CNN according to Serengil (2017). (Retrieved from [36].)

are composed by kernels or filters, square matrices on which matrix multiplication is performed alongside the given input matrix. The kernels are applied in steps or strides, which determine the pixel distance between kernel applications as described in Figure 2.4. Padding is also usually added to the extracted feature map in order to prevent it from shrinking [32, 37].

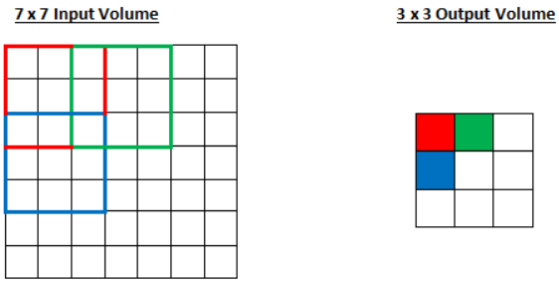


Figure 2.4: Convolution of a 7 x 7 kernel with stride = 2 as depicted by Deshpande (2016). (Retrieved from [34].)

After each convolution layer, CNNs typically feature a pooling layer, which is responsible for reducing the dimensionality in order to help prevent overfitting. The most common pooling technique consists in keeping the maximum value of each window, thus taking the name *max pooling*. The final feature map is performed by the third type of layer, called fully connected layer, in which neurons have connections to all the activations present in the previous layer, computing the outcome in the same way a regular neural network would [32, 37].

2.2.4 Performance indicators for machine learning

Let us consider a binary classifier which aims to predict the presence or absence of a desired property A. Since the presence of A is a desired trait, that would represent the positive class, whereas the absence of A would be the negative one. However, when the data is run through the classifier, one of four scenarios can occur [38]:

- A was present and the classifier predicted it as being present. Since the positive class was predicted correctly, this scenario is a **true positive** (T_P).
- A was present and the classifier predicted it as being absent. Since the negative class was predicted incorrectly, this scenario is a **false negative** (F_N).

- A was absent and the classifier predicted it as being present. Since the positive class was predicted incorrectly, this scenario is a **false positive** (F_P).
- A was absent and the classifier predicted it as being absent. Since the negative class was predicted correctly, this scenario is a **true negative** (T_N).

These four scenarios will help inform the indicators presented below [39, 40].

Confusion matrix

A confusion matrix, or contingency table, can be informally considered a table of counts, recording the amounts of correctly and incorrectly recognised examples for each class. The table is arranged so that its lines represent the real or expected classes, while the columns represent the classes actually returned by the classifier. Therefore, all the correctly predicted values will be placed in the diagonal of the matrix [39]. A general example, with the positive class represented by 1 and the negative one by 0, can be found in Table 2.1.

Table 2.1: General confusion matrix for a binary classifier.

		Predicted	
		1	**0**
Real	**1**	T_P	F_N
	0	F_P	T_N

The confusion matrix for a classifier is a powerful tool from which many metrics can be calculated. Moreover, the real and predicted classes are specific cases of an inter-observer agreement problem, widely studied in statistics. Therefore, in order to properly introduce these problems, two common definitions must also be introduced. Let $N = T_P + F_P + F_N + T_N$ [41, 42]:

- **Bias** is said to occur when two observers differ in their opinion on the frequency of occurrence of a class. The bias index (BI) is then defined as the difference in proportions for the positive class. Ideally, BI = 0, as can be inferred from Equation 2.1.

$$BI = \frac{T_P + F_N}{N} - \frac{T_P + F_P}{N} = \frac{F_N - F_P}{N} \tag{2.1}$$

- **Prevalence** relates to the relative probabilities of the positive and negative classes. The prevalence index (PI) is hence defined as the difference between these probabilities, taking values comprised between -1 (when all the results are T_N) and 1 (when all the results are T_P). This index is defined in Equation 2.2.

$$PI = \frac{T_P - T_N}{N} \tag{2.2}$$

The notions of true/false positive/negatives can only exist if a threshold is taken into account. This threshold or cut-off value (C) decides which values fall into each category of the confusion matrix as described in Table 2.2 [43].

Table 2.2: Effect of a cut-off value C in a classifier's output. (Adapted from [43].)

		Predicted probability	
		above C (1)	**below C (0)**
Real	**1**	T_P	F_N
	0	F_P	T_N

By default, the cut-off value is defined as 0.5, meaning that a classifier will judge a value as belonging to the positive class if the probability returned for that value is greater or equal than 0.5. Therefore, a classifier willing to accept almost any case as being positive would lower its threshold. Likewise, augmenting the value of C would mean the classifier would establish more rigorous patterns for selecting a value as being positive, and therefore the number of negative classifications would increase [44].

Accuracy and error rate

Accuracy is the most generic and intuitive way of comparing algorithms. Also called classification rate, this indicator measures the ratio of correct predictions, as defined in Equation 2.3 [45].

$$ACC = \frac{T_P + T_N}{T_P + F_P + F_N + T_N} \tag{2.3}$$

Although high accuracy can mean a classifier is performing well, this evaluation can only be trusted when all the other parameters of comparison are the same [46]. In fact, this metric is highly susceptible to bias and prevalence, being unable to distinguish between the number of correct labels for different classes [39, 47]. This indicator is sometimes indirectly alluded to through another one, called error rate. The error rate, or misclassification rate, is a measure of how much a classifier fails, being defined in Equation 2.4 [48]. This value may be contrasted with the error rate the classifier would assume if it were to always predict the negative class as a baseline, called null error rate. However, this comparison does not always predict the best classifier [49].

$$ER = \frac{F_P + F_N}{T_P + F_P + F_N + T_N} = 1 - ACC \tag{2.4}$$

Precision, recall, precision/recall curve and F-score

Precision is an estimate of the predictive value of a label, being usually defined for the positive values, and can be regarded as a measure of exactness [39, 50]. Sometimes titled confidence or positive predictive value (PPV), this metric addresses the amount of values predicted as positive that actually belong to the positive class, as defined in Equation 2.5 [47].

$$PRC = \frac{T_P}{T_P + F_P} \tag{2.5}$$

By definition, precision focuses only on analysing positive cases, ignoring the negative ones altogether. Therefore, this metric alone does not suffice to characterise a classifier. Therefore, precision is usually paired with a second metric, called recall, true positive rate (TPR) or sensitivity, which is defined in Equation 2.6 [47].

$$REC = \frac{T_P}{T_P + F_N} \tag{2.6}$$

Recall is a measure of correctly classified and misclassified positive examples (T_P and F_N, respectively), being a measure of completeness [39, 50]. In practice, this performance indicator assesses the amount of actual positive values that were predicted as being positive. However, similarly to what happens with precision, this metric does not provide measurements for the negative values [39]. In fact, precision and recall are both unable to detect a change in T_N if all the other elements in the confusion matrix remain the same. This constitutes a problem when the negative class is also important and well-defined (for instance, a binary classification between male and female may see either class considered as the positive one). Therefore, Sokolova and Lapalme (2009) believe that "an appropriate evaluation measure should take into account the classification of negative examples and reflect the changes in T_N when the other matrix elements stay the same" [40].

One attempt to weigh precision and score as a whole is the definition of a precision/recall curve [44, 50]. This plot takes into account not only the intuitive inverse relationship the two metrics share, but the fact that precision is sensitive to the way the data is distributed. Therefore, by plotting the two variables against each other and comparing them with a baseline determined by the dataset distribution ($y = \frac{P}{P+N}$), some information can be gained about a classifier's performance [44, 51]. The main goal of a precision/recall curve is to reside in the upper right hand of its space. Nevertheless, and although these curves provide visual output, comparing two curves is not easy, and so this method is best used when complemented by others [51]. Saito and Rehmsmeier (2017) illustrated how generic perfect curve for a 1:1 positive to negative ratio should look like (Figure 2.5 (a)) and the comparison between two more realistic precision/recall curves (Figure 2.5 (b)), where the comparison between two curves is relatively easy, since classifier A (in red) clearly outperforms classifier B (in blue) [52].

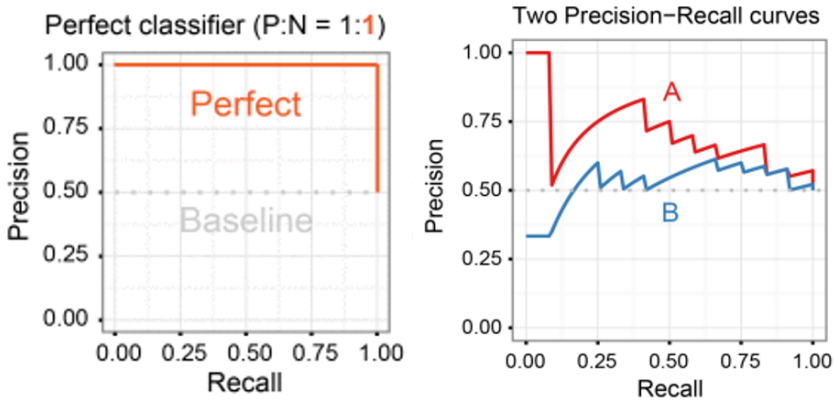


Figure 2.5: Precision/recall curve samples by Saito and Rehmsmeier (2017). (a) Ideal curve for a 1:1 positive to negative ratio. (b) Comparison between two classifiers where classifier A (**red** curve) outperforms classifier B (**blue** curve). (Adapted from [52]).

It should be noted that this curve is threshold-free, being plotted across a range of cut-off values and showing model-wide behaviour patterns. Moreover, intermediate values between two points A and B pertaining to this curve cannot be obtained through linear interpolation, needing to follow Equation 2.7 [44, 52].

$$y = \frac{T_{P_A} + x}{T_{P_A} + x + F_{P_A} + \frac{F_{P_B} - FP_A}{T_{P_B} - T_{P_A}} x}, \quad 0 \leq x \leq |T_{P_B} - T_{P_A}| \tag{2.7}$$

Another way of assessing precision and recall is by computing a new indicator. The F-score or F-measure is a harmonic mean of precision and recall that attempts to unite both indicators into a sole one that attempts to translate the relationship between the positive labels and the labels predicted by a classifier [40]. Since different models might benefit more from one of the indicators than the other, this harmonic mean may be weighted by a parameter β such that $\beta \geq 0$. This general formula for the F-score is described in Equation 2.8 [53].

$$F\text{-score } = (1 + \beta^2) \frac{PRC \times REC}{\beta^2 PRC + REC} = \frac{(1 + \beta^2) T_P}{(1 + \beta^2) T_P + \beta^2 F_N + F_P} \tag{2.8}$$

Depending on the chosen β, this score may favour either metric as follows: [53]

- $\beta < 1$: F becomes precision-oriented.
- $\beta > 1$: F becomes recall-oriented.
- $\beta = 1$: F is the exact harmonic mean between precision and recall, with both contributing equally. In this case, F takes the special denomination F_1 score, defined in Equation 2.9. This is the most widely used form of F-score in the literature.

$$F_1 \text{ score } = 2 \frac{PRC \times REC}{PRC + REC} \tag{2.9}$$

The F_1 score is a proportion of specific agreement for the positive class [47]. Since it is a similarity measure, its ideal value is 1, denoting maximum similarity. In contrast, an F_1 score of 0 indicates complete dissonance between real and predicted values. However, despite its being a metric indicator, it has a number of disadvantages. For instance, it inherits the flaws inherent to accuracy, precision and recall in their susceptibility to bias, focuses only on one class and doesn't average well in real situations. F-scores also assume that the real and predicted sets are identically distributed [54].

Sensitivity, specificity and ROC curve

As stated above, a classifier's recall can also be called sensitivity or true positive rate (TPR). These two alternative denominations are usually employed when this measure is used in conjunction with another, called specificity. This indicator measures how effectively a classifier identifies negative labels, being synonymous with a classifier's true negative rate (TNR) and defined as seen in Equation 2.10. Conversely, the false positive rate (FPR or fallout) can also be defined as shown in Equation 2.11 [40].

$$SPC = \frac{T_N}{T_N + F_P} \tag{2.10}$$

$$FPR = \frac{F_P}{T_N + F_P} = 1 - SPC \tag{2.11}$$

The receiver operating characteristic (ROC) curve describes the inherent detection characteristics of a classifier by letting the receiver of the test information operate at any point of the curve

through the choice of an appropriate threshold, hence its name [46]. This curve is built by plotting the false positive rate (FPR) along the x-axis and the TPR along the y-axis [55]. Therefore, this plot shows the trade-off between sensitivity and specificity, as shown in Figure 2.6 [44].

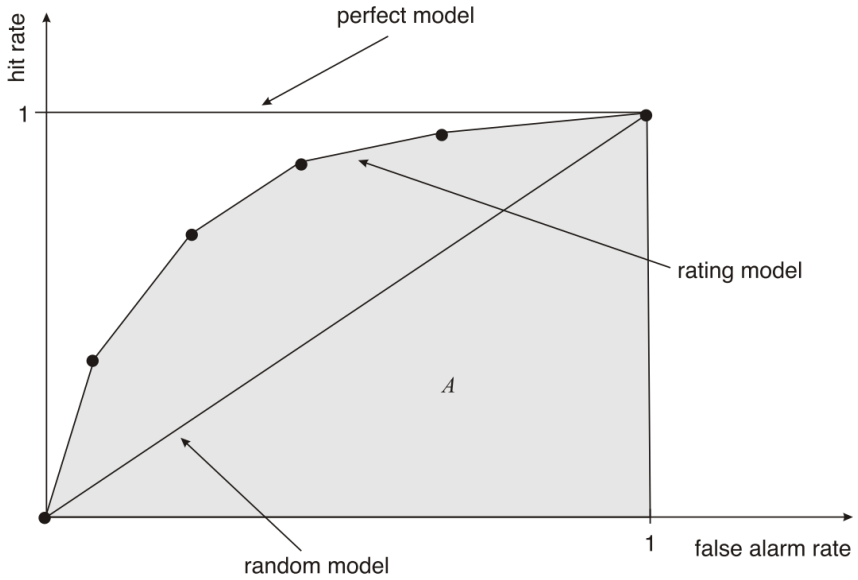


Figure 2.6: ROC curve as explained by Engelman et al. (2013). The term *hit rate* is synonymous with true positive rate and *false alarm rate* is the same as the false positive rate. (Retrieved from [43]).

As Figure 2.6 shows, ROC curves have a distinctive baseline in the $y = x$ axis, corresponding to the performance of a random or naïve classifier. [43] Conversely, an ideal classifier would pass through the point $(0, 1)$, indicating the absence of misclassification [56]. Nevertheless, any ROC curve must pass through two specific points. The first, at $(0, 0)$, corresponds to the case where the cut-off value is highest and all cases are considered negative. On the other hand, these curves also include $(1, 1)$, since setting the threshold at its lowest level will cause all the values to be considered positive [46].

A property of ROC curves is their insensitivity to class distribution [56]. This means that, unlike precision/recall curves, these curves may overestimate the performance in skewed datasets. Another difference is the interpolation between values, which is linear for ROC curves [44].

The area under a ROC curve (AUC) can be translated into the classifier's ability to avoid false classification [40]. ROC AUC can also be interpreted as the average power of the tests corresponding to all possible cut-off values C, being is 0.5 for a random model with no discriminative power and 1.0 for a perfect model [43]. The area under the curve is most easily estimated by using trapezoidal integration [57, 58].

Cohen's kappa and inter-rater agreement

The kappa (or κ) statistic was introduced by Cohen (1960) to measure the degree of concordance between two raters in a binary classification task, taking into account agreement expected by chance [42, 59]. When translated into machine learning terminology, besides the observed concordance (p_0) given by a classifier's accuracy, a second, more intricate term p_e is introduced to account for chance agreement. Numerous ways of estimating p_e are described in the literature, generating different notions for the kappa statistic [41]. However, Cohen assumed that the disagreement between different categories would be equally weighted, with the order they were

presented in being irrelevant [42]. Moreover, these categories would have different distributions that could be estimated as the products of the corresponding marginal coefficients of the confusion matrix [59]. Cohen's kappa can hence be described by Equation 2.12 [60].

$$\kappa = \frac{p_0 - p_e}{1 - p_e} \tag{2.12}$$

where

$$p_0 = ACC \text{ and } p_e = p_{POS} + p_{NEG}$$
$$p_{POS} = \frac{(T_P + F_N)}{T_P + F_P + F_N + T_N} \times \frac{(T_P + F_P)}{T_P + F_P + F_N + T_N}$$
$$p_{NEG} = \frac{(T_N + F_P)}{T_P + F_P + F_N + T_N} \times \frac{(T_N + F_N)}{T_P + F_P + F_N + T_N}$$

The Cohen's kappa score agreement can be analysed as per Landis and Koch (1977) [60]:

- $\kappa \leq 0.00$: Poor
- $0.01 \leq \kappa \leq 0.20$: Slight
- $0.21 \leq \kappa \leq 0.40$: Fair
- $0.41 \leq \kappa \leq 0.60$: Moderate
- $0.61 \leq \kappa \leq 0.80$: Substantial
- $0.81 \leq \kappa \leq 1.00$: Almost perfect

However, the kappa statistic has some inherent flaws. Feinstein and Cicchetti (1990) identified two problems they termed as "paradoxes". Firstly, high values of concordance can yield low values of κ. Moreover, asymmetric and imbalanced tables tend to have a higher κ than perfectly balanced and symmetric ones [42]. These two paradoxes occur because the statistic is affected by both bias and prevalence issues [41]. Kappa can hence be adjusted for prevalence and bias as described in Equation 2.13, taking the name PABAK (prevalence-adjusted bias-adjusted kappa) or weighted relative accuracy [47].

$$PABAK = 2 \times p_0 - 1 = 2 \times ACC - 1 \tag{2.13}$$

Moreover, Cohen's kappa score can also be influenced by the maximum κ achievable for the classifier, described in Equation 2.14. This value (κ_{max}) can be translated into the extent to which the raters' ability to agree is constrained by pre-existing factors that result in unequal marginal totals and may explain low κ scores due to these marginal totals instead of them being directly caused by low concordance [42].

$$\kappa_{max} = \frac{p_{0_{min}} - p_e}{1 - p_e} \tag{2.14}$$

where

$$p_{0_{min}} = min(\frac{T_P + F_P}{N}, \frac{T_P + F_N}{N}) + min(\frac{T_N + F_N}{N}, \frac{T_N + F_P}{N})$$

Another, albeit less generic, statistic for inter-rater agreement is Matthews correlation coefficient (MCC), which is calculated from all four values of the confusion matrix [44]. Varying

between -1 and +1, this coefficient measures the way normalised variables tend to have the same sign and magnitude. MCC applies to binary classification only, being described in Equation 2.15 [61].

$$MCC = \frac{T_P \times T_N - F_P \times F_N}{\sqrt{(T_P + F_P) \times (T_P + F_N) \times (T_N + F_P) \times (T_N + F_N)}} \tag{2.15}$$

Root mean square error and mean average error

Classifier evaluation is not solely dependent on confusion matrix-based metrics. One of the most widely used error measures in academia is the root mean square error, or RMSE. This indicator is a dimensioned measure, expressing average prediction error in the units of the variable of interest [62, 63]. The squaring of each individual error is a means to remove the sign and take only the magnitude of the values into account, as shown in Equation 2.16 [62].

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(x_{i_{real}} - x_{i_{predicted}})^2}{N}} \tag{2.16}$$

An alternative to the use of RMSE is the mean average error, or MAE, which is conceptually simpler to understand, since it averages the unaltered magnitude of each error and does not take the root of N as a factor [62]. This metric is displayed in Equation 2.17 [62, 63].

$$MAE = \frac{\sum_{i=1}^{N} |x_{i_{real}} - x_{i_{predicted}}|}{N} \tag{2.17}$$

2.3 Computer Vision

Computer vision (CV) is a field within computer science that can be defined as the process of using machines to understand and analyse images. Unlike computer photography, the contents of an image are of paramount importance, with the focus residing not the image itself, but on the interpretation of its context. The main objectives for CV can be described on a biological and engineering level, since this field aims not only to establish computational models of the human visual system, but also to build autonomous systems that either perform the same tasks as this system, or surpass its capabilities [64, 65]. This technology implements deep learning techniques which aid in building functions such as image classification, object detection, tracking and image manipulation. However, CV still faces a number of complex challenges nowadays, such as differing light conditions, zoom noise, image angles and motion, background clutter or even the existence of variations of the same object, such as different types of cars. In fact, computer vision plays a considerable role in the automotive industry, both from devices installed in the vehicles themselves (such as sensors and cameras), as well as from external sources, which can help ensure the vehicle's compliance with parking or tolling situations [31].

2.3.1 Image fingerprinting

The distinction of digital content is a major concern when it comes to copyright laws, in order to detect improper or unsanctioned use of said content. In order to identify images, two main approaches have been considered: watermarking, which means adding information to the content of the object in order to identify it, and fingerprinting, or extracting content-based signatures from

an object [66]. Drawing a parallel to how human fingerprints uniquely identify an individual, Boneh and Shaw (1995) first attempted to define digital fingerprints using a terminology that defined a mark as a bit of information encoded in an object, and a fingerprint as a collection of such marks. [67]. Retrieving a signature from an image can be achieved by using techniques similar to the concept of locality-sensitive hashing (LSH), which differs from normal hashing in that similar items will have similar hashes [66, 68].

Image content fingerprinting is closely associated with the field of object recognition, aiming to identify image duplicates. In order for this identification to occur, image features have to be identified. These features correspond to properties or attributes of an image. Image features can be classified into three main categories: edge, corner and region features [69, 70].

2.3.2 Edge detection

Edges are a section of an image that represent information about its contours, being therefore one of the principal contributors of information about an image. An edge is defined as a change in the colour or brightness or an oriented localized change in intensity of an image, and can belong to one of four types (step edge, ramp edge, roof edge or line edge, sometimes also called bar edge), as described in Figure 2.7. [71–73].

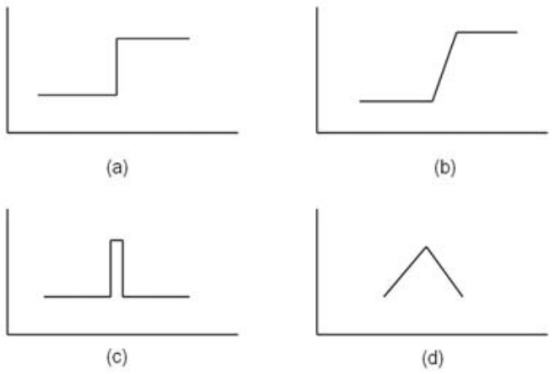


Figure 2.7: Edge types as described by Senthilkumaran and Rajesh (2009). (a) Step edge (b) Ramp edge (c) Line edge (d) Roof edge. (Retrieved from [72]).

Edge detection (ED) is a process that simplifies the analysis of images by reducing the amount of data to be processed, while at the same time preserving useful structural information about object boundaries [74]. ED is often the first step undertaken in image segmentation, transforming images into simpler edge images by taking advantage of changes of grey tones in the original ones [72].

ED consists of three main steps [73]:

1. **Filtering**: in the filtering stage, a filter is applied in order to remove or reduce noise in the image [73]. It should be noted, however, that there is a trade-off between edge strength and noise reduction, since more filtering results in loss of strength [72].
2. **Enhancement**: this step aims to improve the quality of the digital image, sharpening it [73]. The purpose of this sharpening is the facilitation of the process of determining changes in intensity in the neighbourhood of a point, emphasising pixels where there appears to be a significant change. This stage is usually performed by computing the gradient magnitude [72].
3. **Detection**: this phase's objective is to determine which of the obtained points are actually edge points, since many points in an image have a non-zero value for the gradient, and not

all are edge points. Thresholding is frequently used for detection [72, 73].

Moreover, ED can be subdivided into two main types: gradient-based and Laplacian-based [71]. Gradient-based edge detection works by looking for the maximum and minimum in the first derivative. Examples of use of this method include the Sobel, Prewitt and Roberts edge detectors [75]. Laplacian-based ED, however, takes this one step further, detecting edges by calculating zero-crossings in the second derivative of the image in order to further highlight edge locations [76]. This type of detector, which is sometimes called Laplacian of Gaussian (LOG) includes the Marr-Hildreth and Canny edge detectors, the last of which is the most commonly used edge detection method [75].

Canny edge detection

Canny edge detection (CED) was primarily proposed by J. Canny (1986), who defined a detection operator based on the maxima of Gaussian-smoothed images [74]. Canny assumed the edge to be detected as being a step edge subject to white Gaussian noise [50]. The Gaussian filter used by CED works by eliminating noise from the image, enabling clearer edge detection [74].

In his original work, Canny defined three main criteria for CED that aimed to posit it as an improvement over other ED algorithms. The first and most obvious criterion considered was a low error rate, minimising the responses to non-edges and maximising the detection of actual edges. Secondly, Canny intended that his improved detection method have good edge localisation, meaning that the distance between the pixels considered as edge pixels and the edge itself should be minimal. The last criterion applied was single response, in that there should be one and only one response for an edge [74, 76, 77].

The CED algorithm comprises the following steps [73, 75, 77]:

1. **Smoothing the image**: this first stage is accomplished with the application of a Gaussian filter.
2. **Computing the magnitude**: the next step consists of finding the gradient magnitude and orientation.
3. **Applying non-maximal suppression**: the third step takes the gradient magnitude and applies non-maximal suppression to it, hence converting all the points that are not maxima to zero.
4. **Thresholding**: next, the hysteresis thresholding algorithm is used to detect and link edges to one another. Hysteresis is a method that makes use of two thresholds, one high and the other low.
5. **Edge tracking**: finally, the final edges are determined by suppressing all edges that are not connected to a very certain and strong edge [73, 76].

Sobel edge detection

The Sobel edge detection method consists of a pair of 3 x 3 convolution kernels, which differ by a rotation of 90°, as shown in Equation 2.18 [76]:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.18}$$

The kernels represented in Equation 2.18 respond maximally to edges that run either vertically or horizontally to the pixel grid. These masks are slid over the image, with calculations being made against a square of pixels at a time in order to calculate the gradient of the image in each point. Edges are then identified as the areas that represent strong intensity contrasts [73, 76].

2.3.3 Corner detection

Corners are one of the most intuitive types of feature point, being image points that show a strong 2D intensity change and can thus be easily distinguished from neighbouring points [78]. Corner detection is used as the first step of many vision tasks such as SLAM (simultaneous localisation and mapping), tracking, localisation, image matching and recognition [79]. The detection of these localised isolated events must obey three main criteria, listed below [78, 80]. It should be noted that the first criterion gives a measure of detection, whilst the second provides a way to determine localisation, which makes them conflicting since good detection can lead to poor localisation [80].

1. **Consistency**: if an algorithm seeks to use corners as features, it must detect them consistently. Detected positions should be insensitive to the variation of noise, nor should they move when multiple images are acquired.
2. **Accuracy**: corners should be located precisely or as close as possible to their correct position, since errors in location are magnified in subsequent steps.
3. **Complexity**: real-time jobs require speed, which favours corner detection algorithms with reduced algorithmic complexity.

Feature point and corner detectors can be divided into three main groups: contour-based, intensity-based and parametric [78, 81]. Contour-based methods start by extracting contours and then searching for their maximal curvature or inflexion points, or perform a polygonal approximation and search for intersection points [81]. These algorithms consist of three cascaded basic blocks: edge detection, contour extraction and corner decision on those contours [82]. On the other hand, parametric or model-based methods fit a parametric intensity model to the signal, often providing sub-pixel accuracy but being of limited use [81]. These algorithms find corners by matching image patches to predefined models, which makes it difficult to cover all the corners present in real images [82].

Intensity-based methods compute a measure that indicates the presence of an interest point directly from the grey-values [81]. These methods exploit an intuitive definition of a corner as a point on the boundary of two image regions where the curvature is sufficiently high. Therefore, the "cornerness" of a region was defined by Kitchen and Rosenfeld (1982) as a product of the magnitude of the gradient of image intensity and the rate of change of gradient direction at the considered point [78].

The majority of feature detection algorithms work by computing a corner response function across the image. Pixels which exceed a threshold "cornerness" value and are local maxima are retained while the rest are discarded [79].

Moravec's detector and Harris corners

In 1980, Moravec defined the notion of points of interest as being those where there was a large intensity variation in every direction. [78] The author then introduced an interest operator for corner detection, since he considered corners to be regions of high contrast in orthogonal directions, which made for good features. [83] This interest operator functions by considering a local window in the image and determining the average changes in intensity that result from shifting that window slightly in different directions[84].

Moravec's algorithm uses four discrete shifts in directions parallel to the rows and columns of the image, detecting an interest point if the minimum is superior to a threshold [81]. However, since this variation is computed along only four directions, this operator is sensitive to strong edges under certain directions [78].

Harris and Stephens (1988) introduced a corner operator by modifying Moravec's interest operator, using only first order derivatives to approximate the second derivatives used in the original

detection method [80]. For each pixel location, these authors compute a 2×2 auto-correlation matrix $\mathcal{M}$ with eigenvalues α and β proportional to the principal curvatures of the local autocorrelation function. Therefore, three different scenarios can occur [78, 84]:

1. Both curvatures are small: the local autocorrelation function is flat, and so the windowed image is flat.
   Parameters: α and β are small.
2. One curvature is high and the other low: the local autocorrelation function is ridge-shaped, indicating the presence of an edge in the image.
   Parameters: $\alpha \gg \beta$ or $\beta \gg \alpha$.
3. Both curvatures are high: the local autocorrelation function is sharply peaked, indicating a corner.
   Parameters: α and β are large, with $\alpha \sim \beta$.

A graphic depiction of curvatures in Harris corners according to the parameters α and β can be found in Figure 2.8 [84, 85].

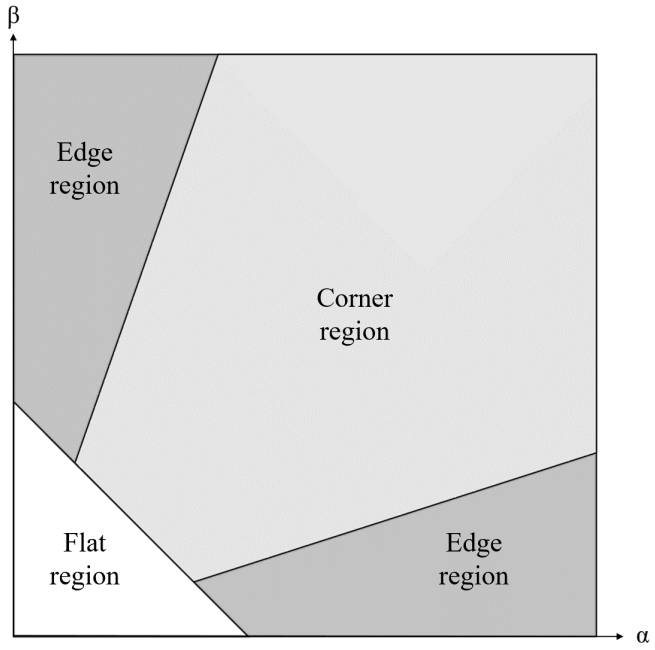


Figure 2.8: Three possible scenarios for curvatures in Harris corners (1988). (Adapted from [84, 85]).

Given the eigenvalues α and β, the authors defined the Harris operator for corner response, described in Equation 2.19. The constant parameter k was empirically determined to be optimal when equal to 0.04, although values in the $[0.04, 0.06]$ range were deemed acceptable [82, 84]. A commonly used variant for the Harris corner response operator is also described in Equation 2.20 [85].

$$R = \alpha\beta - k(\alpha + \beta)^2, \text{ with } k \in [0.04, 0.06] \tag{2.19}$$

$$R = \frac{\alpha\beta}{\alpha + \beta} \tag{2.20}$$

In order to categorise a given point as being a corner, Harris and Stephens then consider local maxima. A corner region pixel is considered as effectively being a corner if it has a positive response R which is an eight-way local maximum. Similarly, an edge region pixel is classified as an edge if R is both a local minimum and negative in either the x or y directions, according to which gradient is larger. The authors also applied thresholds in order to perform hysteresis to enhance the continuity of edges [84].

FAST

The features from accelerated segment test (FAST) algorithm was developed by Rosten and Drummond (2006) to identify interest points in an image, being is ten times faster than the Harris corner detector without degrading performance, according to Jeong and Moon [79, 86]. This algorithm works by examining a small patch of an image to see if it resembles a corner. For that purpose, the authors built a training set where all edges met at a multiple of 45° and near the centre of an 8×8 window, thus determining the kind of corner features to be detected [79].

The segment test criterion for FAST considers a set of 16 pixels around the pixel chosen as a corner candidate (p). p is then considered a corner if there are n contiguous pixels in the circle which are either brighter than the intensity of pixel p (I_p) plus a threshold t or darker than $I_p - t$ [86]. The authors chose $n = 12$, since that number admits a high-speed test that improves computation by excluding a large number of non-corners. This test takes pixels 1, 5, 9 and 13, which are placed at the four compass directions as shown in Figure 2.9. In order for p to be a corner, then at least three of these have to be brighter than $I_p + t$ or darker than $I_p - t$. If none of these two conditions apply, p cannot be a corner. However, if it does, the test can then be applied to the remaining pixels. It should be noted that this test does not generalise well for $n < 12$ [79].

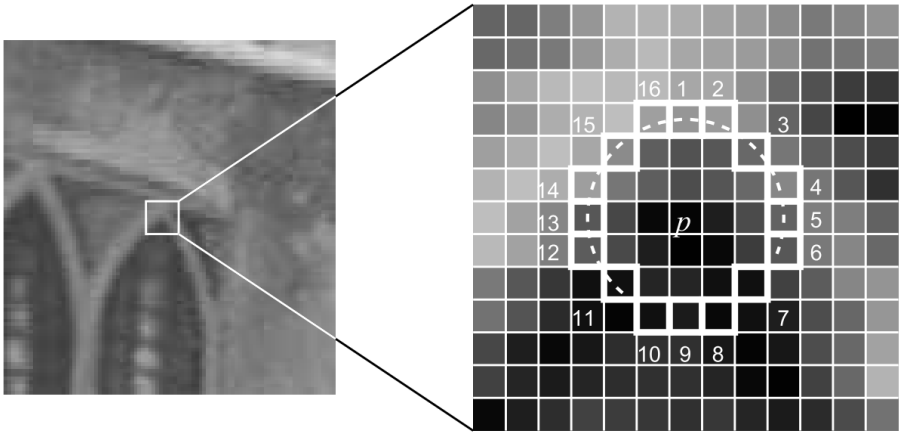


Figure 2.9: Corner detection with FAST using 16 pixels around a pixel p. (Retrieved from [79]).

2.3.4 Image feature descriptors

Feature extraction is the process of computing a representation of some attribute of digital images to derive the image contents, being directly related to the visual characteristics of the image. These representations are usually numerical or alphanumerical and are associated with image regions, with an object within an image being characterised by its own measurements [50, 87].

A type of object within an image may be characterised perfectly by one feature, but if this proves impossible, a similar result may still be achieved through the combined use of multiple features [50]. Among the information used in content-based image retrieval, the most popular

features are colour, texture and shape [88]. Colour is the most intuitive physical characteristic and possesses robustness for some basic image operations, such as rotation, scaling, and translation, making it a popular feature [88]. In turn, texture is defined as a function of the spatial variations in the pixel intensity of an image [89]. This feature is a property of any surface that describes the visual patterns present in it and gives insight into its structure [90]. Shape is a more general concept which pertains to relative positional information. Because of its wide scope, the use of shape features tends to suffer from difficulties such as losing information by mapping the shape to a small amount of numbers or limiting applicability by restricting the descriptors to silhouettes and closed curves [91].

The use of features in lieu of an image's pixels themselves has a number of advantages. For instance, features can act to encode domain knowledge that is difficult to learn using a finite quantity of training data. Moreover, the use of image features helps reduce in-class while increasing out-of-class variability compared to the raw data, making classification easier. A feature-based system also operates much faster than a pixel-based one, causing gains in efficiency [92, 93].

Haar-like features

Wavelets are functions that integrate to zero. Haar wavelets are the simplest kind of wavelet, consisting of a step-function defined in Equation 2.21 and depicted in Figure 2.10 [94, 95].

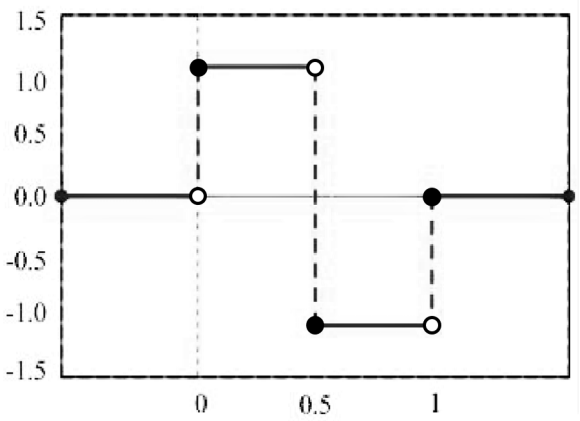


Figure 2.10: Haar wavelet function graph as depicted by Khadtare et al. (2011). (Adapted from [95]).

$$\Psi(t) = \begin{cases} 1, & 0 \leq t < \frac{1}{2} \\ -1, & \frac{1}{2} \leq t < 1 \\ 0, & \text{otherwise} \end{cases} \tag{2.21}$$

Haar-like features (HLF) are an over-complete set of two-dimensional (2D) Haar functions which can be used to encode local appearance of objects, having been proposed by Papageorgiou et al. in 1998, based on theoretical work by Mallat in the previous decade [96–98]. HLF are well-suited to the detection of horizontal, vertical, and symmetric structures and consist of k rectangular regions ($k \geq 2$) enclosed in a template [97, 99].

HLF were explored by Viola and Jones in order to build a cascade classifier. These authors defined three kinds of rectangular Haar-like features with different values based on Papageorgiou's work [92]:

- **Two-rectangle feature**: difference between the sum of the pixels within two rectangular regions (Figure 2.11 A and B).
- **Three-rectangle feature**: sum within two outside rectangles subtracted from the sum in a centre rectangle (Figure 2.11 C).
- **Four-rectangle feature**: difference between diagonal pairs of rectangles (Figure 2.11 D).

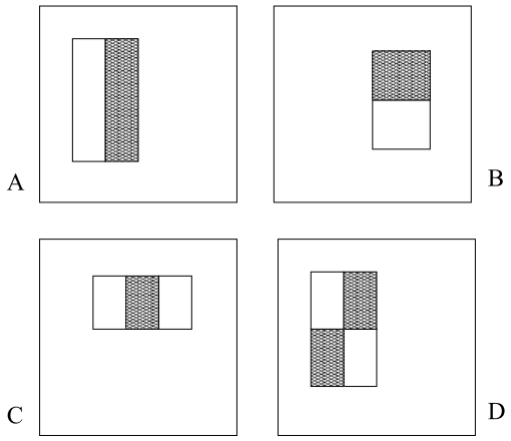


Figure 2.11: Haar features described by Viola and Jones (2001), surrounded by their respective detection windows. A) vertical two-rectangle feature. B) horizontal two-rectangle feature. C) three-rectangle feature. D) four-rectangle feature. Black areas represent negative and white areas positive weights. (Retrieved from [92]).

Lienhart and Maydt build upon the work of Viola and Jones by introducing $45°$-rotated features, which are used by the open-source computer vision library OpenCV, depicted in Figure 2.12 [93, 100]. It is worth noting that the four-rectangle features described by Viola and Jones (Figure 2.11 D) have been removed from the set, since Lienhart and Maydt considered that they could be well approximated using features 2(e) and 2(g) from Figure 2.12 [92, 93].

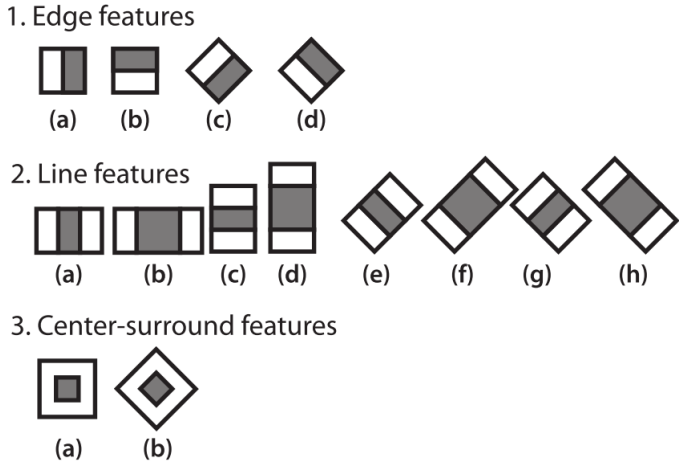


Figure 2.12: Haar features as defined by Lienhart and Maydt and used by OpenCV, according to Bradski and Kaehler (2008). (Retrieved from [100]).

Local binary patterns

The local binary patterns (LBP) operator is a texture descriptor first proposed in 1994 by Ojala et al. [101, 102]. The operator works by labelling pixels through thresholding the 3×3 neighbourhood in relation to the central pixel value, in a total of eight neighbours per pixel. This process is described mathematically in Equation 2.22. Let $P = (x_p, y_p)$ be the central pixel, n the position of a neighbouring pixel relative to P ($n \in \{0, ..., 7\}$), and i_n and i_p the grey levels of the neighbouring and central pixel, respectively [89]:

$$LBP\left(x_p, y_p\right) = \sum_{n=0}^{7} s\left(i_n - i_p\right) 2^n, \quad \text{where } s\left(x\right) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2.22}$$

Equation 2.22 produces a value between 0 and 255 ($\sum_{n=0}^{7} 2^n$), being in fact a two-level version of a three-level texture unit described by Wang and He (1990). This original unit consisted of a 3×3 matrix where the central cell represented the pixel P and the remaining ones were its neighbourhood, akin to what was defined in Ojala et al.'s work. The main difference between the two approaches lies in the number of values each cell can take (called *levels* by the authors), with the original unit assuming values in the $\{0, 1, 2\}$ range and Ojala et al.'s being binary, and thus simplifying the process, since the total number of possibilities for texture units is reduced from $3^8 = 6561$ to $2^8 = 256$ [89, 101, 103]. Each pixel surrounding P is uniquely identified in P's neighbourhood by a number n between 0 and 7, as depicted in Figure 2.13 (a), and its corresponding value in the neighbourhood is given by 2^n (Figures 2.13 (b) and 2.14 (c)).

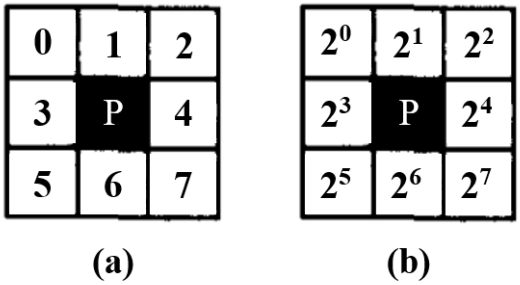


Figure 2.13: Definition of a pixel's neighbourhood in the LBP operator by Ojala el al. (1994). (a) Unique identifiers for each neighbour pixel. (b) Values used by the operator for each neighbour. (Adapted from [101]).

The basic steps undertaken by the operator for each texture unit are described succinctly in Figure 2.14 [102]. After selecting a pixel, a matrix is created with the grey level of that pixel in the central position and the grey level of each neighbour in its due place (Figure 2.14 (a)). Each neighbour is then evaluated according to the $s(x)$ component in Equation 2.22 as follows: if the neighbour's grey level is greater or equal than the central pixel's grey level, its value is set to one; otherwise, the value is set to zero (Figure 2.14 (b)). The resulting values are then multiplied by the intrinsic value of the cell they belong to (depicted in Figures 2.13 (b) and 2.14 (c)). The values resulting matrix shown in Figure 2.14 (d) are then added together in order to produce the final result (in this case, 169, represented in binary as `10101001`) that will be assigned to pixel P.

However, the information provided by a single texture unit may not be enough to characterise the amount and spatial structure of local texture. Therefore, Ojala et al. identified the need for the use of complementary feature pairs, which consider joint occurrences of features, improving the classification accuracy [101, 102]. Another way to regard this problem would be a look at the feature histograms. LBP coefficients can be collected in a LBP histogram, which results in a loss of spatial information. A way to mitigate this problem is to divide the image into several smaller blocks and compute a histogram for each one. The result is a descriptor that contains information about features at the global, region and pixel levels [104].

Both LBP and Haar-like features are commonly used in OpenCV cascade classifiers, which are multistage classifiers where the output of a stage is fed to the next stage, discarding non-matching regions along the way. This means that each classifier may be weak, but their combination can

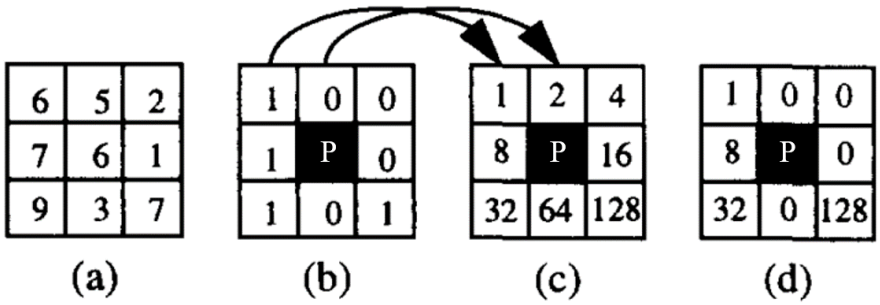


Figure 2.14: Two-level version of LBP as described by Ojala et al. (1994). (a) Original pixel matrix. (b) Evaluation according to pixel P. (c) Intrinsic pixel values. (d) Final result. (Adapted from [102]).

produce fast and strong results even with low computational power. A region is considered a match when it successfully traverses all the assembled stages [105].

SIFT and SURF

The scale invariant feature transform, commonly known as SIFT, was developed by David G. Lowe in 1999 and patented in 2004 [106, 107]. SIFT works by transforming an image into a collection of local feature vectors, each of them invariant to translation, scaling, and rotation, as well as partially invariant to illumination changes and affine or 3D projections [100, 106]. A SIFT descriptor for a keypoint is a 128-dimensional vector created containing 16 orientation sub-histograms, each consisting of 8 bins [106, 108–111].

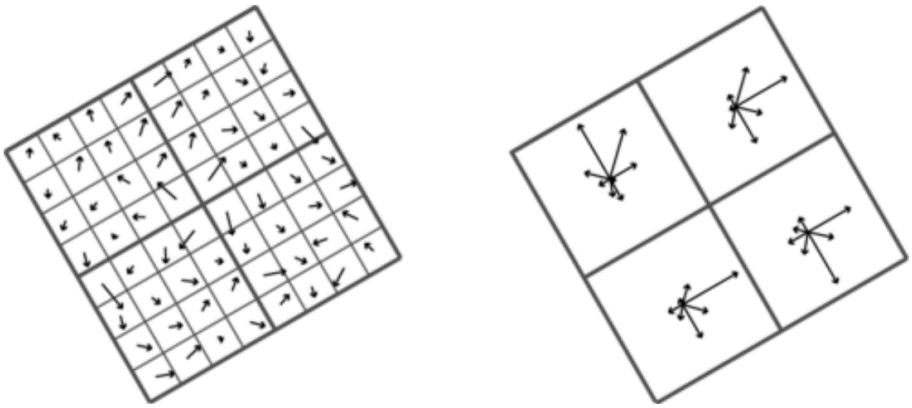


Figure 2.15: Schematic computation of the SIFT descriptor from sampled values of gradient orientation as depicted by Lindeberg (2012). (Retrieved from [112]).

A number of authors have introduced variations to SIFT, namely: [113, 114]

- **PCA-SIFT**: This technique was developed by Ke and Sukthankar in 2004 and introduces principal component analysis (PCA) to reduce the dimensionality of describing each keypoint, making the image feature matching faster [110, 113, 115].
- **GSIFT**: Introduced by Mortensen et al. in 2005, this variation aims to introduce global information instead of relying only on local context. This is done via the introduction of a global texture vector to the basis of SIFT, giving this approach the acronym GSIFT for its use of global features [111, 113].

- **CSIFT**: SIFT was designed with grey-scale images in mind. However, Abdel-Hakim and Farag (2006) considered colour to provide valuable information in object description and matching tasks, which led to their development of colour-invariant SIFT, or CSIFT, which integrates colour invariance into the basis of SIFT [113, 116, 117].
- **ASIFT**: Affine-SIFT or ASIFT was first proposed by Morel and Yu in 2009. ASIFT simulates scale, camera longitude angle, and latitude angle or tilt, normalising translation and rotation [113, 118].
- ***n*-SIFT**: This approach, called n-dimensional SIFT, extends SIFT features to images with an arbitrary number of dimensions, being especially adequate for 3D and 4D medical imagery [114].

A special SIFT derivative is the use of speeded-up robust features (**SURF**). This algorithm was developed by Bay et al. in 2006, being structurally similar to SIFT but adopting different processing methods for each step [113]. An application of SURF was patented in 2012 by its creators [119]. This detector is considered to possess good performance in computation time and accuracy [120]. This algorithm can be described by modified Gaussian functions, coding each pixel point in grey-scale according to the value of the filter at that point: grey for zero, black for negative values and white for positive ones. The cropping is achieved by limiting the mask sizes to $M \times M$ pixels, where M is preferably odd so as to allow the existence of a central pixel. The ratio between M and the standard deviation for the approximated Gaussian function σ is kept constant for all filter sizes. The derivatives are then further approximated by the use of digital filters with discrete sub-regions, shown in the bottom half of Figure 2.16 [119].

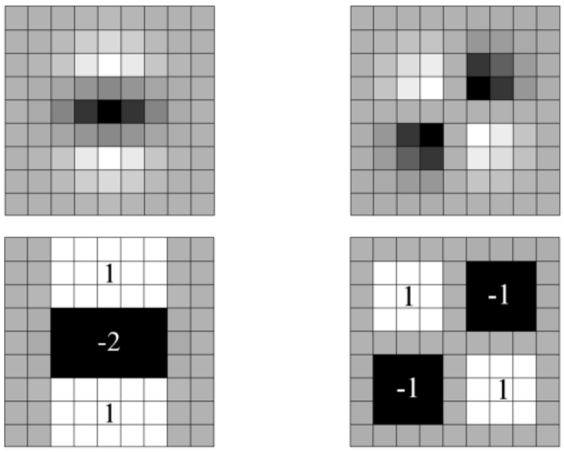


Figure 2.16: Box filter approximation of the Gaussian second-order partial derivatives used in SURF by Bay et al. (2006). (a) Gaussian second order partial derivative in the y direction (above) and Bay et al.'s approximation (below). (b) Gaussian second order partial derivative in the xy direction (above) and Bay et al.'s approximation (below). Both partial derivatives have been discretised and cropped and the grey regions are equal to zero. (Adapted from [120]).

The SURF descriptor is also based on the same basic principles as SIFT's, possessing a lower complexity than the latter [120]. A qualitative summary of the strengths and weaknesses of each SIFT-derived algorithm according to Wu et al. (2013) can be found in Table 2.3. A higher number of + signs indicates better performance, with ++++ being the highest and + being the lowest [113].

Table 2.3: Comparison of performance between SIFT and its derivatives according to Wu et al. (2013). (Adapted from [113].)

	Scale and rotation	**Blur**	**Illumination**	**Affine**	**Time Cost**
SIFT	++++	++	+++	++	+++
PCA-SIFT	+++	+++	+++	++	+++
GSIFT	++	++++	++++	++	+++
CSIFT	++++	+++	++	+++	++
ASIFT	++	+	+	++++	+
SURF	+	+	+	+	++++

Key: ++++ Best +++ Better ++ Good + Common

BRIEF and ORB

While SIFT is highly discriminant, being a 128-dimensional vector (or 128-vector, for short), means it is relatively slow to compute and match. Even approaches such as the one employed in PCA-SIFT, which involves dimensionality-reducing techniques, require that the whole descriptor be computed before having its dimensions reduced, which wastes time and resources. Therefore, Calonder et al. (2011) created a new feature descriptor, called binary robust independent elementary features (BRIEF), that bypasses this first stage, directly computing binary strings from image patches [121]. This algorithm describes the appearance of an image patch p centred on the detected feature point with a binary string computed by comparing the intensities of a number of pair-wise pixels in p. BRIEF defines an individual test τ on a smoothed patch p of size $S \times S$ as described in Equation 2.23, where $p(x)$ is the pixel intensity for patch p at a location $x = (\mathbf{u}, \mathbf{v})^T$ [122].

$$\tau(p; x, y) = \begin{cases} 1, & \text{if } p(x) < p(y) \\ 0, & \text{otherwise} \end{cases} \tag{2.23}$$

A unique set of binary tests can then be defined by choosing a set of n_d (x, y)-location pairs, with the BRIEF descriptor being the n_d-dimensional bit string that corresponds to the decimal counterpart of the value computed in Equation 2.24 [121, 122].

$$f_{n_d}(p) = \sum_{i=1}^{n_d} 2^{i-1} \tau(p; x_i, y_i) \tag{2.24}$$

An alternative to the use of SIFT and SURF was introduced in 2011 by Rublee et al., under the self-explanatory name of Oriented FAST and Rotated BRIEF, or ORB for short. This method uses FAST to locate keypoints, followed by a direction-normalised BRIEF to extract descriptors, therefore combining the fast and accurate orientation component from FAST with the efficient computation of features provided by BRIEF, with modifications to allow for rotational invariance [123, 124].

KAZE and AKAZE

KAZE features are a method for feature detection and description in non-linear scale spaces, having been introduced by Fernández Alcantarilla et al. (2012) [125]. This method uses an edge-preserving non-linear filtering strategy to locate image features, reducing noise [124]. AKAZE

differs from KAZE in that its features are invariant to scale, rotation, and have more distinctiveness at varying scales because of non-linear scale spaces [126].

2.4 Goals

The principal aims of the present work are as follows:

1. Conceive a system architecture, quality model, requirements and their respective priorities, as well as performance measures for a putative system that should be able to:
   (a) Identify a vehicle's manufacturer and model in near-real-time and without disturbing the work of the current system in place.
   (b) Implement a more efficient way of identifying tolling violators, reducing the associated costs related to misidentification of the perpetrators.
   (c) Significantly reduce the number of violations necessitating human intervention and the costs associated with manual image review.
2. Provide proof of concept for the possible workings of such a system and analyse techniques that could be used in its development.

2.5 Project life-cycle

2.5.1 Quality models and requirement prioritisation

Quality models

Quality standards are a defining characteristic of successful and durable projects. In particular, software quality is defined by Pressman as "conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software" or "an effective software process applied in a manner that creates a useful product that provides measurable value for those who produce it and those who use it" [127, 128].

The definition of quality attributes for requirement definition in a software project is not consensual, leading to the creation of various analytical models that define different types of requirements that should be considered in a software project [129].

McCall's quality model Originating in the US military and General Electrics and first presented in 1977, Jim McCall's quality model for defining software requirements attempts to bridge the gap between users and developers by focusing on a number of software quality factors that reflect both the users' views and the developers' priorities [129, 130]. Having started by contemplating 55 quality characteristics or factors, this model then compressed those into a set of 11 main factors: correctness, efficiency, flexibility, integrity, interoperability, maintainability, portability, reliability, reusability, testability and usability [131]. These factors reflect the external view of the software, that of a user, and can be further decomposed into 23 quality criteria that reflect the internal view of the software, as regarded by a developer[1] [130].

McCall's quality model contemplates three main categories or abilities a software product most possess [131]:

- **Product revision**: also defined as a product's ability to undergo changes, this category includes the maintainability, flexibility and testability factors [131, 132].

[1] Definitions for each factor and respective criteria can be found in the Glossary.

- **Product transition**: defined as a product's adaptability to new environments, this category addresses the factors portability, reusability and interoperability [129].
- **Product operations**: reporting to the product's ability to be quickly understood, operated and capable of providing the required results, this category covers the correctness, reliability, efficiency, integrity and usability factors [132].

Boehm's quality model Boehm's quality model was proposed similarly to McCall's model, introducing an additional second set of quality factors in an attempt to automatically and quantitatively evaluate the quality of software [130, 131]. This model was presented according to an hierarchical structure containing three basic software requirements (also called primary uses) at the highest level, which can be broken down into intermediate characteristics, which in turn can be divided into primitive constructs[2] [129, 130, 132].

- **High-level characteristics**: also called primary uses, this set of characteristics is constituted by as-is utility, maintainability and portability.
- **Intermediate-level characteristics**: this level comprises a total of seven characteristics, distributed by each of the three high-level ones as follows:
  - As-is utility characteristics: comprised in this group are reliability, efficiency and human engineering.
  - Maintainability characteristics: the characteristics constituting this group are testability, understandability and modifiability.
  - Portability characteristics: not subdivided at this level.
- **Low-level characteristics**: also called primitive constructs, this group comprises the following distribution of 15 characteristics:
  - Reliability constructs: self-containedness, accuracy, completeness, robustness/integrity and consistency.
  - Efficiency constructs: accountability, device efficiency and accessibility.
  - Human engineering constructs: robustness/integrity, accessibility and communicativeness.
  - Portability constructs: device independence and self-containedness.
  - Testability constructs: accountability, communicativeness, self-descriptiveness and structuredness.
  - Understandability constructs: consistency, structuredness, conciseness and legibility.
  - Modifiability constructs: structuredness and augmentability.

Dromey's quality model R.G. Dromey presented an alternative product-based quality framework in the mid-90s, stating that high-level quality attributes could not be built directly into software [129, 132]. This framework consisted in three separate models: the requirement quality model, the design quality model and the implementation quality model. Focusing on the relationship between quality attributes and their sub-attributes, as well as attempting to connect software product properties with software quality attributes, Dromey established a hierarchical structure that depicts the implementation model at the top, branching into intermediate-level correctness qualities present in the design quality model, which in turn were based on the characteristics present in the requirements model[3] [132].

- **Implementation model**: deals with the final quality of the software product.

[2] Definitions for each characteristic and respective subcharacteristics or constructs can be found in the Glossary.
[3] Definitions for each property and respective attributes can be found in the Glossary.

- **Design model**: lists the high-level product properties correctness, internal, contextual and descriptive.
- **Requirements model**: comprises four sets of quality attributes that must be present for the high-level properties in the design model:
  - Correctness: includes the functionality and reliability software quality attributes.
  - Internal: comprises the quality attributes of maintainability, efficiency and reliability.
  - Contextual: deals with the maintainability, reusability, portability and reliability attributes.
  - Descriptive: addresses the quality attributes maintainability, reusability, portability and usability.

FURPS and FURPS+ quality models The FURPS model was proposed by R.Grady and Hewlett-Packard Co. in 1987, being primarily defined by characteristics that were based on the system requirements, and therefore divided into functional (FR) and non-functional (NFR). The name FURPS is acronym that stands for the following characteristics, some of which can be obtained from sets of others [129, 131]:

- **FR characteristics**:
  - **F**unctionality: the only functional attribute of the lot, functionality is defined as including feature sets, capabilities and security.
- **NFR (or URPS) characteristics**:
  - **U**sability: defined by human factors, overall aesthetics, consistency and the existence of documentation.
  - **R**eliability: includes the frequency and severity of failure, recoverability, predictability, accuracy and MTBF.
  - **P**erformance: defined as the conjunction of processing speed, response time, resource consumption, throughput and efficiency.
  - **S**upportability: includes testability, extensibility, adaptability, maintainability, compatibility, configurability, serviceability, installability, and localisability.

This model has since been built upon by IBM and expanded into the FURPS+ model. FURPS+ includes every characteristic present in FURPS, besides taking into consideration design, implementation, interface and physical constraints (DIIPC). However, akin to what happens in the original FURPS model, this expansion lacks attributes to specify the system's portability [131].

ISO-9126 quality model Owing to the amount of quality models proposed, ISO decided to include a standardised quality model in its international standard for the evolution of software, ISO-9126, which also includes sections addressing external metrics, internal metrics and quality in use metrics. This standard specifies and evaluates the quality of a software product in terms of internal and external software qualities and their connection to attributes, categorising quality attributes into six independent high-level quality characteristics, which can in turn be broken down into secondary quality attributes or subfactors [4] [129, 132]:

- **Functionality**: decomposable into suitability, accuracy, security, interoperability and compliance.
- **Reliability**: can be broken down into maturity, fault tolerance, recoverability and compliance.

[4] Please note that compliance is present for every attribute. Further definitions can be found in the Glossary.

- **Usability**: may be further decomposed into understandability, learnability, operability, atractiveness and compliance.
- **Efficiency**: comprising time behaviour, resource behaviour and compliance.
- **Maintainability**: subfactor constituted by analysability, changeability, stability, testability and compliance.
- **Portability**: composed by adaptability, installability, co-existence, replaceability and compliance.

Requirement priorities

Although many requirements may be defined for a project, their fruition depends on real-world constraints such as limited resources or budget, scheduling or differing levels of importance in the requirements themselves. It is, therefore, important to establish priorities regarding a project's requirements, taking into account either their implementation order or their relative importance to the stakeholders [133]. A number of techniques have been described and compiled in the literature and are summarised below [134–136].

- **Analytical hierarchy process** or **pairwise comparison technique**: establishes a pairwise comparison for the requirements to determine which of the two is more important, and to what extent, using weighed scores and producing numerical outputs.
- **Bubble sort**: starts by aligning the requirements in a vector and comparing each requirement with the one directly above, swapping positions if their relative priorities seem misadjusted. The process is repeated for every single combination of requirements until the vector is ordered.
- **Cost-value prioritization**: determines the top requirements by creating graph plots to visualise the perceived value of the requirement vs. its implementation cost.
- **Hundred-dollar test**: works through a system of cumulative voting, in which imaginary USD 100 bills are distributed across the requirements according to their perceived relative importance. The requirements are then ordered by the number of bills each one has garnered.
- **Minimal spanning tree**: similar to the pairwise comparison, but avoids establishing comparisons that can be inferred (e.g. if requirement A is judged as being of higher priority than requirement B and requirement B is of higher priority than requirement C, then A is automatically judged to be of higher priority than C and that comparison does not have to be explicitly stated).
- **MoSCoW rule**: categorises requirements according to whether they are mandatory, highly desirable or nice to have in relation to the final product. The requirements are ranked from higher to lower priority by using the following mnemonic:

  M: MUST HAVE. Non-negotiable or mandatory requirements. Failure to deliver these requirements results in the failure of the project itself. These are the requirements with the highest priority.

  S: SHOULD HAVE. High priority requirements. Features that the project would benefit from, if their implementation is possible.

  C: COULD HAVE. Nice-to-have requirements. Features considered desirable but not necessary, that will be included if there is any time left after completing M and S.

  W: WILL NOT (or WISH TO) HAVE. "Wish list" requirements. Features that will not be implemented within the desired time frame, but may be considered in the future. These requirements have the lowest priority.

- **Numerical assignment**: works by grouping requirements into different categories determined by the person or team doing the assignment (e.g. high, medium and low).

2.5.2 Project planning

The current project was initially set to have duration of nine months, with its workload having been planned as described in Table 2.4, which can be translated into the Gannt charts shown in Figures 2.17, 2.18, 2.19 and 2.20.

Table 2.4: Project planning

	Activities	Begin date	End date
1	***Analysis***	**20/09/2017**	**22/12/2017**
1.1	Understanding the business problem	20/09/2017	03/11/2017
1.2	Identification of business and operational requirements	06/11/2017	23/11/2017
1.3	Requirement documentation and high-level solution design	24/11/2017	07/12/2017
1.4	Preliminary report	11/12/2017	22/12/2017
2	***Design***	**03/01/2018**	**31/01/2018**
2.1	Business and operational process design	03/01/2018	09/01/2018
2.2	Functional architecture design	10/01/2018	16/01/2018
2.3	Technical architecture design	17/01/2018	23/01/2018
2.4	Test strategy definition	24/01/2018	25/01/2018
2.5	Functional and technical design documentation	26/01/2018	29/01/2018
2.6	Test strategy documentation	30/01/2018	31/01/2018
3	***Development and testing***	**01/02/2018**	**31/05/2018**
3.1	Development environment setup	01/02/2018	01/02/2018
3.2	Component and unit test coding	02/02/2018	26/04/2018
3.3	Component tests and bug fixing	23/02/2018	26/04/2018
3.4	Integration tests and bug fixing	27/04/2018	10/05/2018
3.5	System tests and bug fixing	11/05/2018	24/05/2018
3.6	Setup, coding and test documentation	25/05/2018	31/05/2018
4	***Statistical results and report production***	**25/05/2018**	**20/06/2018**

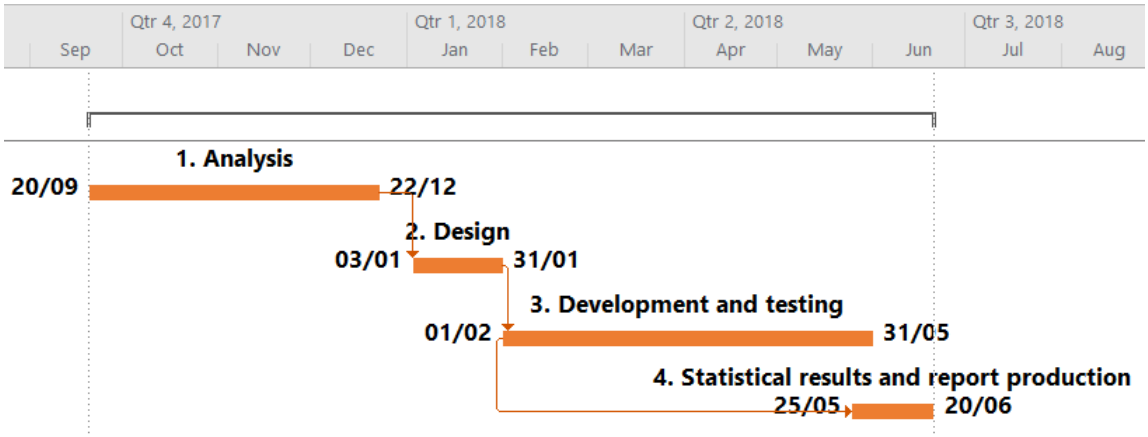


Figure 2.17: High-level Gantt chart containing the main tasks of the project.

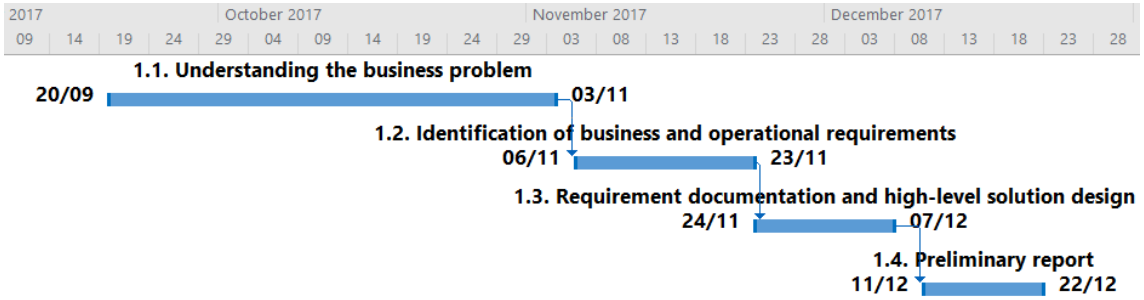


Figure 2.18: Gantt chart containing the subtasks of the project pertaining to task 1: Analysis.

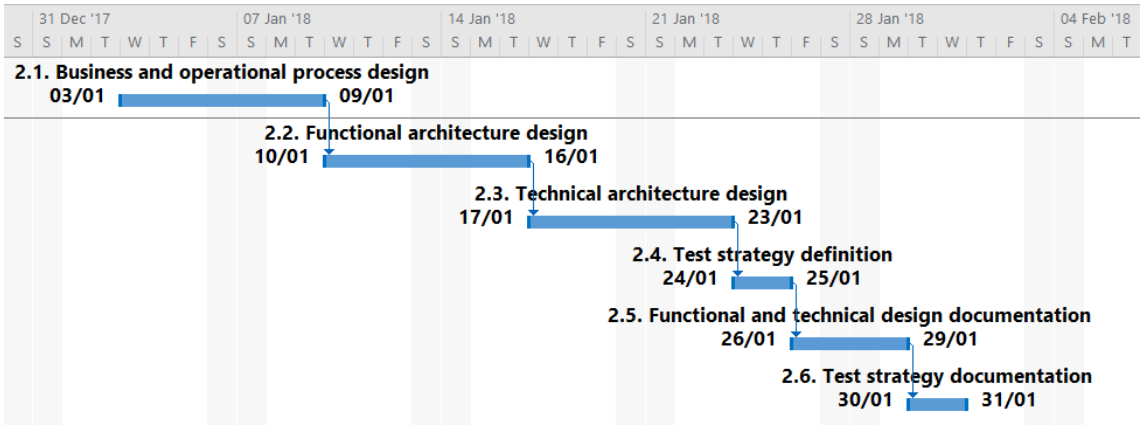


Figure 2.19: Gantt chart containing the subtasks of the project pertaining to task 2: Design.

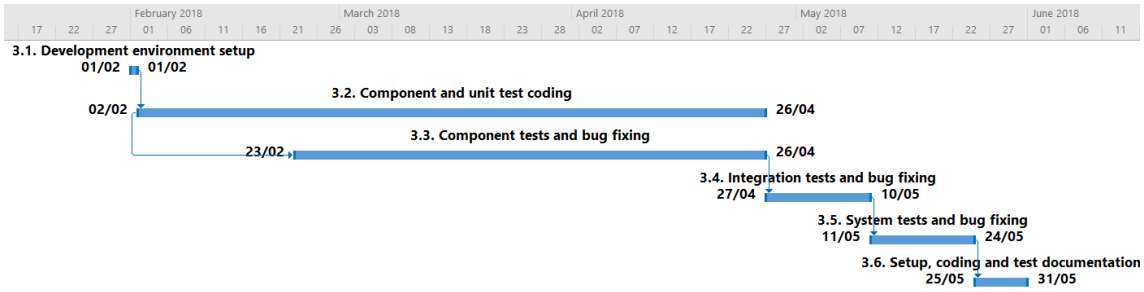


Figure 2.20: Gantt chart containing the subtasks of the project pertaining to task 3: Development and testing.

For the most part, the practical tasks for stages 1 and 2 (Analysis and Design, respectively) adhered to the specified time frame, with the preliminary report having been delivered in an incomplete state. The main exception, however, lies in the lack of definition, and therefore also the documentation of a testing strategy. The failure in delivering these steps is mainly due to the lack of focus on the methods to be used in the practical aspect of the putative solution. From this point on, an architecture was defined and basic unit tests were performed within the time limits, but the developed system was never subject to integration and system tests, nor was its possible incorporation with the parent system considered. The lack of useful results and general disperse nature of the work then made the production of the final report drag on for the better part of two years, with a report having finally been finished in July 2020.

The presented outcome, which differs significantly from the proposed schedule, can mainly be attributed to demotivation. I chose this project because I wished to perfect my skills in software engineering while learning a bit about artificial intelligence (AI) and computer vision, which I had never explored significantly. However, the project proved much more AI-oriented than I had foreseen, which curbed my initial enthusiasm. This demotivation was then built up in the following months, being compounded with the lack of promising results and general frustration with my difficulty to understand the technologies I was working with and what I could do to fine-tune the system and improve the results based on a solid theoretical background. This frustration was directly translated into the development of a poor and unstructured work ethic, with results being analysed and filed away instead of committing said analysis to paper immediately. The end result was the prospect of writing the whole work almost from scratch at the end, which eventually caused me to give up for the best part of two years, due to how insurmountable the task seemed and how little pride I managed to muster for the work that had been done. An aggravating factor also lied in the fact that I ceased being a full-time student in July 2018, which drastically reduced the time available for writing. The emergence of the COVID-19 pandemic in early 2020 and the resulting confinement rekindled my desire to see this work handed in, and therefore it was eventually finished in July 2020.

Chapter 3

Related work

Make and model recognition (MMR) is an increasingly important tool in traffic monitoring, license plate and vehicle recognition and law enforcement surveillance. The most distinctive feature in recognising a vehicle's manufacturer is the logo, which is usually present at the front of the vehicle and easily identifiable to the customer. Model recognition, however, is not so easily achieved, having to rely on the vehicle as a whole in order to extract patterns that allow for its classification [137, 138].

3.1 Recognising a vehicle's manufacturer

Wang et al. (2007)

Wang, et.al. (2007) [139] attempted logo recognition through the application of mathematical formulae. In order to achieve their ends, 11270 test images were used, pertaining to 17 classes which represented vehicle manufacturers. The general flow used can be found in Figure 3.1.

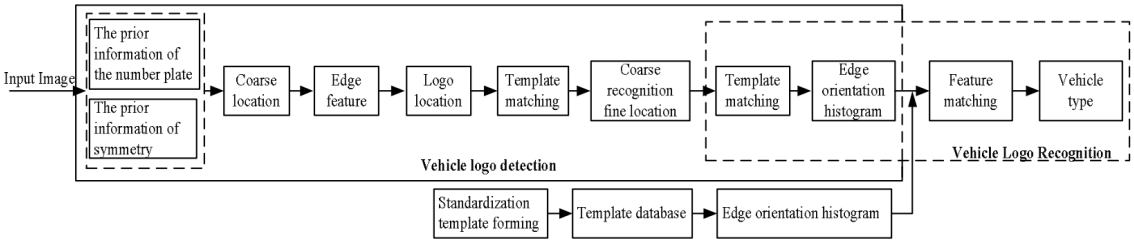


Figure 3.1: Processing flow used by Wang et al. (2007). (Retrieved from [139].)

The logo, or region of interest (ROI), was firstly subject to vertical coarse detection from the number plate, using the mathematical formula represented in Equation 3.1, where x is the rough position of the number plate, y is the rough position of the vehicle logo, y_1 is the bottom boundary, y_2 is the top boundary, and n is an adjustable parameter, $1 \leq n \leq 3$, with the authors choosing $n = 2$. These parameters are depicted in Figure 3.2.

$$\begin{cases} |y_1 - y_2| > nh = n|x - y| \\ y_2 > y \\ x < y_1 < y \\ |y_2 - y| = |y - y_1| \end{cases} \tag{3.1}$$

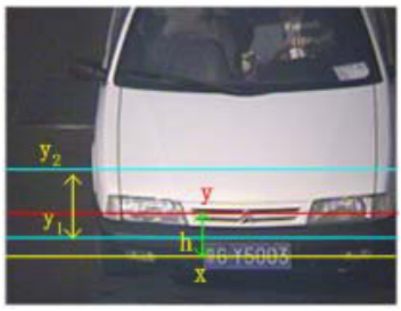


Figure 3.2: Vertical logo coarse location deployed by Wang et al. (2007). (Retrieved from [139].)

Following the determination of this vertical coarse location of the logo, edge features were computed using Sobel edge detection and projection of edge images, with determination of maxima and minima that helped refine the coarse location already obtained.

Given the symmetry inherent to the disposition of the logo in a grille image, an axis was then computed in order to obtain the left and right boundaries for a horizontal coarse location, as described by Equation 3.2, where x_1 and x_2 are the left and right boundaries, respectively, x_0 is the symmetry axis, and n is a coefficient, with $n \in [1, 2]$.

$$\begin{cases} x_1 = x_0 - n|x - y| = x_0 - nh \\ x_2 = x_0 + n|x - y| = x_0 + nh \end{cases} \tag{3.2}$$

Finally, the obtained results were refined taking all the calculations into account, as depicted in Figure 3.3.

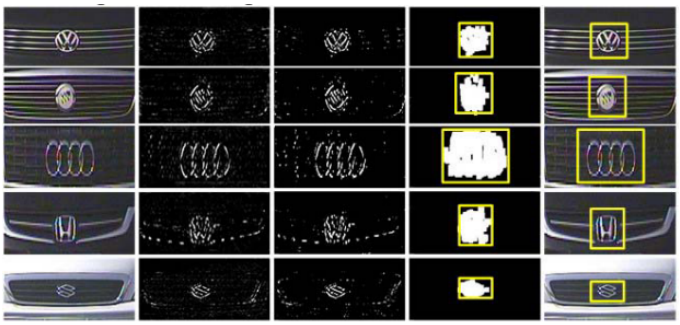


Figure 3.3: Logo location results obtained by Wang et al. (2007). Key: (a) coarse location results, (b) gradient image, (c) edge detection, (d) logo location results, (e) actual location in the original image. (Retrieved from [139].)

The obtained samples were then subject to recognition using edge orientation histograms, which were compared with histograms obtained for each logo in order to perform the matching. These histograms can be viewed in Figure 3.4.

Two different approaches were attempted: template matching in the location vehicle logo area (M), and fine location and coarse recognition by edge feature information (M+E). From this process, a recognition rate of 89.85% was achieved for the M+E method, with a lower performance (78.01%) being obtained for M, which did not use edge information. However, this work is limited in scope, since the proposed methods fail to account for factors such as different light conditions, as the authors acknowledge in their final conclusions.

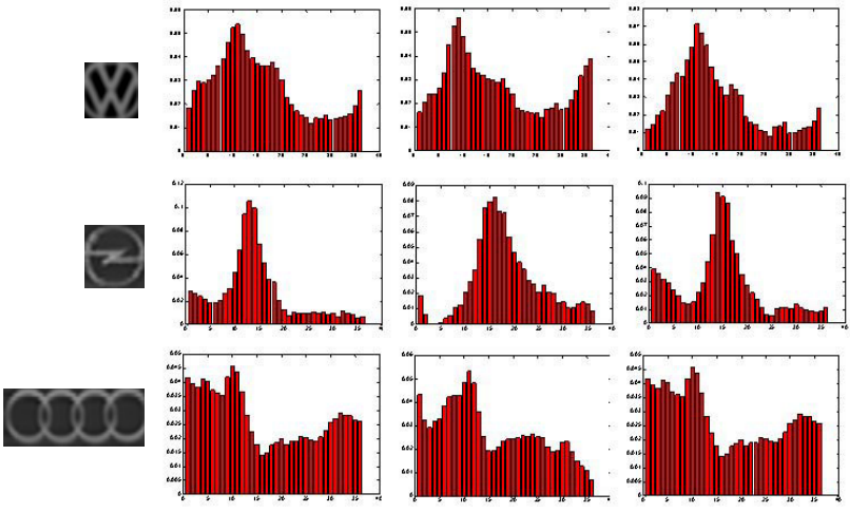
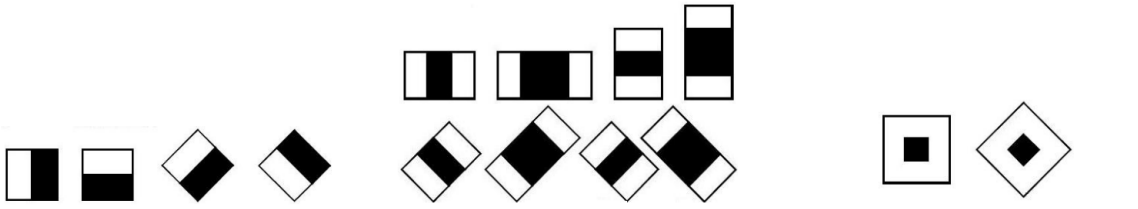


Figure 3.4: Edge histograms for different logos, depicted by Wang et al. (2007). (Retrieved from [139].)

Sam and Tian (2012)

In 2012, Sam and Tian [140] presented a solution that aimed to take distortions due to viewpoint variation into account. In order to do so, the authors made use of the extended set of Haar features proposed by Lienhart and Maydt [93], depicted in Figure 3.5. The value of each feature was calculated as the difference of the sum of the pixels of rectangular areas.



(a) Edge features (b) Line features (c) Center-surround features

Figure 3.5: Haar features used for vehicle logo detection by Sam and Tian (2012). (Retrieved from [140].)

Using these features, detection was then performed using a modest AdaBoost algorithm, which makes use of a series of weak classifiers and searches over its pool to find the one with the lowest classification error, rectifying its weights with each iteration, as shown in Figure 3.6. The results were then normalised and classification was made by making use of Euclidian distances between points of interest, with a recognition rate of 92 % having been achieved.

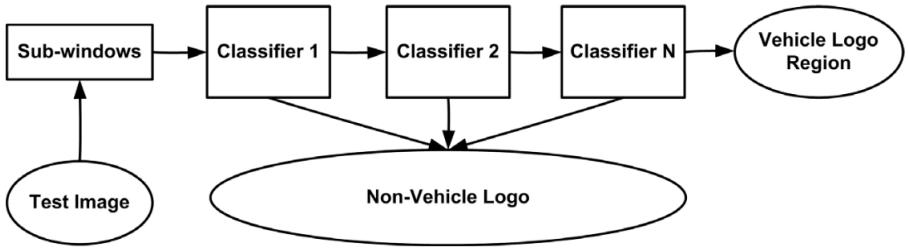


Figure 3.6: Modest AdaBoost algorithm as described by Sam and Tian (2012). (Retrieved from [140].)

Huang et al. (2015)

In 2015, Huang et al. [141] presented a promising work on car model recognition through the logo region. The authors chose to use CNNs containing maxpooling layers, and pre-training the data by use of principal component analysis (PCA). A total of 11 500 images from the 10 manufacturers considered to be the most popular in China were used, obtaining a classification accuracy of 99.07%. The authors operated on assumptions about the location of the vehicle's license plate (LP), using coarse segmentation to define a general ROI. The CNN constructed operates on these ROIs thus obtained, according to a hierarchy. To Huang et al., logos are objects in these images, which can be deconstructed into motifs, which in turn can be decomposed into edges, making this a hierarchical extraction of features. PCA was used in order to improve the CNN's performance, since it allows for initialisation at interesting points (as opposed to completely random ones) and therefore requires less iterations, avoiding the use of backpropagation (BP). The architecture of the CNN implemented is translated as follows:

1. Input layer
2. 1^{st} convolutional layer: six feature maps, with a kernel size of 7×7
3. 1^{st} pooling layer: six feature maps, with a kernel size of 2×2
4. 2^{nd} convolutional layer: twelve feature maps, with a kernel size of 21×21
5. 2^{nd} pooling layer: twelve feature maps, with a kernel size of 2×2
6. Fully-connected layer
7. Output layer

The CNN's architecture is illustrated in Figure 3.7.

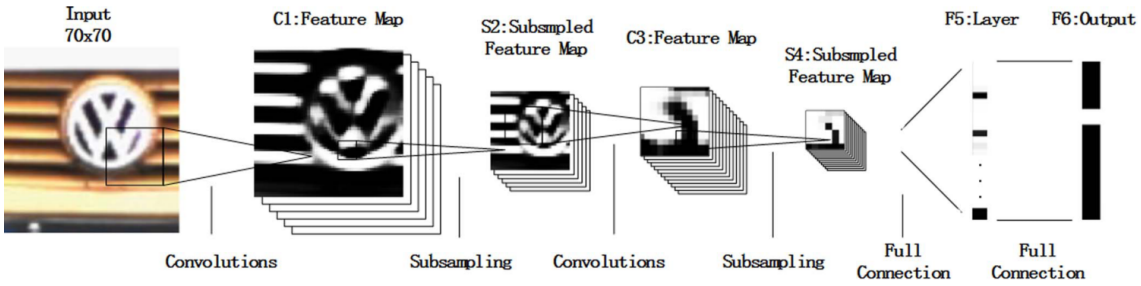


Figure 3.7: Architecture of the convolutional neural network used by Huang et al. (2015). (Retrieved from [141].)

The authors obtained an overall accuracy of 99.07 %, with perfect recognition for manufacturers such as Honda and Peugeot. These results are displayed in Table 3.1.

Manufacturer	True	False	Accuracy
Honda	150	0	100.00%
Peugeot	150	0	100.00%
Buick	142	8	94.67%
VW	150	0	100.00%
Toyota	150	0	100.00%
Lexus	150	0	100.00%
Mazda	148	2	98.67%
Chery	150	0	100.00%
Hyundai	147	3	98.00%
Citroen	149	1	99.33%
Average	1486/1500	14/1000	99.07%

Table 3.1: Detailed results obtained by Huang et al. (2015). (Retrieved from [141].)

3.2 Recognising manufacturer and model

Psyllos et al. (2011)

Psyllos et al.'s 2011 study [142] attempted RT matching in two phases, first recognising a vehicle's manufacturer and then, using one DB for each vehicle make, detecting its model. The authors used frontal photos only, identifying the ROI through a mask based on the LP's location, using SCW segmentation. The general architecture presented in the paper is described in Figure 3.8.

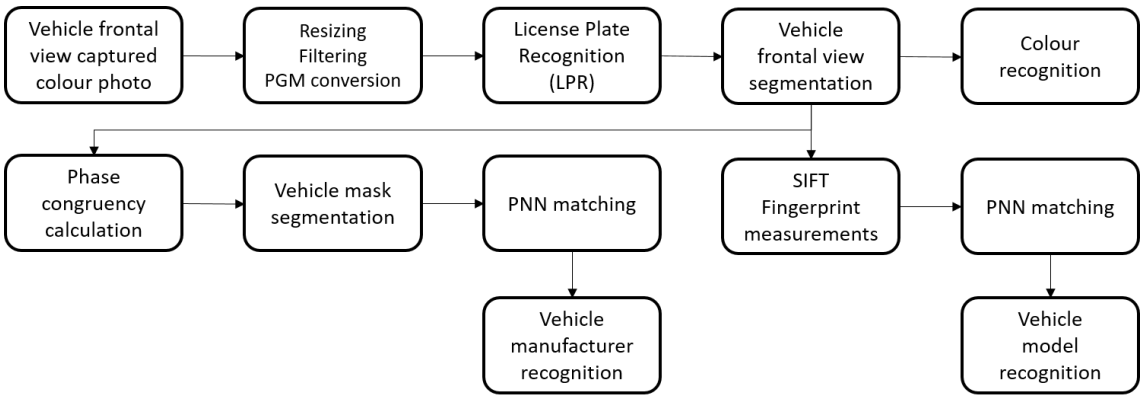


Figure 3.8: Vehicle manufacturer and model recognition system architecture used by Psyllos et al. (2011). (Adapted from [142].)

- **VLPR**: The first step undertaken by the authors was to convert each image to greyscale, using the PGM format and 8-bit resolution, and scaling it. The images were then fed to a LPR module, which used SCW segmentation followed by the application of Anagnostopoulos's mask (a binary mask produced by SCW) [143] and Sauvola's binarisation method, which takes a greyscale image and computes a threshold at each pixel in order to clean up the image [144, 145]. The binarised image was then labelled using connected-component labelling, which transforms a binary image into a symbolic image so that each connected component is assigned a unique label, and binary measurements were applied [146]. The output of this module are the coordinates of a rectangular area that includes the vehicle plate number. The success rate for plate segmentation was of 96.5% and for content recognition of 89.1%.
- **Frontal view image segmentation**: In this section, the ROI is defined as the vehicle's mask.

This mask is defined as possessing width and height $W_{mask} = 4 \times W$ and $H_{mask} = 2 \times H$, where W and H are the width and height of the segmented LP. This method was introduced by co-author Anagnostopoulos in 2002 [147].

- **Mask image segmentation**: At this stage, the authors applied a method based on phase congruency calculation to assess the existence of significant features. The phase congruency map gradient threshold applied yielded five general segments (left light, left part of the grill, logo, right part of the grill and right light), as described in Figure 3.9.

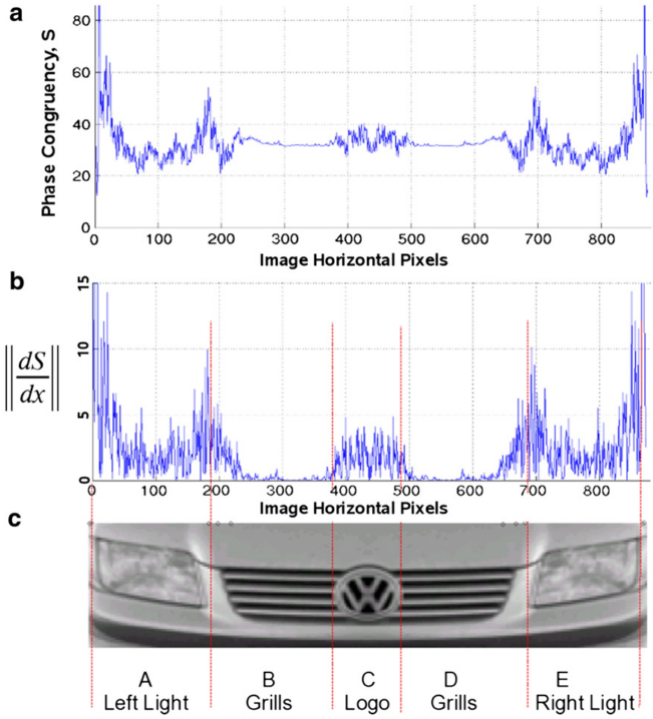


Figure 3.9: Mask image segmentation used by Psyllos et al. (2011). a) Phase congruency feature map. b) Derivative of phase congruency. c) Mask segmentation based on the phase congruency map gradient threshold. (Retrieved from [142].)

- **Colour recognition**: After determining the vehicle mask, the image was cropped using the area over the mask and covering the hood. RGB colour histograms were computed, and the peak of each histogram (R_{max}, G_{max}, B_{max}) defining the colour of the vehicle.
- **Manufacturer recognition**: the vehicle brands were subject to a probabilistic neural network (PNN) with 4 layers, as depicted in Figure 3.10, achieving an accuracy of 85%.

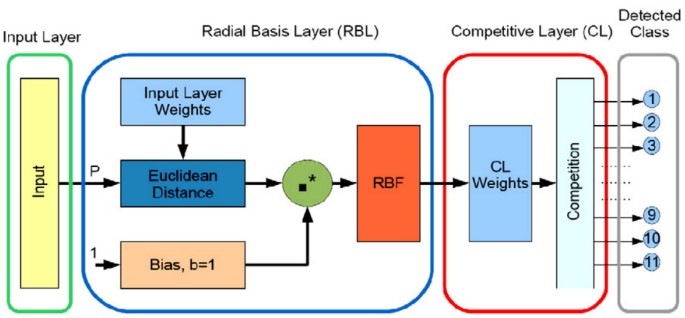


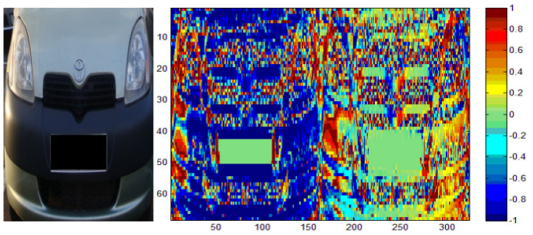
Figure 3.10: Architecture for the PNN developed by Psyllos et al. (2011). (Retrieved from [142].)

- **FP measurements and model recognition**: this technique employed SIFT keypoint descriptors, achieving an accuracy of 54% in model recognition.

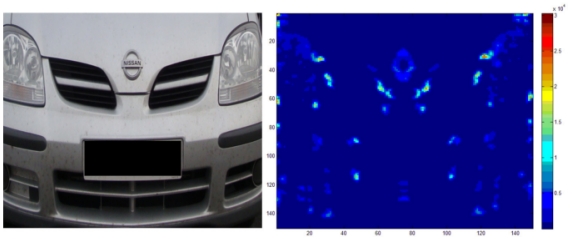
Pearce and Pears (2011)

In 2011, Pearce and Pears [148] presented a comparison work for various methods of MMR, using 177 test images pertaining to 74 distinct make and model classes. The ROI, comprising the full width of the vehicle, the lights and grill, was defined manually, using 6-DOF affine mapping, and normalised for each of 60 images, allowing for a definition of an optimal estimated ROI for the whole dataset. A set of methods was posteriorly applied and successively refined.

- **Canny-edge detection**: The authors started by building upon the work of Munroe and Madden (2005) [149], who had studied multi-class and single-class techniques of model recognition in frontal images of cars, settling on techniques based on kNN as the most promising ones. The features were detected by a Canny edge detector, using thickened edges (1px) and then applying kNN with one and three neighbours, evaluating the results using Euclidean metrics. The results were of 79.1% correct classification rate for 1-NN and 81.4% for 3-NN. The authors established this solution as the baseline to measure the subsequent experiments against.
- **Square mapped gradients**: Using the methodologies proposed by Petrović and Cootes (2004) [150], the authors concatenated the values of g_x and g_y in order to form a single feature vector with two values for each pixel, as exemplified in Figure 3.11a. The correct classification performances achieved for 1-NN and 3-NN were of 91.0% and 89.8%, respectively.
- **Harris corners**: Basing this section on the work of Harris and Stephens (1988) [84], the authors decided to use Noble's (1989) [151] corner strength measure to construct a feature vector, as shown in Figure 3.11b, and achieving a performance similar to the CED (78.0%) when using a 1-NN classifier.
- **Recursive partitioning and local normalisation**: After performing the methods already described, the authors noticed that, since the structures did not line up perfectly after normalisation, one of two approaches should be attempted: either employ a metric with a concept of neighbourhood, replacing Euclidean metrics with "earth mover" ones, or rethink the feature vector so that it would be less sensitive to alignment. The authors then decided to recursively divide the images into quadrants (2 columns $\times$ 2 rows), constructing summed feature outputs within the divisions of each level, and taking normalisation in a localised way. Using this technique with a recursion depth of 5, the authors retested Harris corners, achieving a performance of 94.9% when using a 1-NN classifier.
- **Naïve Bayes**: The authors decided to test a Naïve Bayes classifier in order to exploit prior information about the relative frequencies of cars in each class, assuming independence between observations. The use of Naïve Bayes improved the correct classification rates of locally normalised Harris corners and square mapped gradients to 96.0%, although the authors state these results to be harder to replicate than the ones obtained using kNN.



(a) Results for the square mapped gradients. Left: original image. Right: gradient results for the g_x (left) and g_y (right) components, prior to concatenation.



(b) Results for the Harris corners algorithm. Left: original image. Right: corner strength.

Figure 3.11: Graphic results for the algorithms used by Pearce and Pears (2011). (Retrieved from [148].)

Ramnath et al. (2014)

A different attempt was performed by Ramnath et. al. [152] in 2014, through the use of 3D curve-matching using partial 3D models based on different viewpoints. For this purpose, the authors made use of Chamfer matching, which minimises the distance between edge points in two images, and which was then projected into a 3D virtual hull, or approximate shape representation, using silhouette cones. The authors then proceeded to estimate each vehicle's pose by observing the relative positioning of its wheels. A representation of the results obtained in each stage is depicted in Figure 3.12.

Figure 3.12: Stages for the development of a vehicle's 3D model used by Ramnath et al. (2014). Top: three of the images used to generate the hull; centre: visual hull; bottom: 3D space curves projected into the visual hull. (Retrieved from [152].)

This work was focused in a single brand (Honda), having used 20 images for each of 8 distinct Honda models as well as 30 images for other vehicles, in a total of 190 test images. The authors determined a 95% recall rate as the threshold for rejection, with the results being displayed in the confusion matrix depicted in Table 3.2.

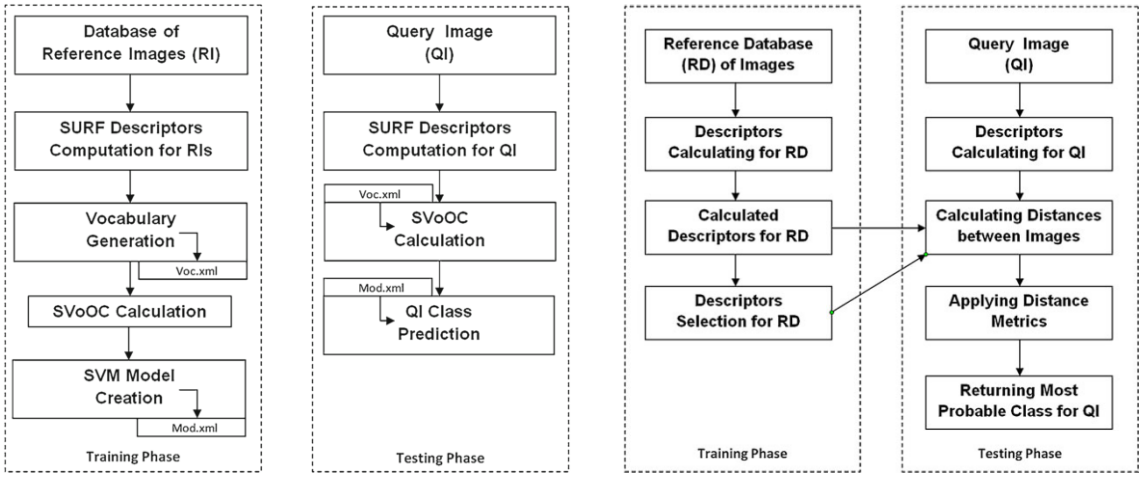
Table 3.2: Confusion matrix obtained by Ramnath et al. (2014). (Retrieved from [152], complemented by information from Hsiao et al. (2014) [153].)

	CivC	CivS	InsH	Ody	Pil	Rid	Ele	Fit	other
CivC	**0.85**	0.10	-	-	-	-	-	-	0.05
CivS	0.05	**0.75**	0.05	-	-	-	-	0.10	0.05
InsH	-	-	**0.90**	-	-	-	-	0.05	0.05
Ody	-	-	-	**0.90**	-	-	-	0.05	0.05
Pil	-	-	-	-	**0.85**	0.10	-	-	0.05
Rid	-	-	-	-	0.10	**0.85**	-	-	0.05
Ele	-	-	-	-	0.05	-	**0.95**	-	-
Fit	-	-	-	-	-	-	-	**0.95**	0.05
other	0.03	0.15	0.03	-	0.30	0.03	0.03	0.03	**0.40**

Key: CivC = Honda Civic Coupe, CivS = Honda Civic Sedan, InsH = Honda Insight, Ody = Odyssey 2011, Pil = Pilot 2011, Rid = Ridgeline 2011, Ele = Element SC 2010, Fit = Honda Fit 2011.

Baran et al. (2015)

Baran et al. (2015) [137] made use of two different approaches to the MMR problem, a real-time one based on speed and another, non-real-time, based on accuracy. The region of interest (ROI) for each image was calculated using Haar-like features (present in OpenCV's *haartraining* functionality).



(a) Real-time algorithm (b) Non-real-time algorithm

Figure 3.13: Algorithms used by Baran et al. (2015). (Retrieved from [137].)

- **Real-time**: the real-time solution aimed to estimate a car's model in milliseconds, combining the use of a SURF detector with a SVM algorithm. The SURF descriptors were calculated for each image's ROI and the collection of descriptors was then partitioned by use of the k-means algorithm. The assignment of SURF descriptors related to a given pattern image lead to the creation of sparse vectors of occurrence counts (SVOOC). A multi-class SVM was then constructed using those SVOOC, with a one-against-one algorithm. The training and testing processes are summarised in Figure 3.13a.
- **Non-real-time**: the non-real-time solution aimed to achieve a higher classification accuracy, making use of visual descriptors. The authors used visual content classification (VCC) with MPEG-7 EH descriptors, as well as SIFT and SURF descriptors for local features, applying

distance metrics to the images presented for classification and merging the various outputs in order to establish a final prediction. The process is summarised in Figure 3.13b.

The authors used a set of 1360 training images and 2499 test images, depicting 17 models of cars from 6 manufacturers. The accuracy reported was of 91.7% for the real-time algorithm and of 97.2% for the non-real-time one.

3.3 Overview

3D models

Table 3.3: Summary of related works: 3D models

Work	Type of 3D model	Techniques	Performance
Ramnath et al. (2014) [152]	Partial, based on different viewpoints	Virtual hull model + minimisation of reprojection errors + pose estimation	95% recall

ROI extraction

Table 3.4: Summary of related works: ROI extraction

Work	ROI	ROI extraction	Accuracy
Baran et al. (2015) [137]	Grill + head and indicator lights	Haar-like detector	97.2%
Huang et al. (2015) [141]	Logo	Based on LP location (coarse segmentation)	N/A
Pearce and Pears (2011) [148]	Full width + lights + grill	Average of manually defined ROI for samples	N/A
Psyllos et al. (2011) [142]	Grill + head and indicator lights	"Mask" based on LP location	N/A
Sam and Tian (2012) [140]	Logo	Extended Haar-like features + modest AdaBoost	N/A
Wang et al. (2007) [139]	Logo	Sobel edge detection + coarse detection + refinement	N/A

Fingerprinting by use of feature descriptors with ML classifiers

Table 3.5: Summary of related works: feature descriptors with ML classification techniques

Work		Descriptors	Classification	Accuracy	Observations
Baran et al. (2015) [137]	RT	SURF	K-means	91.7%	33.69 ms (SVOOC) 4.24 ms (classification)
Huang et al. (2015) [141]		N/A	CNN	99.07%	Pretraining with PCA
Pearce and Pears (2011) [148]		Harris corners + partitioning	Naïve Bayes classifier	96.0%	N/A
Psyllos et al. (2011) [142]	Make	SIFT	PNN	85%	PNN setup: 1428 ms
	Model			54%	Fixed model (VOLK)
Sam and Tian (2012) [140]		Radial Tchebichef moments + normalisation	1-NN	92%	>2 s/image

Fingerprinting by use of of feature descriptors with distance metrics

Table 3.6: Summary of related works: feature descriptors with distance metrics

Work		Descriptors	Accuracy	Observations
Baran et al. (2015) [137]	NRT	SIFT + SURF + MPEG-7 (edge histogram)	97.2%	1.93 s/frame
Wang et al. (2007) [139]		Sobel edge detectors	89.85%	N/A

Chapter 4

Analysis

4.1 System requirements

4.1.1 Stakeholders

Stakeholders are defined as individuals or groups who have a stake in, or expectation of, a project's performance [154]. The current project's stakeholders can be divided into three main groups: clients, Accenture's TCoE team and FCUL's academic interested parties.

- **Accenture clients** want to accurately identify transgressors from RSE images to charge them in the minimum time possible, to minimise transaction costs and to be able to configure the system to their will if they so desire.
- **TCoE's team** comprises a number of people with distinct interests.
  - The project managers want a new and innovative MMR product that can be monetised and sold to their clients as fast as possible, with quality and minimal costs.
  - The software architect wants the new system to interact modularly with the existing ones and adopt the same architectural constraints without compromising the parent system's efficiency.
  - The developers and testers want the new system to be easily maintainable, not provoke undesired changes in the existing system's code or functionality and to not compromise it in case a fault occurs.
- **Academia** members also have separate and well-defined roles that determine what each of them want from the project.
  - The internal supervisor wants to obtain innovative and publishable results in the MMR field.
  - The external supervisor wants to obtain innovative and marketable results in the MMR field, as well as a viable plan of action and descriptions of the challenges his team may face while developing an MMR solution.
  - The student wants to design a system that fulfils the requirements imposed by FCUL's PEI and Accenture, including the proper documentation.

4.1.2 Quality models

Of the models presented for the definition of quality for a project, the FURPS quality model was chosen for the definition of the project requirements, due to its simplicity and explicit distinction between FR and NFR. However, this model omits some important concerns. Therefore, besides the requirements based on the FURPS quality model, Accenture- and TCoE-enforced architecture

concerns on functional and technical requirements were considered for the NFR. These concern the following requisites, giving definitions about how they should be interpreted and defined:

- Availability: "Is the system available 24 $\times$ 7? And is it available 99.9% or 99.99% of the time?"
- Maintainability: "How easy is it to maintain?"
- Operability: "How easy is it to use and manage the system? How is it operated?"
- Performance: "How fast does the system respond? What is the maximum throughput?"
- Recoverability: "How does the application recover from a fault?"
- Resilience: "If there is an issue, is the data protected? What needs to be preserved?"
- Scalability: "How many users or transactions can the system process concurrently?"
- Security: "Can the system sustain attacks? Is the data protected?"

It should be noted that performance is also addressed in FURPS. Moreover, in the definition of the system requirements, operability requirements have been combined with usability, since in this case there will not be final users, but only developers and systems that access this one. All requirement types are identified in capitals, using the first three letters of their name (FR for functional requirements, the specific type otherwise, e.g. AVA for availability).

4.1.3 Functional requirements

Functional requirements (FRs) are those requirements that deal with the way a software system is supposed to work according to its stakeholders. In other words, FRs are focused solely on the system's functionality [155, 156].

The functional requirements considered for this project are listed as follows:

FR-01 The system shall recognise the manufacturer of a vehicle from frontal and rear images taken from a selected image set.

FR-02 The system shall recognise the manufacturer of a vehicle from frontal and rear images taken in real visibility conditions.

FR-03 The system shall recognise the model of a vehicle from frontal and rear images and given its manufacturer.

FR-04 The system shall recognise vehicle makes from all the top-selling manufacturers in the US.

4.1.4 Non-functional requirements

Non-functional requirements (NFR) are those that deal with constraints and attributes of the system or system environment, being sometimes colloquially referred to as "-ilities" or "-ities". The definition of NFR is not consensual between authors, but it is generally agreed that the term encompasses those concerns that are not related to the functionality of the software [155, 156].

The non-functional requirements considered for this project are listed by type as follows:

Availability

AVA-01 The system shall possess the same availability values as the parent system (OBO).

Maintainability

MAI-01 The system shall be accompanied by a high-level document explaining what the configuration values affect, to ensure that anyone can tweak the system for optimal performance.

MAI-02 The system's architecture shall be documented and accompanied by an element catalogue detailing the responsibilities of each architecture module.

MAI-03 The code produced for the system shall be properly commented and annotated.
MAI-04 The code produced for the system shall adhere to the standardised company good practices.

Performance

PER-01 The system shall deliver its results in near RT, that is, within the same range as OBO.
PER-02 The system shall have a maximum time-limit to produce an answer, passing which a default answer will be provided.
PER-03 The system shall be able to support a throughput at least equal to OBO's.
PER-04 The system shall be able to incorporate a new element into the model and update said model with no downtime.

Recoverability

RCV-01 Should the system experience a fault, there shall be mechanisms to restore it as close to its previous state as possible.
RCV-02 The system shall implement logging.
RCV-03 In case of fault, a back-up system or server shall be kicked into action.

Reliability

REL-01 The system shall be able to recognise a vehicle's manufacturer with an accuracy of at least 80% and with as few false positives as possible.
REL-02 The system shall be able to recognise a vehicle's model given its manufacturer with an accuracy of at least 80% and with as few false positives as possible.
REL-03 System downtime shall not affect OBO's performance.
REL-04 The system shall always provide an answer.

Resilience

RES-01 The system shall guarantee the persistence of every image fingerprint calculated.
RES-02 The system shall guarantee the persistence of the computed models and networks.
RES-03 The system shall guarantee the persistence of the output labels it computes.
RES-04 The system shall guarantee the recording of every transaction it handles.

Scalability

SCA-01 The system shall be able to process as many parallel transactions per time unit as its parent system.

Security

SEC-01 Any permanently stored data shall be subjected to the same security measures as the data stored by the parent system.
SEC-02 Any flux of information between logical components and/or physical machines shall be subjected to the same security measures as those implemented in the parent system.

Supportability

SUP-01 The system's services shall be accessible through an API that provides them independently of the kind of system that might use them.

SUP-02 The system implementation shall withstand the addition of further modules without compromising its performance and reliability properties.

Usability/operability

USA-01 The system shall be delivered/hosted as software-as-a-service (SaaS).
USA-02 The system shall provide an interface that renders its implementation transparent to other using services.
USA-03 The system shall be accompanied by a high-level document explaining how it works to increase its understandability.
USA-04 The system's weights and configurations shall be able to be changed without causing system downtime.
USA-05 The system shall be able to undergo fine-tuning according to consumer tendencies in a given region/country.

4.1.5 Requirement prioritisation

The requirements identified in the previous sections do not all hold the same degree of importance or urgency to the project. Therefore, a prioritisation technique must be employed to ensure the proper relative importance of these requirements is established. Among the techniques described for requirement prioritisation, the MoSCoW rule was chosen, since it has a set of intuitive pre-established categories that allow for a relatively fast categorisation of requirements without the need for exhaustive comparisons. Since the project is set as an isolated entity, in that it is considered as a single release for academic purposes, it was agreed that the **W** level would not be used, as the requirements defined are all supposed to be part of the project to some extent.

Functional requirements

The priorities established for the FR are described in Table 4.1.

Table 4.1: Priorities established for the functional requirements.

Code	Requirement	Priority	Observations
FR-01	The system shall recognise the manufacturer of a vehicle from frontal and rear images taken from a selected image set.	M	Frontal or back recognition is critical for the recognition of the vehicle's manufacturer, and the system must be able to recognise images in "ideal" conditions (i.e., images that have a clear shot of a car, with adequate lighting and favourable poses) before being able to graduate to real visibility conditions.
FR-02	The system shall recognise the manufacturer of a vehicle from frontal and rear images taken in real visibility conditions.	S	The RSE photos are taken in real conditions, and so the training samples should reflect that if the present work is to reliably identify manufacturers from such images. However, this is a more difficult task than the already complex one of performing identification on "ideal" images, and therefore must be addressed as a way of fine-tuning the system, under penalty of no recognition in any visibility conditions.
FR-03	The system shall recognise the model of a vehicle from frontal and rear images and given its manufacturer.	S	There is a standardisation in car models that may difficult this task, so manufacturer recognition must take precedence over model recognition, under penalty of there being no recognition at all.
FR-04	The system shall recognise vehicle makes from all the top-selling manufacturers in the US.	C	Adapting the system to the US market specifically could prove to be a good way of fine-tuning it, but it is not of high priority.

Non-functional requirements

The priorities established for the NFR are described in Table 4.2.

Table 4.2: Priorities established for the non-functional requirements.

Code	Requirement	Priority	Observations
AVA-01	The system shall possess the same availability values as the parent system (OBO).	S	Highly desirable, but the main priority would be to have a fully functioning system, even if its availability is slightly lower than OBO's.
MAI-01	The system shall be accompanied by a high-level document explaining what the configuration values affect, to ensure that anyone can tweak the system for optimal performance.	M	Configurations must be properly documented so the system can be understandable.
MAI-02	The system's architecture shall be documented and accompanied by an element catalogue detailing the responsibilities of each architecture module.	S	System architecture documentation will ideally be present at a high level, and detailed save for time constraints.
MAI-03	The code produced for the system shall be properly commented and annotated.	S	The system's code should theoretically be self-explainable, but comments and annotations are desirable.
MAI-04	The code produced for the system shall adhere to the standardised company good practices.	M	The system's code must be legible and easily changeable if necessary, and so adherence to good practices is vital.
PER-01	The system shall deliver its results in near RT, that is, within the same range as OBO.	S	It is highly desirable that the results be delivered in a time frame that does not cause system bottlenecks.
PER-02	The system shall have a maximum time-limit to produce an answer, passing which a default answer will be provided.	M	The system cannot, in any way, get stuck waiting for an answer, and thus a time-limit is essential.
PER-03	The system shall be able to support a throughput at least equal to OBO's.	S	It is highly desirable that the incoming images be processed in a time frame that does not cause system bottlenecks.

Code	Requirement	Priority	Observations
PER-04	The system shall be able to incorporate a new element into the model and update said model with no downtime.	M	After classifying a received input, the system must be able to incorporate it into its model and recalculate it without ceasing to be available for new classifications.
RCV-01	Should the system experience a fault, there shall be mechanisms to restore it as close to its previous state as possible.	M	Restoring the system to a previous state is vital and not doing it could mean losing all the data and rendering the system unusable.
RCV-02	The system shall implement logging.	M	Logging is an essential tool to re-establish the system in case of failure and would aid in error depuration.
RCV-03	In case of fault, a back-up system or server shall be kicked into action.	C	While a back-up would be a nice improvement to have, the system itself is meant to be parallelised, making this a lower priority requirement.
REL-01	The system shall be able to recognise a vehicle's manufacturer with an accuracy of at least 80% and with as few false positives as possible.	S	Ideally the system would have very high accuracy and few to no false positives, but obtaining a working system, even if it does not completely fulfil these conditions, is the top priority.
REL-02	The system shall be able to recognise a vehicle's model given its manufacturer with an accuracy of at least 80% and with as few false positives as possible.	S	Ideally the system would have very high accuracy and few to no false positives, but obtaining a working system, even if it does not completely fulfil these conditions, is the top priority. Moreover, the recognition of the manufacturer with high precision is of higher priority.
REL-03	System downtime shall not affect OBO's performance.	M	Under no circumstances can the system compromise OBO or the systems and databases it interacts with.
REL-04	The system shall always provide an answer.	M	The system cannot get stuck waiting for an answer or enter a loop, and so a time-limit to provide a default answer must be established.
RES-01	The system shall guarantee the persistence of every image fingerprint calculated.	S	Image fingerprints are the means to an end, classification. Therefore, it is highly desirable that they be persisted, but not mandatory.

Code	Requirement	Priority	Observations
RES-02	The system shall guarantee the persistence of the computed models and networks.	M	The persistence of the computed models is essential to the way the system works.
RES-03	The system shall guarantee the persistence of the output labels it computes.	M	The system's purpose is to produce classifications under the form of output labels, which means these results must be persisted.
RES-04	The system shall guarantee the recording of every transaction it handles.	M	The system must maintain a record of the transactions it handles so anomalies can be detected.
SCA-01	The system shall be able to process as many parallel transactions per time unit as its parent system.	C	The system cannot condition the way its parent system works, but the amount of transactions processed will probably be lower than OBO's, since not all vehicles will need identification through MMR.
SEC-01	Any permanently stored data shall be subjected to the same security measures as the data stored by the parent system.	M	All the data pertaining to the project must be protected using the same mechanisms.
SEC-02	Any flux of information between logical components and/or physical machines shall be subjected to the same security measures as those implemented in the parent system.	S	Communications and transference of data will ideally be protected, but this is not as essential as the protection of stored data.
SUP-01	The system's services shall be accessible through an API that provides them independently of the kind of system that might use them.	S	The system's implementation will ideally attempt to be established through an independent API.
SUP-02	The system implementation shall withstand the addition of further modules without compromising its performance and reliability properties.	M	The system must be sufficiently modular to allow for the addition of further capabilities.
USA-01	The system shall be delivered/hosted as software-as-a-service (SaaS).	M	The establishment of the system as SaaS is one of the main requirements from the project leads.
USA-02	The system shall provide an interface that renders its implementation transparent to other using services.	S	The transparency of the way the system is implemented is a highly desirable quality.

Code	Requirement	Priority	Observations
USA-03	The system shall be accompanied by a high-level document explaining how it works to increase its understandability.	M	Understandability of the system is vital to modifying and expanding upon it.
USA-04	The system's weights and configurations shall be able to be changed without causing system downtime.	M	Changing configurations must be easily doable so the system will not have to be shut down to modify its functionality.
USA-05	The system shall be able to undergo fine-tuning according to consumer tendencies in a given region/country.	C	Adaptability to regional consumer tendencies could lead to the creation of a product family and improve the product, but it is not the main focus of the project.

4.2 Variability in MMR

The process of vehicle make and model recognition (MMR) is a complex one, since a diverse array of factors contribute to determine its success. It was, therefore, desirable that these factors be properly identified and documented so that future efforts by the TCoE team might be able to identify and prevent potential issues related to variability in MMR.

In order to identify potential sources of variability, a divide-to-conquer strategy was adopted. Therefore, three main categories were considered:

- Road-side equipment (RSE) properties
- Capture event properties
- Vehicle properties

Each category was then further divided in order to determine some causes for variability, as described below. It should be noted that each category was discussed under the assumption that the remaining ones would remain constant, so as to reduce complexity.

4.2.1 RSE properties

Road-side equipment (RSE) is a vital component of electronic toll collection (ETC) systems, being responsible for recognising a tolling event and creating the corresponding transaction. According to documentation made available by EasyGo, the main components of a RSE module are [157]:

- Beacon: responsible for identifying and communicating with on-board units (OBU) of passing vehicles.
- Controller: computer that collects information from beacons and other subsystems, generates transactions according to that information and is responsible for their transmission for processing.
- Sub-modules and peripherals: including, but not limited to, a vehicle detection system capable of detecting, positioning and, if applicable, classifying a vehicle. These modules may also include a video registration system for enforcement, traffic lights and/or a barrier, differing according to the needs of the toll collecting company using the RSE.

When it comes to MMR, the registration of images of passing vehicles is particularly important, since those images will serve to analyse those vehicles. However, these images and their quality depend on a number of factors related to each RSE, namely:

- **Camera resolution**: this factor is inherent to each camera, and therefore constant for each RSE, barring equipment replacements. Image quality is, nonetheless, extremely relevant in vehicle classification of any kind, since an insufficient resolution may lead to misclassification due to lack of data.

- **Type of image capture performed by the RSE**: each equipment is typically focused on capturing a certain type of image, be it lateral, frontal or rear shots of passing vehicles.

- **RSE position or angle**: vehicle recognition may be affected by the angle in which a photo is taken even if the type of capture performed is the same, since a change of perspective may enhance or disguise important or distinguishing features of the brand and/or model of a vehicle.

- **Frame rate**: the frame rate of video capture affects the amount of images captured per second, meaning that a lower frame rate will result in fewer images of the vehicle being available for analysis.

4.2.2 Capture event properties

Besides the restrictions imposed by the equipment, another main source of variability is the way the tolling event is captured given its surrounding environment, actions taken by the driver and constraints imposed by the act of driving itself.

- **Distance of the vehicle to the camera**: although various stills of the same vehicle may be captured, given that a vehicle is an object in motion when it comes to open-road tolling (ORT), the distance between said vehicle and the camera when each image is captured may vary greatly. This may render some of the stills unusable or diminish the information that can be retrieved from them if the vehicle is either too far away from the camera or too close to it.
- **Angle of the vehicle in relation to the camera**: although the camera is theoretically always focused on the same point (which may not be true in reality due to strong winds or disruptions of any other kind, such as accidents), ORT provides a flexible enough environment for a vehicle to adopt slight changes in position relative to the camera while circulating in a lane, as well as change lanes while crossing the tolling gantry.
- **Complete or partial image capture**: when the tolling event is detected, the images captured by the camera may capture a whole vehicle or only part of it, depending on the way the vehicle is positioned on the road, its distance to the camera and the speed at which it circulates.
- **Luminosity and visibility conditions**: the identification of a car's brand and model can be severely impacted by the surrounding environmental conditions and their effect on the captured images. Luminosity and visibility issues contemplate not only the problems caused by obtaining photos in dark environments, like at night, but also rain or fog, which might obstruct camera vision and/or blur it, or sunlight, which may interact with reflective elements such as license plates and cause glare effects.
- **Vehicle speed**: the speed at which a vehicle circulates impacts the number of images taken of said vehicle and may produce blurring.
- **Presence of other objects**: since roads subject to tolling typically possess more than one lane and are not completely isolated, other vehicles or objects may appear in the captured images at various distances, thus making the identification process more difficult.

4.2.3 Vehicle properties

- **Vehicle type**: vehicles can generally be divided into passenger, light- or heavy-duty, according to the International Council on Clean Transportation and the United States Environmental Protection Agency. Other definitions are possible, depending from each country or state's jurisdiction. For each category considered, there are a number of sub-categories it can be divided into, with the divisions being done mainly by weight and purpose. (For instance, passenger vehicles can be divided into light-duty - also called passenger cars - and medium duty, according to their weight.) [158] These differences introduce a high degree of variability, since many types of vehicles have to be identified. The complexity increases even further should bikes and other motorised vehicles be also considered.
- **Vehicle brand, model and year**: this is the main focus of MMR. For each brand and model, a change in year may mean a completely different appearance, and thus a reliable solution may be to consider a vehicle's model as being the pair $(model, year)$.
- **Vehicle colour**: many algorithms do not take colour into account. However, even if images are converted to grey-scale, different tonalities of black, white and grey may improve or hinder recognition, depending on the algorithm's ability to tell the vehicle from its surroundings and the ease with which the vehicle's contours can be discerned.

- **General condition of the vehicle**: real world vehicles suffer from the passage of time and use, accumulating dirt, suffering modifications when necessary (for instance, a change of tyres, wing mirrors, having roof racks fitted to transport big objects, replacing or hammering dented components, etc.) or being involved in accidents which change the vehicle's appearance.

4.2.4 Analysis and summary

Given the numerous sources of variability, MMR systems should establish scenarios where some of them are considered constant, at least in the initial stages. A system would, therefore, be composed by a conjunction of several subsystems, each with a specific purpose and constants.

From the factors identified above, the easiest to remain constant would be the equipment. By choosing one RSE at a time, be it a single camera or a similar group, the angle and direction in which the image is captured remains practically unaltered, easing the differentiation of images by eliminating or mitigating rotation factors. Of course, vehicles may still be located in different points in the lane they are travelling in, which means the images will not be completely similar. The use of various shots of the same vehicle may lessen this problem, since the system would theoretically learn to recognise a vehicle at various distances and at a small angle range. In order to simplify the workings of the MMR system as a whole, one further step would be to minimise the amount of foreign objects present in the images the model learns from. Finally, when it comes to the vehicles themselves, a possibility would be to define subsystems according to vehicle type. Likewise, since heavily damaged or dirty vehicles could be difficult to identify, images could be limited to relatively clean and undamaged vehicles to begin with.

A summary of the variability issues that an MMR system may face is presented in Figure 4.1.

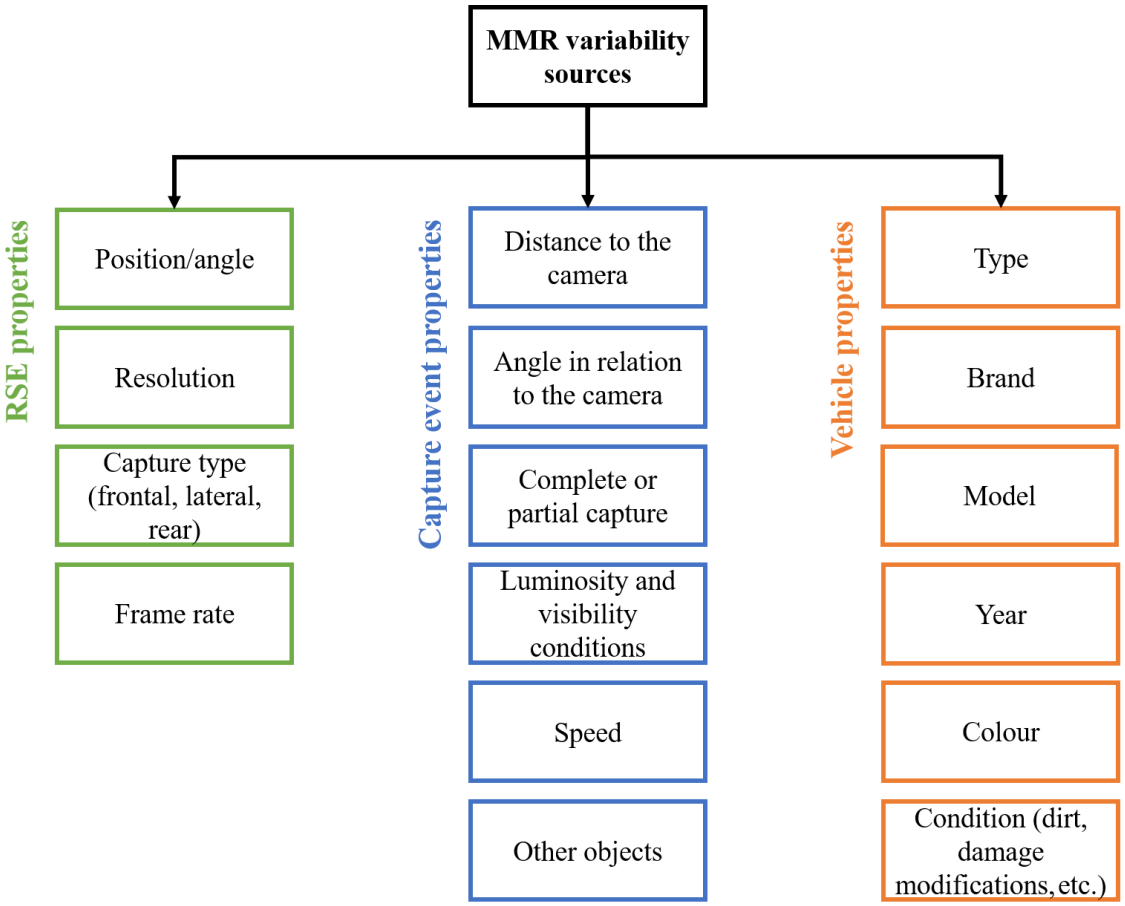


Figure 4.1: Main causes of variability identified for the MMR problem. Left, **green**: RSE properties. Centre, **blue**: image capture event properties. Right, **orange**: vehicle properties.

Chapter 5

Design

5.1 System architecture

5.1.1 Parent system overview

The solution developed by the Tolling Centre of Excellence (TCoE) team relies on three main components: the road-side equipment (RSE), the operational back-office (OBO) and the commercial back-office (CBO). Users can interact with the system via two websites. The first, named public website or PW, aims to provide information and allow interactions with social media, possessing a content management system where the tolling company publishes any information it deems pertinent. In turn, the self-service website (SSW) is used by customers to manage their information and that of their transponders and vehicles, as well as checking their transactions or violations and providing payment details.

The RSE information is handled externally through company ETCC's RITE system [159], generating event and transaction information that is in turn handled by TCoE's solution. CBO is a systems, applications and products (SAP) system developed by Accenture which handles billing and payments, as well as performing account maintenance and management. This project, however, focuses solely on functionalities provided by the OBO component.

OBO is a custom application built by TCoE to process the information generated by RITE, being the owner of the client account balance and information. It is a modular system responsible for the creation of transactions, with loosely-coupled components. OBO possesses, however, a set of core components that are required if any other modules are to function properly, namely:

- Publishing and services: module which allows bidirectional communication between the OBO and CBO components.
- BSMS: module responsible for statistics and alerts.
- Validation and formatting: module that converts information received from external sources into a format compatible with OBO's internal representation.
- PMM: module which concentrates operational data, relationships and status for customer, vehicle and tag information, charging the client's account balance.

Besides these core components, OBO contains a set of stand-alone functional modules:

- RSE event gathering: includes information extraction, transformation and loading (ETL) capabilities, preconfigured controls and the ability to convert RSE events into a standard internal format.
- RSE provisioning: provides back-office to RSE communication capabilities and is prepared by default to create black and white lists and tariff lists, controlling the upload process.

- Intelligent automatic license plate recognition (IALPR): supports manual and automatic license plate reading, using optical character recognition (OCR) to identify the vehicle depicted in an image and supporting human-based validation as well.
- Trip builder: converts toll point crossing events into transactions.
- Reporting: provides a set of reports for each functional module and allows for configuration of further reports if desired.

Of these, IALPR in particular proves to be a major asset in Accenture's tolling solution, having been patented by TCoE's lead supervisors [160]. This process is aimed at combining data from multiple sources in order to identify a vehicle through the use of OCR and fingerprinting. However, neither of these techniques are infallible, having limitations when it comes to low-quality images and producing false positives sometimes. Therefore, IALPR also implements 2nd level OCR as a mitigating strategy, using a second OCR engine from a different supplier. Another mitigating strategy adopted is the use of business rules, which compare and evaluate, for example, the results of reading frontal and rear license plates or the results returned by the 1st and 2nd level OCR engines.

However, accuracy problems are not completely solved by this new layer of automation, necessitating the use of image review (IR) by human operators. Nevertheless, this method of overriding the result can also introduce inaccuracy and increase costs, making the introduction of further automation methods desirable, such as evaluating a vehicle's attributes and behaviour and the context of the trip made.

Overall, the architecture of this application can be described using three layers atop a data model layer, as described by the architecture module view depicted in Figure 5.1.

Element catalogue for Figure 5.1:

- **Matching layer**: the topmost layer is responsible for calculating confidence levels and deciding whether or not there is a match.
  - Automatic Vehicle Recognition Engine: module that holds the algorithms that allow for the computation of a composite value of confidence in the recognition of a vehicle based on the evaluations performed and decides if a match has occurred or not based on the business rules.
- **Evaluation layer**: the middle layer produces the necessary information to augment the knowledge basis.
  - Vehicle Fingerprint Engine: extracts and correlates the vehicle fingerprint data.
  - 2nd Level OCR Engine: performs a second pass over the license plate images.
  - Traffic Behaviour Analyser: evaluates traffic patterns that can be associated with a specific vehicle.
- **Management layer**: this layer contains the Image and Transaction Management APIs, which contain methods to validate system data.
  - Image Management API: system module that performs validation of image data for completeness, consistency and relevance.
  - Transaction Management API: system module that performs validation of transaction data for completeness, consistency and relevance.
- **Data access layer**: comprises the knowledge bases (KB), indexes and metadata of the application, as well as any computed behaviour models. This layer is not truly applicational, serving instead as a persistence unit.

The technical architecture of the IALPR application was set up to be horizontally and vertically scalable, with the evaluation modules separated from the KBs in application server domains/farms.

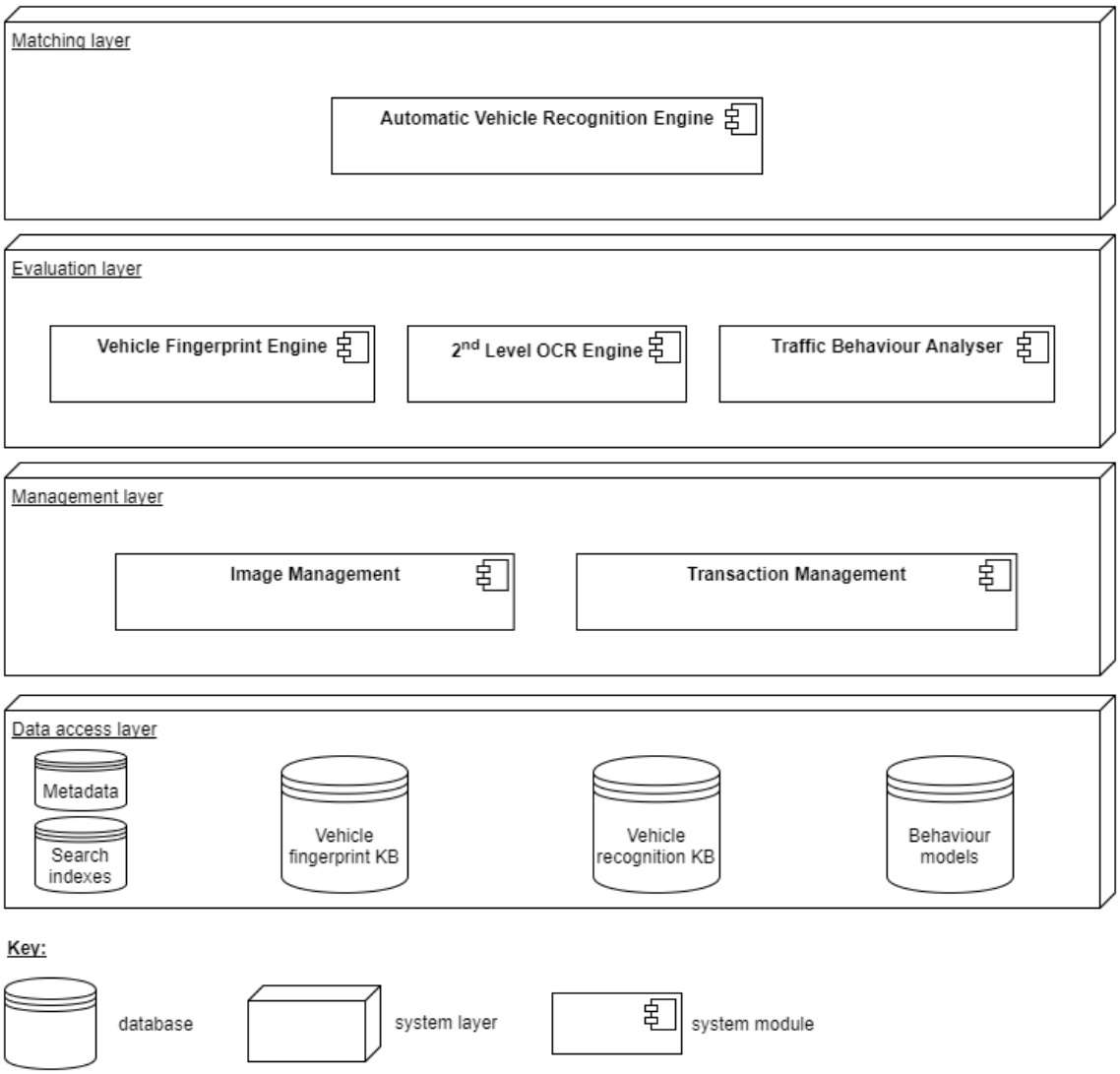


Figure 5.1: OBO architecture: module view of the IALPR system.

Each evaluation module holds an instance of the management APIs in order to perform independent processing. The image data and files are stored in a specific separate infrastructure (typically a storage area network, or SAN). The allocation view of the architecture is described in Figure 5.2.

When it comes to the implementation, the application flow is depicted in Figure 5.3 and described as follows:

Element catalogue for Figure 5.3:

1. Initial Data Validation: performed in the management layer by the management APIs. This activity validates the incoming transaction and image data to verify completeness, consistency and relevance so data can be posteriorly evaluated.
2. Fingerprint Extractor: corresponds to the detection and extraction of interest points and image features from a picture.
3. 1st Level OCR: extracts the vehicle's license plate characters, mask, jurisdiction and plate type.
4., **6.** LPN Derivatives: calculates a set of LPN similar to the identified one. May use

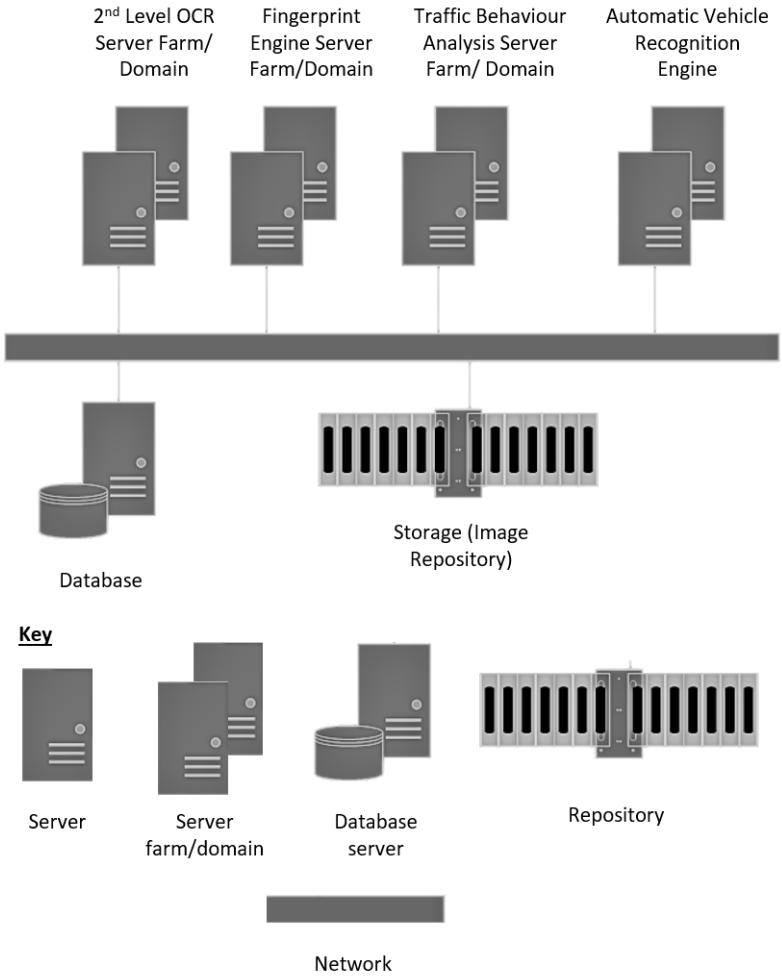


Figure 5.2: Allocation view of the IALPR system.

grammatical inference algorithms based on Hamming or Levenshtein distances.
5. 2nd Level OCR: extracts the vehicle's license plate characters, mask, jurisdiction and plate type. Produced by a different vendor than the 1st Level OCR.
7. Business Rules: this activity evaluates if the front plate matches the rear one with a confidence level above the established threshold.
8. Fingerprint Matching: compares image features from different images in order to detect if the same object is present. This activity includes:

a. Region Interest Detector: algorithm used to detect license plates in a picture, which can be parametrised to detect other regions of interest.
b. Fingerprint Candidate Selector: algorithm used to select the most appropriate fingerprint from the knowledge base to perform the comparison against the new one.
c. Image Normalisation: set of algorithms used to increase image quality and improve fingerprint and 2nd level OCR results.
d. Fingerprint Knowledge Base Builder: algorithm used to create and maintain the fingerprint knowledge base.

9., **10.**, **14.**, **15.** Matching Evaluator: set of algorithms based on fingerprinting, OCR, RSE and all other components of the IALPR that evaluate and decide the identification output, based on a fuzzy Bayesian network.

11. Adjacent Toll Point Analyser: set of algorithms that, based on positive identification on an adjacent toll point, help improve the recognition confidence level.
12. Vehicle-based Behavioural Analyser: set of algorithms that, based on historical traffic patterns, help improve the recognition confidence level.
13. Vehicle Attributes Analyser:set of algorithms that, based on vehicle attributes (e.g. make, model or colour), help improve the recognition confidence level.
16. Human Image Review: activity performed if a decision cannot be reached within the defined confidence level threshold, in which the images of the vehicle are sent for revision by a clerk in order to be identified.
17. Knowledge Base Augmentation: activity that extends the knowledge base information using the results of the evaluation, in order to further tune the evaluation capabilities of the system in the future.

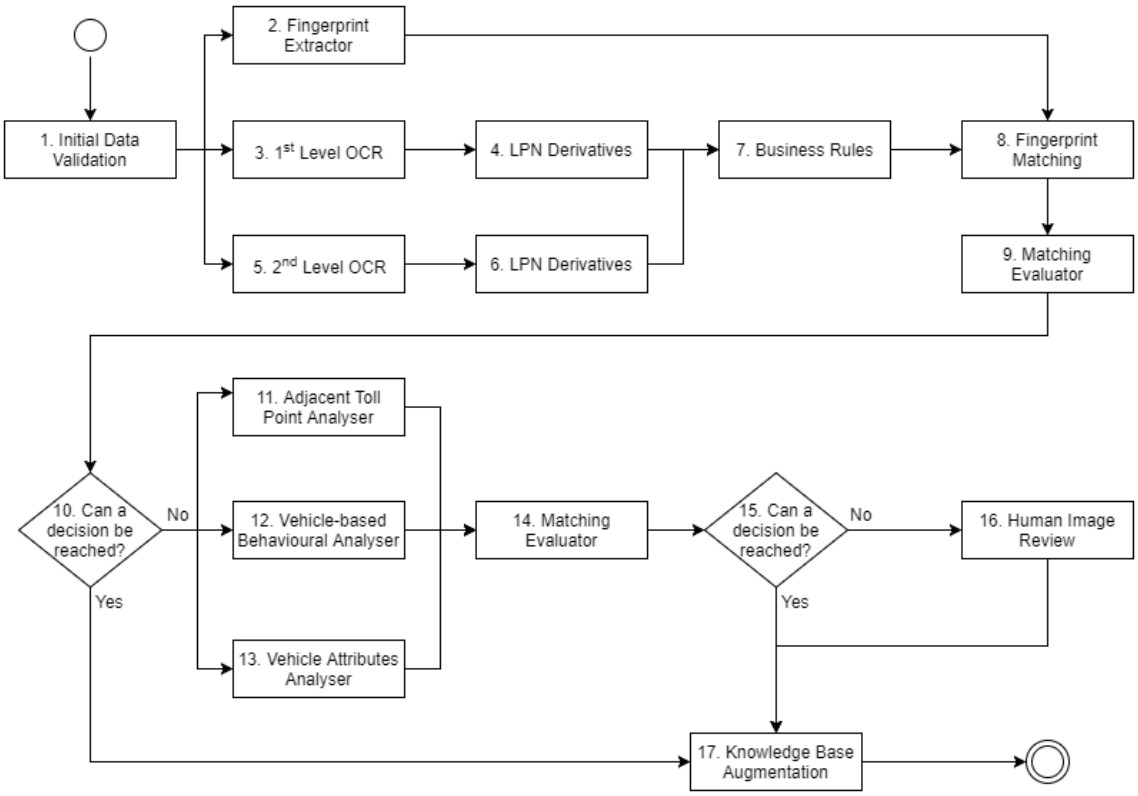


Figure 5.3: Application flow of the IALPR system.

5.1.2 Architecture

The present work aims to design a new stand-alone module complementary to IALPR's OCR that lessens the likelihood that an image will be sent to be reviewed by a human operator, thus lowering the operating costs. The current system's architecture was designed so that it could blend as modularly and effortlessly as possible with the existing architecture. Therefore, the module view remains the same as described in Figure 5.1, with the only addition being a new module (Vehicle MMR Engine) being inserted into the Evaluation layer, as depicted in Figure 5.4.

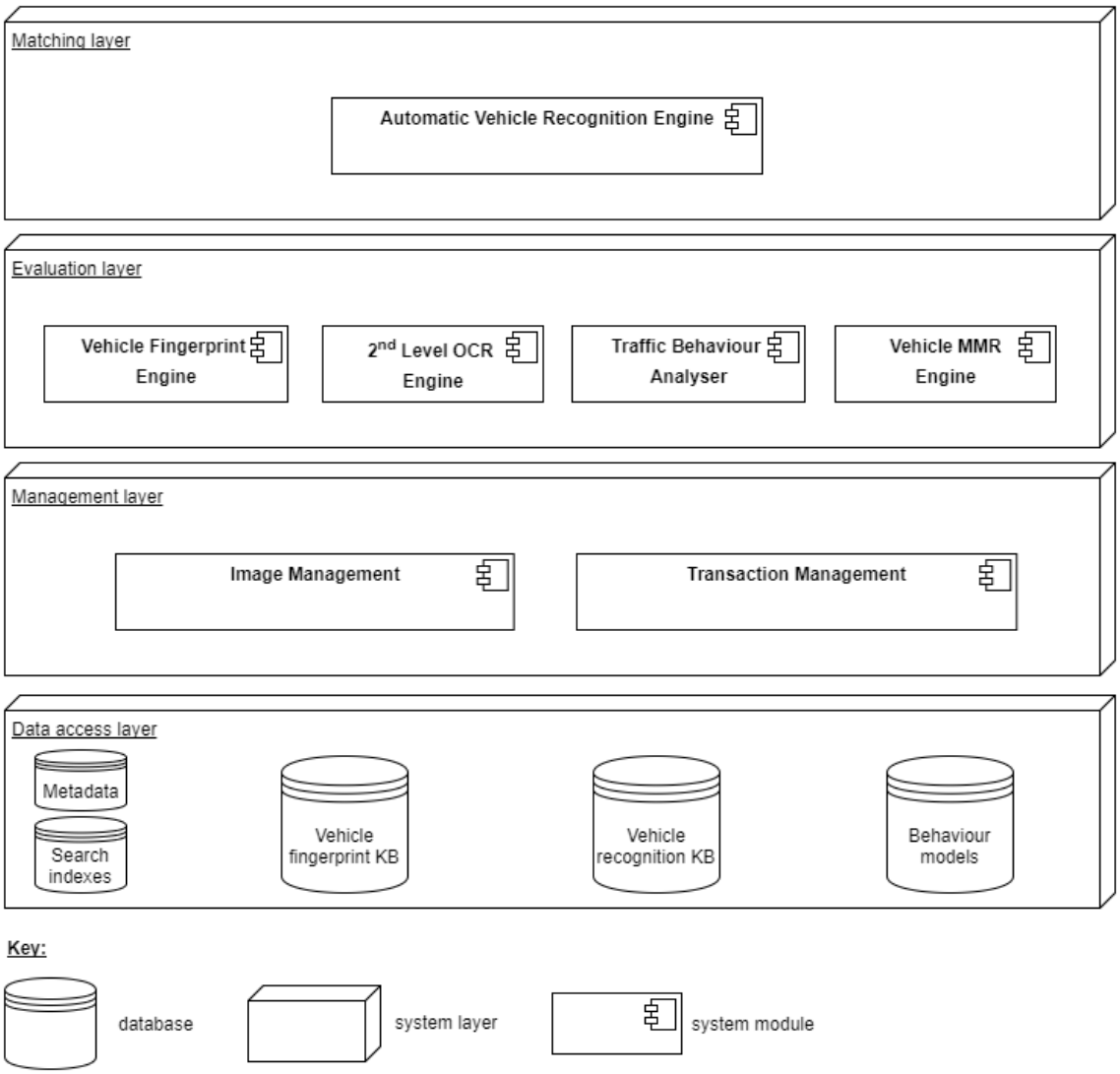


Figure 5.4: Module view of the modified IALPR system.

Element catalogue for Figure 5.4:
The element catalogue is identical to the one described for Figure 5.1, save for the new module introduced, Vehicle MMR Engine.

– Vehicle MMR Engine: responsible for extracting and evaluating data pertaining to the make and model of a vehicle.

When it comes to the deployment view of the system, the less costly solution would be to somehow integrate the MMR component into an existing server farm. However, this would potentially compromise the performance of the existing system, something undesirable according

to the requirements established for this new system. Therefore, the solution found relies on the deployment of a new server farm/domain specifically built for hosting the MMR services. The image repository and system database will remain the ones previously used, so that any other OBO application may access this data.

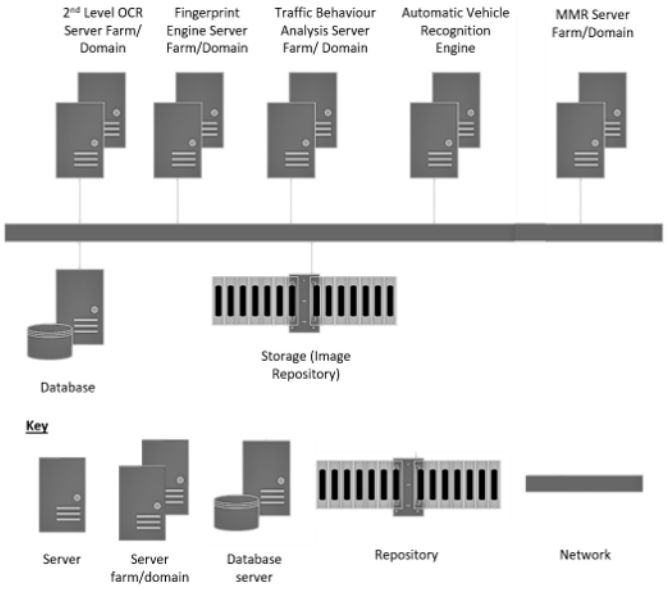


Figure 5.5: Allocation view of the modified IALPR system.

The application flow of the new system deviates from the original one significantly, introducing a new parallel branch of computation to allow for the calculations associated with MMR, as described in Figure 5.6.

Element catalogue for Figure 5.6:
Described below are the elements added to the flow when compared to Figure 5.3. Since there were six elements inserted, starting after number 8 (Fingerprint Matching), the elements numbered **9**–**17** in Figure 5.3 were renumbered to **15**–**23** in Figure 5.6. Nevertheless, their functions remain the same, and so the only elements described in this element catalogue are the following:

9. Manufacturer Extractor: corresponds to the detection and extraction of a region of interest in an image for the logo region.
10. Manufacturer Derivatives: calculates a set of logos similar to the identified one in the image.
11. Business Rules MMR: evaluates whether the vehicle back and front logos match with a confidence level above the established threshold.
12. Matching Evaluator: algorithm that evaluates and decides the manufacturer classification output.
13. Model Extractor: corresponds to the detection and extraction of multiple regions of interest in an image for identifiable characteristics in a model, namely grill, wing mirrors and lights.
14. Make and Model Matching: compares results from different images to evaluate whether the results are the same and rules out impossible make and model combinations.

The MMR application does not, however, stand on its own in terms of technology, needing the aid of external APIs in order to function. Such is the case of the Google Tensorflow API, used

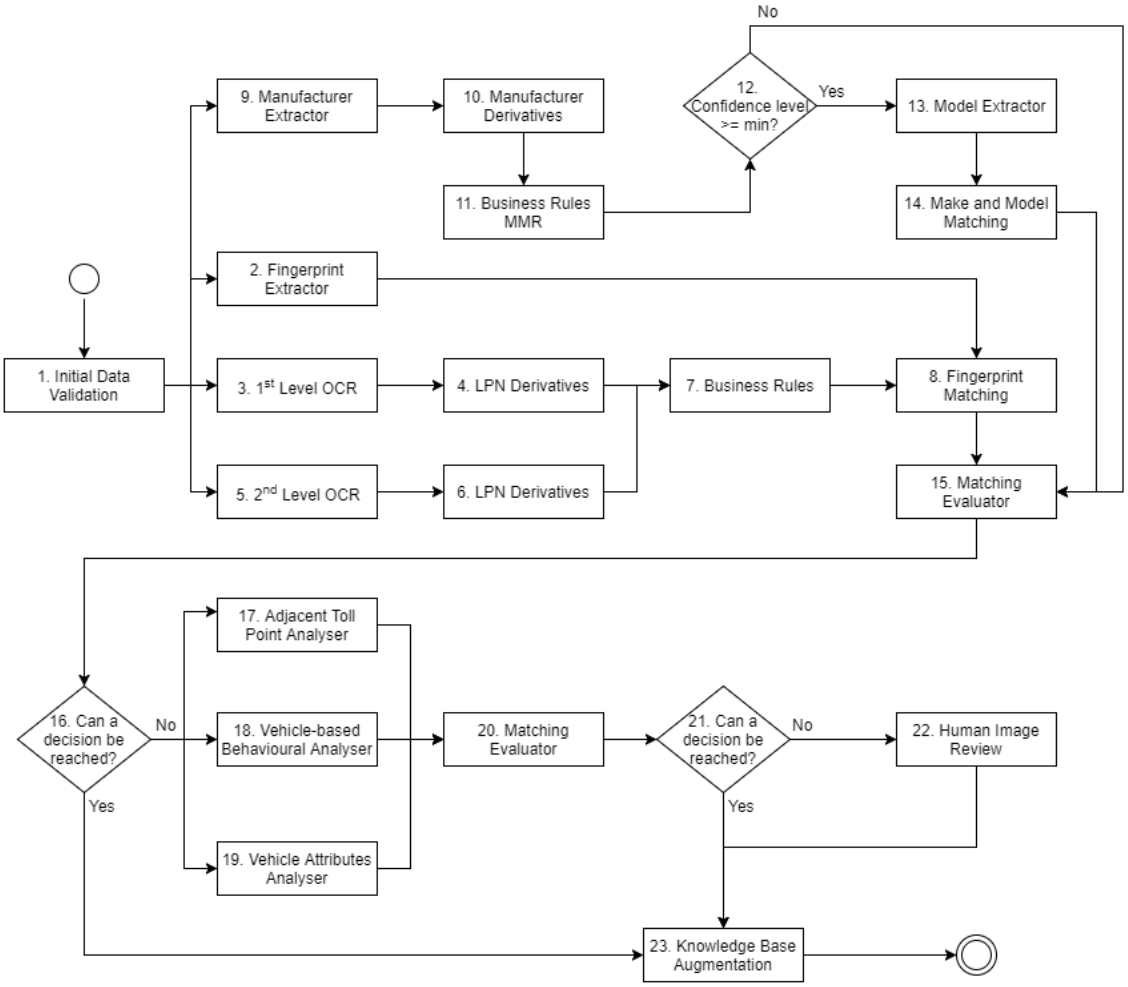


Figure 5.6: Application flow of the modified IALPR system.

to efficiently compute machine learning models, and the OpenCV API, which is used in image processing and fingerprinting.

The interactions established between the system and these APIs are described in the top-level service-oriented architectural view depicted in Figure 5.7.

Element catalogue for Figure 5.7:

Most of the modules described in this top-level SOA view have already been documented in Figure 5.1, so the focus falls on the description of the new components depicted in this view.

- Vehicle MMR Engine: see the element catalogue for Figure 5.4.
  - Manufacturer Recognition module: extracts a region of interest and identifies the vehicle's manufacturer through its logo.
  - Model Recognition module: extracts regions of interest and identifies the vehicle's model through its wing mirrors, grill and lights.
  - Make and Model Matching module: makes a decision about the vehicle's make and model based on the results obtained in the previous two models and excludes impossible results (e.g. if a vehicle's manufacturer is Volkswagen, its model cannot be Astra, an Opel model).
- Google Tensorflow API: external API that provides machine learning capabilities.
- OpenCV API: external API that provides image processing capabilities.

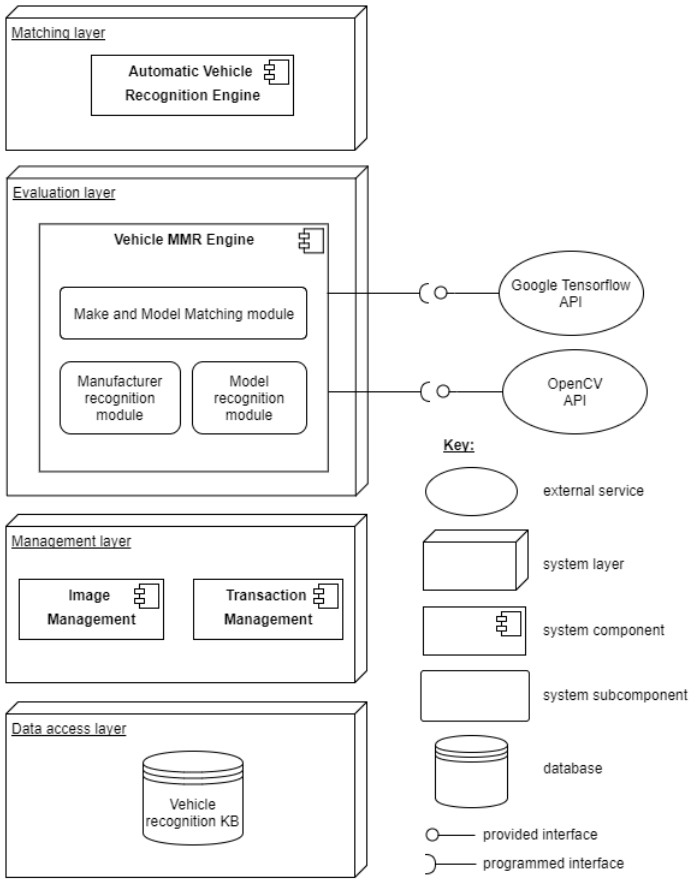


Figure 5.7: Top-level SOA view of the modified IALPR system. For convenience purposes, only some modules are described for each layer, while the others are omitted (see Figure 5.4).

5.2 Implementation considerations

TCoE's existing OBO solution relies on Java 1.7 for its implementation. However, since this work's main focus rests on the employment of machine learning techniques for vehicle make and model recognition purposes, and upon review of the existing documentation, it was concluded that this language might not be the best choice for implementing the MMR components, due to its high level of abstraction hindering performance, as well as the lack of available APIs for the TCoE team's libraries of choice, OpenCV for image processing and Tensorflow for the prospective use of neural networks, as recommended by internal supervisor Dr. Ricardo Pinto. Therefore, Python was chosen as the implementation language, given its fulfilment of the requirements above, as well as the amount of machine learning libraries available and its relative ease of learning when compared to languages such as C++.

Although this step will not be put into practice in the current work, it is nevertheless important to establish how this new code would be integrated into the existing one. The physical establishment of a new server farm discards more low-level implementations such as process-launching from the same machine, which would be impossible to escalate properly. A possible approach would be to make these new functionalities available through a REST API, hiding their implementation details from the caller. This would be a viable approach for communication between OBO and this new MMR system - OBO could simply make a POST call with the image and receive the most probable classification in return, as well as the confidence level for said classification. However, different functionalities within the MMR unit must also be taken into consideration, since,

although the classification of a single image could be done almost simultaneously, this image must also be fed to the existing classification system in order to fine-tune it. Each image must be classified in near real-time so as not to compromise the existing ALPR system, so a possible solution would be load-balancing synchronous requests across a number of distributed server nodes for optimal performance.

Causing the system to recalibrate itself with the arrival of each new image would be impractical, if not completely impossible, since most machine-learning methods are time-costly when it comes to calculating models. A possible solution to allow for the improvement of this knowledge-base without exhaustive use of resources could be batch processing of images - for instance, images that arrive after the start of the new execution of model fine-tuning could be stored temporarily or placed in a queue, since this activity would be inherently asynchronous, which calls for an event-driven architectural style. The recalibration process could then be run using a cron job, which would allow this recalibration to be triggered at the most convenient time for the system (for example, overnight or during the weekend).

Moreover, the pre-existing model would have to be available while the new one was calculated, which makes the use of redundancy advisable. In fact, redundancy and distribution of the entire system would be ideal, were it not for the increasing levels of difficulty in maintaining such a system over extended periods of time and still making it scale efficiently. A possible mitigation plan would be the use of cloud providers such as Google Cloud Platform (GCP), Azure or Amazon Web Services (AWS), which allow for more flexible allocation and management of resources and, in the case of GCP, allow native support for the development and maintenance of Tensorflow models.

5.3 Code quality and performance indicators

5.3.1 Code quality

The current project aims to establish a novel MMR solution for TCoE's tolling projects. Therefore, the tools used to ensure the development and quality of any code produced for the MMR component should match the ones currently used by TCoE's teams.

The TCoE tolling projects use static code analysis to enforce the quality of the code produced by the teams. Static code analysis consists in analysing the source code for both structure and content, ensuring that all code adheres to the industry standards and best practices, and detecting security threats, bugs and maintainability issues [161].

SonarLint was the tool chosen by TCoE to perform static code analysis, being integrated in their IDE of choice, Eclipse. Operating under the basis that poor code quality causes a variety of issues, such as low team velocity, application decommissioning, crashes in production or bad reputation, developer company SonarSource established their tools with three types of issues in mind [162]:

- Maintainability (**code smells**): issues relating to the effort of adding to the code in terms of ability, cost and time it would take to perform those changes. These issues are sometimes referred to as technical debt.
- Reliability (**bugs**): issues that deal with operational risks or unexpected runtime behaviour, generally caused by critical programming errors and/or non-use of best practices.
- Security (**vulnerabilities**): issues pertaining to the existence of flaws that can be exploited to make the code behave differently from its intended purpose.

Complementing the use of this code analyser, TCoE also uses the following assortment of quality project and code management applications:

- **Arquillian**: testing platform for the JVM that enables developers to create automated integration, functional and acceptance tests for Java middleware [163].
- **Git/SourceTree**: git is a widely used open-source, free software version control system, with SourceTree being a commonly-used GUI [164, 165].
- **Jenkins**: open-source automation server, used for project building and deployment [166].
- **JON**: acronym for JBoss Operations network, middleware manager and monitoring system [167].
- **Nexus**: centralised repository manager [168].
- **Phabricator**: toolset platform for software development which includes applications for code review, repository hosting, bug tracking and project management [169].

5.3.2 Performance indicators

The use of performance indicators and measurements in software development is essential to obtain critical and objective information about the software itself, its performance and the decisions that need to be applied to it [170]. An MMR system, being highly centred on image recognition, should rely heavily on machine learning (ML) and/or image fingerprinting. Therefore, the use of ML performance indicators was applied.

Chapter 6

Implementation and results

6.1 Stanford dataset

The Stanford cars dataset was put together by Krause et al. (2013) and consists of 16185 images of cars, roughly equally divided into a training set containing 8144 images and a test set which comprises the remaining 8041. The vehicles displayed in the images of this dataset assume a variety of poses in relation to the camera, with frontal, lateral, rear, bird's eye view, lower view and many combinations thereof being represented. These cars belong to 196 different classes or pairings $(brand, model, year)$, which are referred to as the car's model in this context, and represent 49 different brands, distributed as described in Figure 6.1 [171].

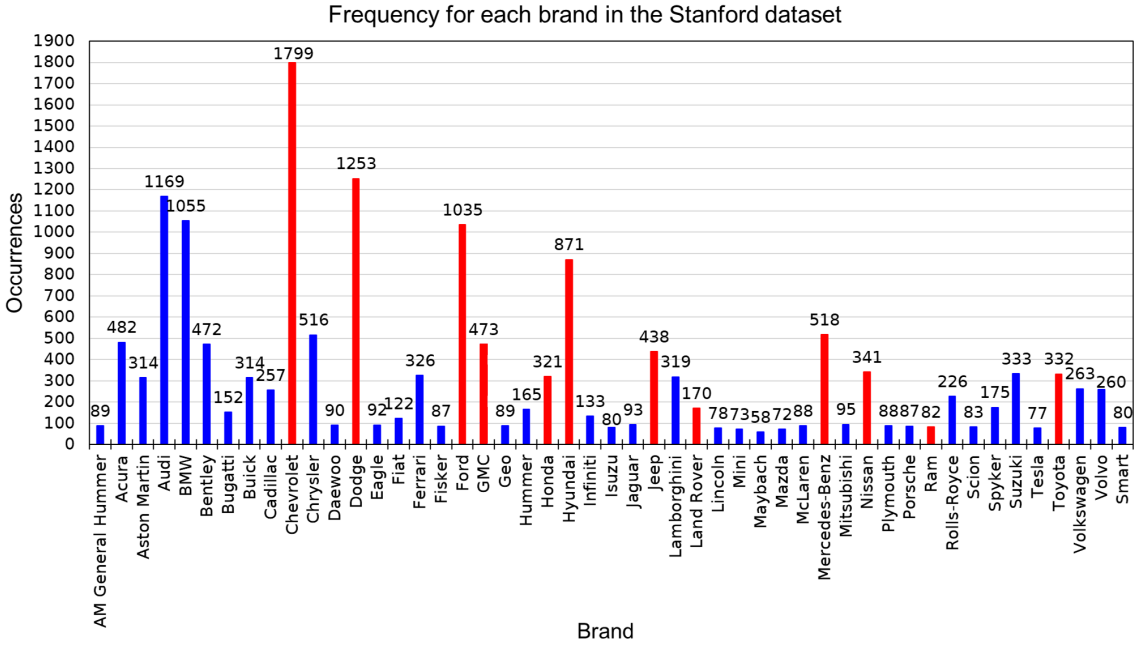


Figure 6.1: Distribution of the images in the Stanford dataset according to the vehicle's brand. The most sold brands in 2006 and 2016 are depicted in **red** and the remaining ones in **blue**. (Based on [171–173].)

Since the number of classes and images was very high and would theoretically require costly computation times, a sample of 12 brands was used for performing evaluations on this dataset. This sample was determined by analysing reports on the most purchased cars in the US in 2006 and 2016 and determining the top vehicles' brands in order to judge which ones would be more

representative. These brands are depicted in Figure 6.1 in **red** bars [172, 173].

A more thorough analysis for the entirety of the Stanford dataset can be found in Appendix A, Table A.1.

6.1.1 Fingerprinting

The first application of the Stanford images was through the use of image fingerprinting with AKAZE features [174].

The main goal of this approach was not to find exact image matches, as would be conventional in fingerprinting, but instead to determine whether the presence of multiple images, taken in various angles, of vehicles belonging to the same model, would allow for successful matches according to the specificities inherent to each model that could contain points of interest (for instance, grills, lights or wing mirrors). Therefore, in this stage all the images from the brands represented by **red** bars in Figure 6.1 (Chevrolet, Dodge, Ford, Honda, Hyundai, Jeep, Land Rover, Mercedes-Benz, Nissan, Ram and Toyota) were used, independently of the angle the vehicle was depicted in.

These images were then assessed on vehicle brand/make and model. For that purpose, 1000 images belonging to the Stanford test set were converted to grey-scale and subject to matching with every image in the training set. This matching was performed in three different ways:

1. Brute-force matcher with ratio test (as proposed by Lowe (2004) in his SIFT paper [109] and illustrated in Figure 6.2)
2. Brute-force matcher with cross-check
3. FLANN matcher with ratio test

Matcher type Two kinds of matcher were considered in accordance with the literature. The first one was a brute-force (BF) matcher. BF or naïve matchers take a descriptor in a set and match it with another in a second set by trying each descriptor in that set, applying distance metrics and choosing the closest one [175, 176]. On the other hand, Fast Library for Approximate Nearest Neighbours (FLANN) matchers contain a collection of algorithms optimised for nearest neighbour search in large datasets and for high dimensional features, being therefore faster than BF matchers [177]. FLANN matchers use randomised *kd*-trees or hierarchical clustering trees for better performance [176].

Restrictions to the neighbour candidates When it comes to the neighbours found for a keypoint descriptor, in order to limit the amount of possible matches, two types of restriction methods were used. Firstly, Lowe's ratio test was applied. In his SIFT paper, Lowe considered that correct matches needed to have the closest neighbour significantly closer than the closest incorrect match in order for reliable matching to be carried out. This author then conducted an experiment on real image data, using a database of 40000 keypoints and calculating the probability density functions for correct and incorrect matches in terms of the ratio of closest to second-closest neighbours of each keypoint. Lowe concluded that matches should be discarded when the distance ratio was greater than 0.8, thereby eliminating 90% of the false matches while discarding less than 5% of the correct ones. This experiment is illustrated in Figure 6.2 [109].

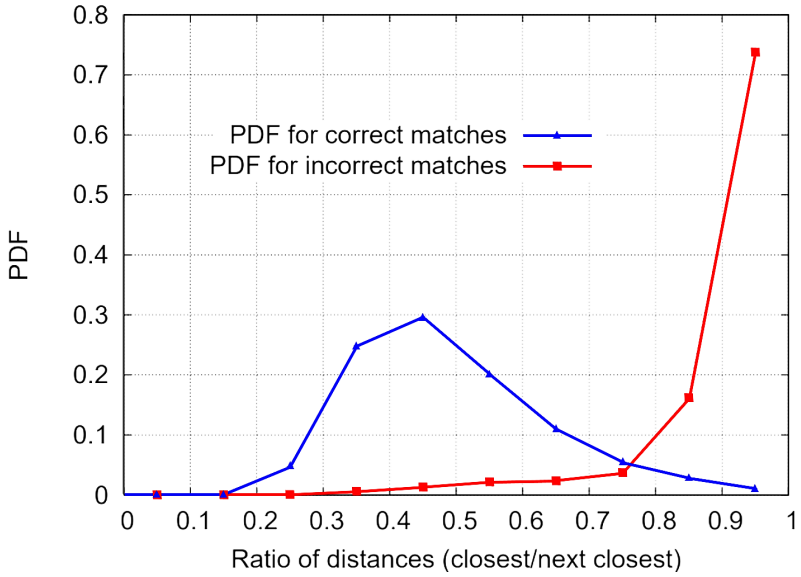


Figure 6.2: Ratio test method developed by Lowe (2004), showing the probability density functions (PDF) of correct (**blue** triangles) and incorrect (**red** squares) matches according to the distance ratio for the closest neighbour relative to the second closest one. (Adapted from [109].)

The second match-discarding technique consisted in using OpenCV's boolean built-in `crossCheck` parameter, which, when set to True, alters the behaviour of the BF matchers. In this case, the matcher will only return (i, j) pairs that have the j^{th} descriptor in the collection as the nearest to the i^{th} query descriptor, and vice-versa, i.e., `crossCheck` causes only consistent pairs to be returned as matches. It should be noted that, since FLANN matchers create an indexing search tree, removing nodes can be hard or even impossible to do, necessitating retraining of the data structure. Therefore, FLANN matchers do not take this parameter [177].

Implementation In order to simplify the analysis, each of the brands belonging to the Stanford dataset was encoded with a number from 1 to 49. The 12 brands considered for fingerprinting with AKAZE features were then extracted, yielding the following list:

10 Chevrolet	**22** Honda	**35** Mercedes-Benz
13 Dodge	**23** Hyundai	**37** Nissan
18 Ford	**27** Jeep	**40** Ram
19 GMC	**29** Land Rover	**46** Toyota

For each brand, all the available models were considered, returning 91 vehicle models that can be consulted in Appendix A, Table A.2.

The evaluations were performed by computing the keypoint descriptors for each image and using, in turn, a brute-force matcher and a FLANN matcher to perform the matching. Since AKAZE computes binary descriptors, the distance metric chosen for the BF matcher was the Hamming distance. The amount of matches was then reduced by applying Lowe's ratio test or cross checking between the descriptors, in order to discard matches where the two nearest neighbours are not too dissimilar to each other. Through the analysis of Figure 6.2, and taking into account that the ratio chosen by Lowe was 0.8, which already showed some probability of incorrect matching, a slightly lower ratio of 0.7 was chosen. A second ratio value of 0.4 was also tested for comparison purposes, since although it would theoretically halve the probability for correct matches, it also rendered the probability of incorrect ones almost null.

Finally, classification was accomplished through ordering the matches obtained for the image and choosing one in accordance to the ordering function used. Three approaches were attempted. In the first, **maximum value**, for each test image, the number of matches for each training image (m) were added and then sorted in descending order. The first image in that list (the one possessing the highest number of matches) was then considered the resulting match, and its label taken as the test image's classification, as described in Equation 6.1 [107, 177].

$$BM_{MV} = max\{m\} \tag{6.1}$$

The second approach, **minimum absolute difference between ratios**, took into account the number of keypoints for the image being matched (kp_{test}) as well as the number of keypoints for each image it was compared against (kp_{train}), by establishing ratios between the number of matches for an image pair (m) and the number of keypoints produced by each image. Therefore, two fractions were calculated for each pair $(testImage, trainImage)$. The evaluation was then performed according to Equation 6.2. The rationale behind this method is that two images would be most similar when both fractions approached 1, meaning each keypoint in the test image would have a match in the training image and vice-versa, and hence taking the absolute difference between the ratios and minimising it would yield better results than simply taking the highest number of matches. Nevertheless, this approach is not without fault, since the objective of these assays is not to find exact image matches, but instead enough similarities that two images could be considered to belong to the same class. In fact, even the same image in different resolutions could pose a problem due to the likelihood of there being a different number of generated keypoints. Likewise, in order for this approach to work, instances where no match was found have to be discarded prior to its application.

$$BM_{AD} = min\left\{\left|\frac{m}{kp_{test}} - \frac{m}{kp_{train}}\right|\right\} \tag{6.2}$$

A further attempt, **maximum match to keypoint ratio**, was performed using solely the information pertaining to the image being tested, as shown in Equation 6.3. This method of choosing the classification result eliminates the need to explicitly take the images the sample is compared against into account by prioritising the highest fraction of keypoints that are being matched in the test image. Therefore, a higher ratio would mean the test image would bear more resemblance to the image the comparison is made against. However, this method does not guarantee the matches being made are satisfactory and translate into real similarities between the two images. Likewise, the comparison against an image that yields a lower number of keypoints when compared to the test image will automatically limit the number of matches that can be performed and may cause that matching to be unduly discarded.

$$BM_{MR} = max\left\{\frac{m}{kp_{test}}\right\} \tag{6.3}$$

Results and discussion The various techniques mentioned were combined to yield 15 different assays, whose results are displayed in Table 6.1.

Table 6.1: Results of the fingerprinting assays for the Stanford dataset.

Matcher type (BF, FLANN)	Match selection (R,CC)	Ratio used	Classification selection	Assay No.	Accuracy		Cohen's kappa	
					Brand	Model	Brand	Model
BF	R	0.7	MV	1	0.128	0.007	-0.098	-0.002
			AD	2	0.091	0.009	0.041	-0.002
			MR	3	0.104	0.020	0.092	0.011
		0.4	MV	4	0.168	0.112	0.045	0.102
			AD	5	0.152	0.083	0.099	0.073
			MR	6	0.168	0.112	0.125	0.102
	CC	N/A	MV	7	0.095	0.005	-0.064	-0.003
			AD	8	0.098	0.009	0.045	-0.002
			MR	9	0.091	0.005	-0.064	-0.003
FLANN	R	0.7	MV	10	0.000	0.000	---	---
			AD	11	0.000	0.000	---	---
			MR	12	0.000	0.000	---	---
		0.4	MV	13	0.000	0.000	---	---
			AD	14	0.000	0.000	---	---
			MR	15	0.000	0.000	---	---

Key: BF = brute force, R = ratio, CC = cross check, N/A = not applicable
MV = maximum value, AD = minimum absolute difference between ratios,
MR = maximum match-to-keypoint ratio

An initial analysis of Table 6.1 shows that the results obtained through the use of FLANN did not bear any fruit, whereas the use of a brute-force matcher did not achieve an accuracy higher than 20%, with the best value being 16.8% for a ratio testing of 0.4, both using the maximum value($MV(r = 0.4)$) and maximum match-to-keypoint ratio ($MR(r = 0.4)$) ordering functions. On the other hand, the minimum value obtained (9.1%) also appears twice on the table, once for $AD(r = 0.7)$ and then again for $MR(cross - check)$. In fact, the following conclusions can be extracted when it comes to the brute-force matching attempts:

Vehicle brand: $\mu_{ACC} = 0.122 \pm 0.033$

1. There was little distinction between the three ordering approaches when taken across the three algorithms used ($r = 0.7$, $r = 0.4$ and cross-checking). The average accuracy values obtained were of 0.130 ± 0.037 for the maximum value function, followed by 0.121 ± 0.041 for the maximum match-to-keypoint ratio function, and finally yielding 0.114 ± 0.033 for the minimum absolute difference between ratios.
2. A more noticeable distinction occurred when accuracy results were compared within the same algorithm and across the three ordering functions. The use of a ratio of 0.4 stands out when compared to the remaining two approaches, with an average accuracy of 0.163 ± 0.009. The remaining two approaches are similar in comparison, with the use of a ratio of 0.7 seeming slightly more advantageous than cross-checking results (0.108 ± 0.019 vs. 0.095 ± 0.004).

Vehicle model: $\mu_{ACC} = 0.040 \pm 0.048$

1. Averaging the results of each of the three ordering approaches did not yield reliable results, with the highest result being 0.046 ± 0.058 for the maximum match-to-keypoint

ratio algorithm. The low accuracies obtained were also reflected in small values of kappa, which invariably fell in the "slight" agreement range, per Landis and Koch [60]. Another characteristic of the model evaluations for MV, AD and MR was the fact that, for each one, the standard deviation was higher than the mean value, indicating a very spread-out distribution of values. Therefore, it can be inferred that a general approach of grouping results in this fashion cannot lead to reliable conclusions.

2. More interesting results were obtained when averaging the values according to the use of ratio-testing value or cross-checking. This analysis shows, once again, that the use of $r = 0.4$ produces higher accuracy results (0.102 ± 0.017) when compared with the use of a ratio of 0.7 or cross-checking, which demonstrates some agreement with the results obtained for brand classification. In fact, the best accuracy values for the prediction of vehicle models were obtained for the same assays that yielded the highest results for brand classification (assays 4 and 6, pertaining to the MV and MR ordering functions, respectively).

The results obtained for the fingerprinting can be explained by a number of factors discussed more thoroughly below. This discussion is mainly centred in the conclusions for brand classification that can be surmised from: (i) the preferred type of matcher, (ii) the use of each ordering function, (iii) the most advantageous alternative between ratio and cross-checking, and (iv) the general effectiveness of this approach to the current MMR problem.

Matcher: The only type of matcher to produce any kind of result was the brute-force one, with the FLANN matcher being unable to produce matches between images presented to it. These results may be explained by two main factors. Firstly, the use of a FLANN matcher requires the use of a number of parameters that describe the algorithm used for nearest neighbour detection, which would have to be fine-tuned to the current problem and would probably have been handled better by someone with more experience in the field. Another contributing factor may have been related to the type of detection desired, since the current work aims to match images based on a few local features and not as a whole. For this purpose, a brute-force matcher will inevitably yield more combinations of possible matches, although they may not be as polished or precise as those yielded by FLANN. It should also be noted that, due to FLANN producing an indexed search tree, matches considered almost aleatory are unlikely to happen when using that algorithm.

Ordering function: When it comes to the ordering functions used, the results for the brand evaluation were very similar to one another. Nevertheless, taking into account that the best values were obtained for the approaches corresponding to the two highest averages, it can be surmised that the second algorithm presented (minimum absolute difference between ratios, or AD for short) is the weakest approach, and thus the less advisable one when ordering matches. This issue can be understood when one considers that similar images at different resolutions may yield a vastly different number of keypoints, shifting the absolute difference of ratios from its ideal value of zero. It should also be considered that a value near zero may also be the result of very poor matches, such as those where only one point in each image has a match, which would cause the fractions to be already tending to zero to begin with. Therefore, the use of this ordering function is ill-advised.

Ratio/cross-checking: The use of Lowe's ratio testing proved overall more effective than the cross-checking approach, both in brand and model recognition. One possible explanation for the cross-checking algorithm to fail to provide good results could be due to the absence of true matching images (i.e., the images being matched were not in fact different representations of the

same image or even of similar ones in most cases). However, the ratio proposed by the literature did not fare much better, averaging an accuracy of 0.108 ± 0.019 for the brand classification (against the 0.095 ± 0.004 achieved by the use of cross-checking), and 0.012 ± 0.007 (vs. 0.006 ± 0.002) for model recognition. The altered ratio relatively to what is proposed by Lowe may also be explained by the absence of true matches in most images, which instead bore small degrees of similarity meant to be detected as matches (for example, a vehicle logo or grille shape as opposed to the whole vehicle). It may be that the results for the use of CC and $r = 0.7$ are not very different because the first acted by eliminating matches that were not considered rigorous enough for a typical fingerprinting assay, such as unidirectional ones, while the second allowed too many false positives to stand for correct matches. Therefore, by lowering the ratio as a means of experiment control, a more flexible option was inadvertently provided, since more false matches were discarded from the potential result set. A more desirable ratio could also be determined had more ratio values been used for comparison, which could serve as a basis for further work.

Classification analysis for the best result: From the assays undertaken in these attempts, numbers 4 and 6 obtained the best results for accuracy and kappa score in the brand scope, with assay number 6 ($MR(r = 0.4)$) also producing a higher kappa value for models. Its brand confusion matrix is presented in Table 6.2.

Table 6.2: Confusion matrix detailing the results obtained for brand classification in assay 6 (ratio-testing with $r = 0.4$ and maximum match-to-keypoint ratio ordering function). The shaded entries in the diagonal correspond to true positive matches.

		Predicted												
		10	**13**	**18**	**19**	**22**	**23**	**27**	**29**	**35**	**37**	**40**	**46**	*Total*
Real	**10**	48	13	11	10	1	10	3	0	2	2	1	5	*106*
	13	14	26	7	5	2	8	3	0	8	0	0	3	*76*
	18	13	4	22	2	6	9	1	2	0	0	1	5	*65*
	19	4	4	3	9	1	5	0	0	1	0	0	0	*27*
	22	1	2	1	0	3	4	0	0	0	1	0	2	*14*
	23	8	7	7	4	2	17	0	0	4	3	0	4	*56*
	27	6	2	6	1	1	5	9	1	4	1	0	0	*36*
	29	3	0	2	1	1	1	0	3	0	0	0	0	*11*
	35	4	5	3	2	2	6	1	0	19	3	0	0	*45*
	37	0	2	4	1	1	2	0	2	1	2	0	0	*15*
	40	1	0	0	0	0	0	0	0	0	0	2	0	*3*
	46	2	1	1	0	2	4	1	1	0	0	0	8	*20*
	Total	*104*	*66*	*67*	*35*	*22*	*71*	*18*	*9*	*39*	*12*	*4*	*27*	*474*

The confusion matrix for the brand results obtained for this assay shows that the numbers of real and predicted images for each of the twelve brands considered in this study are generally similar, which means that Chevrolet (10) was the brand that obtained the most matches, while Ram (40), for example, produced a suitably lower number of classifications. It should also be noted that every brand was correctly classified at least once. However, this is not reflected in the number of true positives, depicted in green, which explains the low accuracy rates. Similarly, from a total sample of 1000 images, only 474 were actually classified, which lowers the accuracy even further. In fact, should only these 474 results be taken into consideration, the accuracy for assay 6 would rise from 0.168 to 0.354, which demonstrates the impact of non-classification in the outcome of this assay, further shown in Table 6.3. Additionally, Ram (40), one of the brands which only produced two true positive matches, was the most difficult to match overall, with 76.9% of

its images not having been classified. In contrast, Mercedes-Benz (35) was the easiest brand to match, with only 34.8% non-classifications, and correctly matching the remaining instances in 42.2% of the cases. These two values stand out in comparison to the rest, although they cannot statistically be called outliers. Ram is especially vulnerable to non-classified images, since the Stanford dataset only contains 82 Ram pictures to begin with, and of those only 3 can be found in the sample used for the assays.

Table 6.3: Frequencies of non-classification for each brand in assay 6.

		Brand											
		10	**13**	**18**	**19**	**22**	**23**	**27**	**29**	**35**	**37**	**40**	**46**
Frequency of non-classification	Absolute	144	81	74	29	20	60	25	11	24	23	10	25
	Relative to real brand total	0.576	0.516	0.532	0.518	0.588	0.517	0.410	0.500	0.348	0.605	0.769	0.556

Overall effectiveness: The brute-force matcher proved to be a poor classifier of vehicle brands, being virtually unable to perform any kind of reliable classification for the model save for two assays (no. 4 and 6). These results are not unexpected, since the images in the Stanford dataset differ wildly in size, resolution, vehicle pose and light conditions, among other factors, which renders the recognition of vehicle models more difficult. Although vehicle brands tend to be distinctive enough for humans to tell them apart, generally through either the logo or grille, the results showed that fingerprinting assays were not the best method to achieve this end, and therefore further attempts using different technologies had to take place.

6.1.2 Cascade classifiers

Through the review of related literature detailed in Chapter 3, vehicle grilles stood out as a possible distinguishing feature. Therefore, a different approach was attempted based on the use of cascade classifiers. In order to avoid bias, the first step undertaken was to generate an independent dataset containing positive and negative examples. The cascade classifiers were then trained using the aforementioned dataset, by making use of Ball and Mehner's tutorial scripts [178, 179] and tweaking them for the desired purpose. Therefore, two kinds of cascade were calculated: faster but less precise LBP cascades, and more accurate but much heavier and more complex Haar cascades. A set of 300 images from the Stanford dataset was then fed to each cascade, which determined the grille ROI when deemed applicable and created a new image from it for further analysis. These outputs were then examined according to a number of factors detailed in Table 6.4.

Table 6.4: Parameters used to classify output images from cascade classifiers.

Code	Name	Description
(a)	Grille present	Did the original image possess a grille?
(b)	Grille detected	Did the output image show the presence of a grille?
(c)	Other detected	Were any other elements detected besides, or instead of, the grille?
(d)	Grille at least partially detected	Did the output capture any part of the grille?
(e)	Grile completely detected	Was the grille detected fully or only partially?
(f)	Only grille detected	Did the output show only the grille or also other elements?
(g)	Grille perfectly detected	Did the output capture the whole grille and nothing else?

Finally, the resulting ROI crops were to be subjected to classification via the fingerprinting techniques presented in subsection 6.1.1.

ROI determination The determination of the vehicle ROI from the cascade output was performed according to two different metrics.

The first one, called the **intersections** metric, consisted in taking the areas judged to be the most likely to contain the ROI by the cascade and choosing the one which intersected more distinct areas. This algorithm seeks to determine the area which most probably contains the ROI as being the one most densely populated by the calculated rectangles (i.e., the rectangle with the most intersections). The rationale for this proposed metric was that a more densely-populated area would be the most likely to contain an actual region of interest.

The second metric evaluated in these assays was named **min-neighbours**, consisting of an adjustable parameter for the sliding concentric window (SCW) approach applied by the cascade classifiers, which specifies how many neighbours each candidate ROI rectangle should possess in order to be retained.

Cascade calculation Each cascade was calculated according to the script shown in Figure 6.3, which makes use of OpenCV's `opencv_traincascade` function as well as Ball and Mehner's previous work for their tutorials [178–180].

```
#!/bin/bash
#create list of positive and negative images
find ./positive_images -iname "*.jpg" > positives.txt
find ./negative_images -iname "*.jpg" > negatives.txt
#create samples from the positive images
perl bin/createsamples.pl positives.txt negatives.txt samples 1500 "
    opencv_createsamples -bgcolor 0 -bgthresh 0 -maxxangle 1.1 -maxyangle 1.1 -
    maxzangle 0.5 -maxidev 40 -w 80 -h 40"
#remove files with 0 KB
find ./samples -size 0 -print0 |xargs -0 rm
#merge all samples into one vector
python ./tools/mergevec.py -v samples/ -o samples.vec
#calculate cascade
sudo ln -nsf /dev/null /dev/raw1394
opencv_traincascade -data classifier -vec samples.vec -bg negatives.txt -
    numStages 20 -minHitrate 0.999 -maxFalseAlarmRate 0.5 -numPos 1000 -numNeg
    600 -w 80 -h 40 -mode ALL -precalcValBufSize 1024 -precalcIdxBufSize 1024
```

Figure 6.3: Code used for cascade calculation, based on Ball and Mehner's tutorial scripts [178–180].

- Preparation of training data: This step was achieved by using Google's image search engine, filtering the search to only yield images that could be reused and modified for both academic and commercial use[1], and finally cropping each image whenever necessary. The positive samples included images of car grilles in various frontal-based poses (Figure 6.4 (a)), whereas the negative ones depicted other car parts (e.g. wheels, windscreen, doors, mirrors, boot, as represented in Figure 6.4 (b1) and (b2)), as well as the surrounding environment (e.g. roads, highways or people, depicted in Figure 6.4 (c1) and (c2)). All the images considered were subject to a step of conversion to grey-scale before being fed to the cascade calculation algorithm.

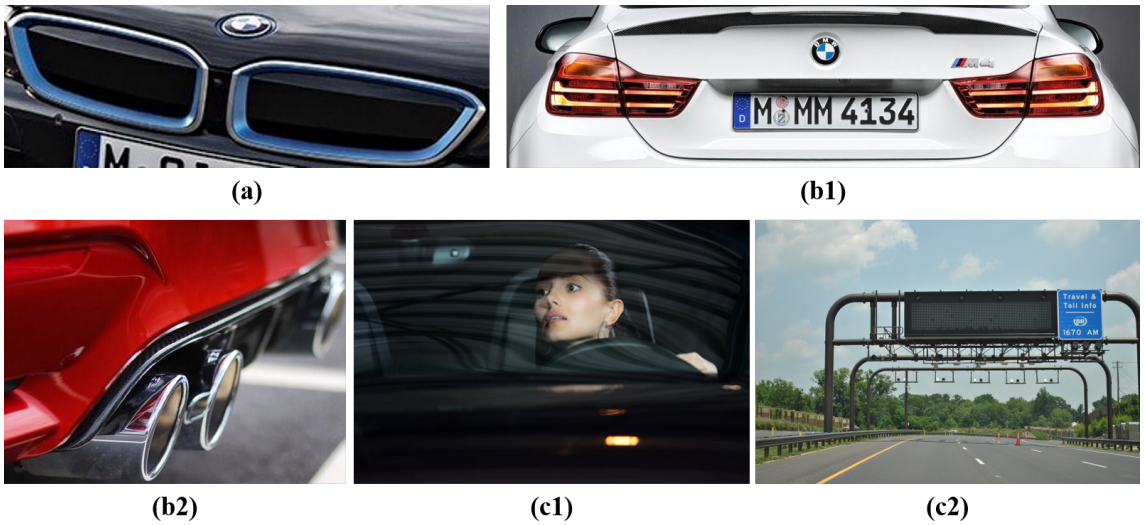


Figure 6.4: Positive and negative examples of images used to train the cascades. (a) Positive example of a vehicle grille. (b) Negative examples representing vehicle components: 1) rear/boot; 2) exhaust pipe. (c) Negative examples representing other undesirable features: 1) driver; 2) road.

Positive samples are generally created from an object image through random rotation, intensity

[1]Usage rights: free to use, share or modify, even commercially.

and background changes, with the degree of randomness being subject to a number of customisable parameters. A set of 1557 **positive** images was chosen to calculate the cascade classifier as foreground samples which represented grilles belonging to vehicles of different brands and in various frontal poses, with Figure 6.4 (a) being an example. These images were then fed into the `opencv_createsamples` utility, being subject to perspective transformation according to the following parameters [180]:

- *maxxangle/maxyangle*: maximum rotation angle (in radians) across the x and y axis, respectively. The value chosen was $1.1rad \approx 63°$.
- *maxzangle*: maximum rotation angle across the z axis, given in radians. The value used was $0.5rad \approx 28°$.
- *maxidev*: maximum intensity deviation of pixels in the foreground samples. The value chosen was 40, as indicated in Ball and Mehner's original scripts [178, 179].
- *vec*: name of the output file.
- *w/h*: width and height of the output samples, in pixels. The values chosen were $w = 80$ and $h = 40$, so that the samples would be twice as wide as they were tall, which is a reasonable approximation of a grille shape. A possible exception to this rule could be, for example, the brand Alfa Romeo, since vehicles belonging to this brand tend to possess a grille in an inverted triangular shape.

For the **negative** examples, two sets of images were chosen, the first containing 637 images and the second 8171. This discrepancy served as a means to study the influence of the amount of background negative samples in the overall quality of detection. It should be noted that the implementations that serve as a basis to this study feature a number of negative examples (600) much higher than the number of positive ones (40), hence why the second attempt was performed [178]. The parameters applied in relation to the negative samples were *bgcolor* and *bgthresh*. The first parameter relates to the background colour, denoting which colour the algorithm will consider as being transparent. A threshold of tolerance may then be defined so that all pixels between $bgcolor - bgthresh$ and $bgcolor + bgthresh$ will interpreted as transparent [180].

- Cascade training: In order to train the cascades, two main algorithms were used, one based on Haar features and another on LBP ones. The samples computed in the previous stage were fed to OpenCV's `opencv_traincascade` function with the following arguments [180]:

- *data*: output directory.
- *vec*: .vec file containing the positive samples, previously created through the use of `opencv_createsamples`.
- *bg*: file containing the background descriptors, which in this case was `negatives.txt`, which comprised the file paths for every image within the negative sample.
- *numStages*: number of cascade stages, or individual classifiers, to be trained. The number chosen for this study was 20, in accordance with the tutorials [179].
- *minHitRate*: minimal desired hit rate for each individual classifier, with the overall hit rate being estimated as $minHitRate^{numStages}$. The minimal hit rate chosen was 0.999, so as to ensure the cascade would be trained as rigorously as possible. Such a hit rate would theoretically yield an overall hit rate of $0.999^{20} = 0.9802$.
- *maxFalseAlarmRate*: maximal desired false alarm rate for each stage, with the overall false alarm rate being estimated as $maxFalseAlarmRate^{numStages}$. The rate chosen was 0.5. This prevents each stage from overfitting, in accordance with the definition of a cascade classifier, whilst ensuring a theoretically good overall performance at a false alarm rate of $0.5^{20} = 9.537 \times 10^{-7}$.
- *numPos/numNeg*: number of positive/negative samples used in each stage.

- *w*/*h*: width and height of the training samples, which must match the values used while creating the samples (i.e. $w = 80$ and $h = 40$).
- *mode*: specified when using Haar-like features, this parameter selects the types of HLF to be used in each stage - `BASIC` for upright features only or `ALL` for the full set of upright and 45 °-rotated features described by Lienhart and Maydt [93]. In order to maximise matches, the `ALL` mode was chosen.
- *precalcBufSize*/*precalcIdxBufSize*: sizes of the buffers for pre-calculated feature values and/or indices, in Mb. The buffer sizes used were of 1024 Mb each, so as to balance speed with the need to not render the machine otherwise unusable, since there were no dedicated servers.
- *featureType*: used to specify whether the features used will be `HAAR` (default) or `LBP`, which needs to be explicitly stated.

Cascade results and discussion Two types of cascades were computed for each method, one for a negative sample containing 637 images (N_{small}) and another for a negative sample of 8171 images (N_{big}).

The first method to be tested was the intersections one, using both Haar and LBP cascades. The results for these attempts for N_{small} are described in Tables 6.5 and 6.6 and Figure 6.5. On the other hand, the results for N_{big} are summarised in Tables 6.7 and 6.8, as well as Figure 6.6.

Table 6.5: Confusion matrix detailing the results obtained for grille detection using the intersections method with Haar features for the N_{small} samples.

		Detected	
		Grille	**No grille**
Real	**Grille**	0.132	0.118
	No grille	0.625	0.125

Table 6.6: Confusion matrix detailing the results obtained for grille detection using the intersections method with LBP features for the N_{small} samples.

		Detected	
		Grille	**No grille**
Real	**Grille**	0.122	0.030
	No grille	0.817	0.032

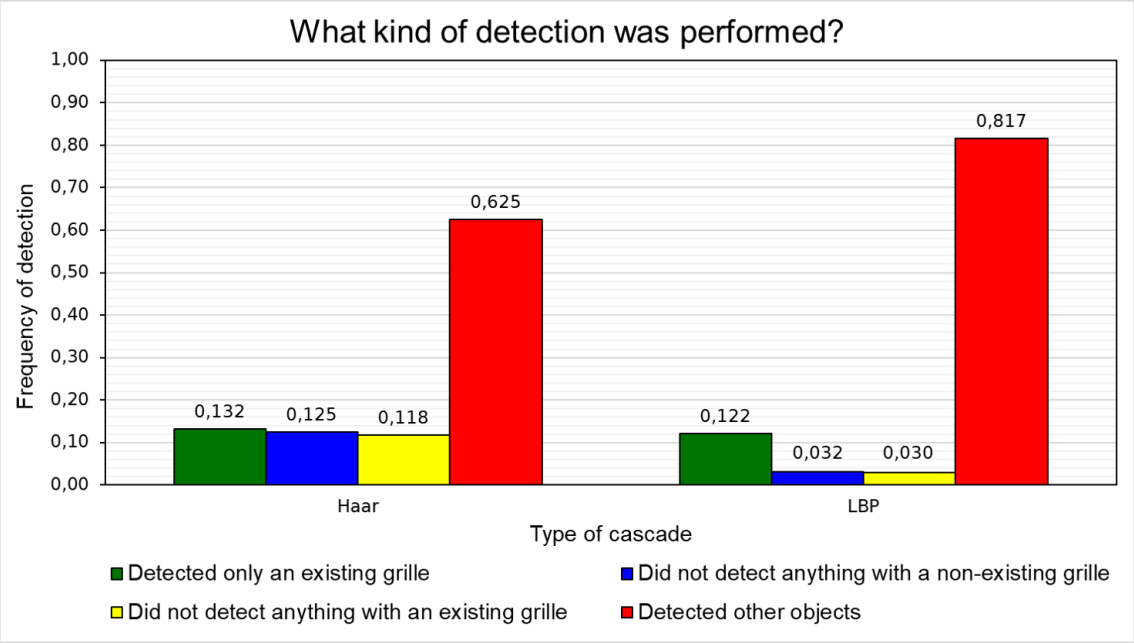


Figure 6.5: Results for the intersections method (N_{small}), comparing Haar (left) and LBP (right) cascades.

Table 6.7: Confusion matrix detailing the results obtained for grille detection using the intersections method with Haar features for the N_{big} samples.

		Detected	
		Grille	**No grille**
Real	**Grille**	0.052	0.070
	No grille	0.798	0.080

Table 6.8: Confusion matrix detailing the results obtained for grille detection using the intersections method with LBP features for the N_{big} samples.

		Detected	
		Grille	**No grille**
Real	**Grille**	0.038	0.032
	No grille	0.887	0.043

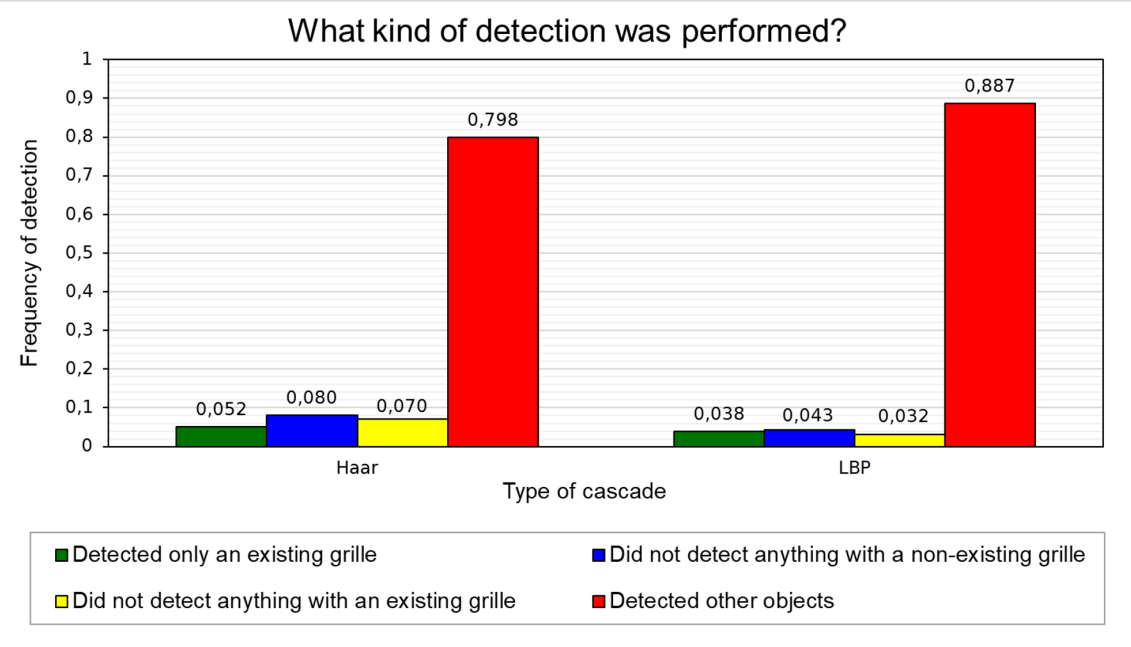


Figure 6.6: Results for the intersections method (N_{big}), comparing Haar (left) and LBP (right) cascades.

An analysis of Tables 6.7, 6.8, 6.5 and 6.6, as well as Figures 6.6 and 6.5, shows significant discrepancies in detection accuracy depending on the number of negative samples used to train the cascade. It should be noted that these results can be translated into metric c (detection of other objects) as described by Table 6.4.

The LBP results for N_{big} show that only 8.1% of the detection was accurate, with 3.8% of true positives and 4.3% true negatives, which deems it inadequate for the detection of car grilles in the given conditions. However, despite some of the available literature stating that a ratio of 1 positive sample for at least 2 negative ones is desirable [178, 181], the results for N_{small} proved better, with an accuracy of 15.4% (12.2% true positives and 3.2% true negatives detected). These results are also consistent with those verified for the Haar assays, in which the N_{big} samples (ACC = 13.2%, TP = 5.2% and TN = 8.0%) ranked poorly when compared to the N_{small} ones (ACC = 25.7%, TP = 13.2% and TN = 12.5%).

An overall analysis of the results for this method shows a large prevalence of false positives in all cases, which would hinder reliable recognition of a vehicle's make and model. Even so, should one of these cascade models be picked for fine-tuning, the N_{small} Haar cascade is the one that shows the most promise, with a higher accuracy and lower ratio of false positives.

The results for the second method used, the neighbours method, are evidenced by Tables 6.9 and 6.10 for the N_{small} assays, as well as 6.11 and 6.12 for the N_{big} ones. Figures 6.7 and 6.8 show the comparison between the two cascade types for N_{small} and N_{big}, respectively.

Table 6.9: Confusion matrix detailing the results obtained for grille detection using the neighbours method with LBP features for the N_{small} samples.

		Detected	
		Grille	**No grille**
Real	**Grille**	0.242	0.003
	No grille	0.747	0.008

Table 6.10: Confusion matrix detailing the results obtained for grille detection using the neighbours method with Haar features for the N_{small} samples.

		Detected	
		Grille	**No grille**
Real	**Grille**	0.323	0.002
	No grille	0.503	0.172

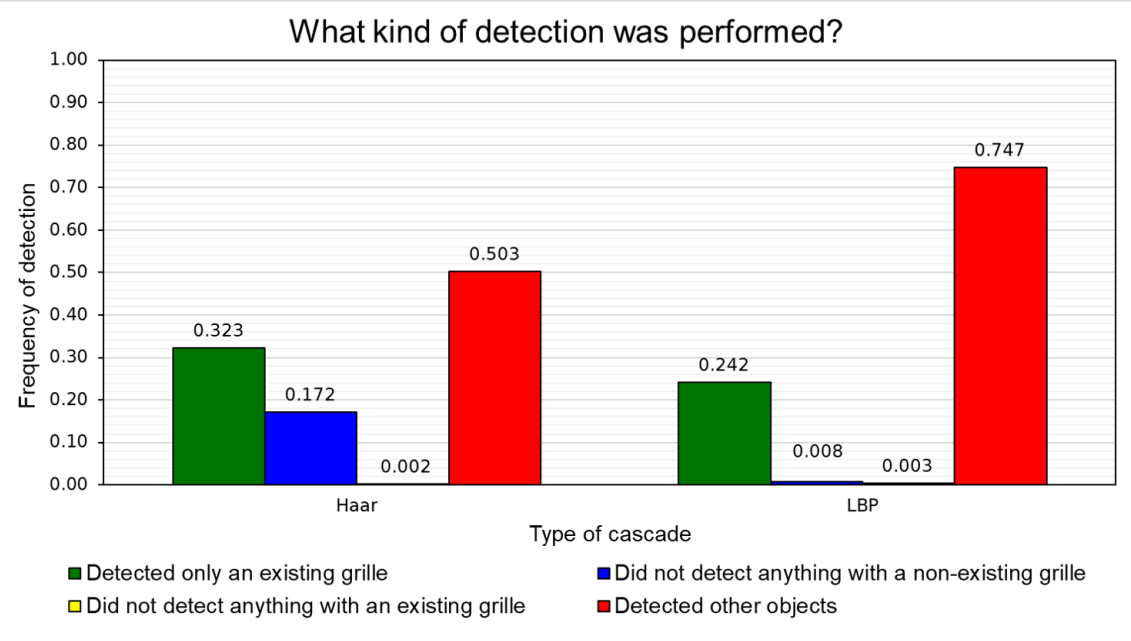


Figure 6.7: Results for the neighbours method (N_{small}), comparing Haar (left) and LBP (right) cascades.

Table 6.11: Confusion matrix detailing the results obtained for grille detection using the neighbours method with LBP features for the N_{big} samples.

		Detected	
		Grille	**No grille**
Real	**Grille**	0.138	0.097
	No grille	0.722	0.043

Table 6.12: Confusion matrix detailing the results obtained for grille detection using the neighbours method with Haar features for the N_{big} samples.

		Detected	
		Grille	**No grille**
Real	**Grille**	0.215	0.018(3)
	No grille	0.743(3)	0.023(3)

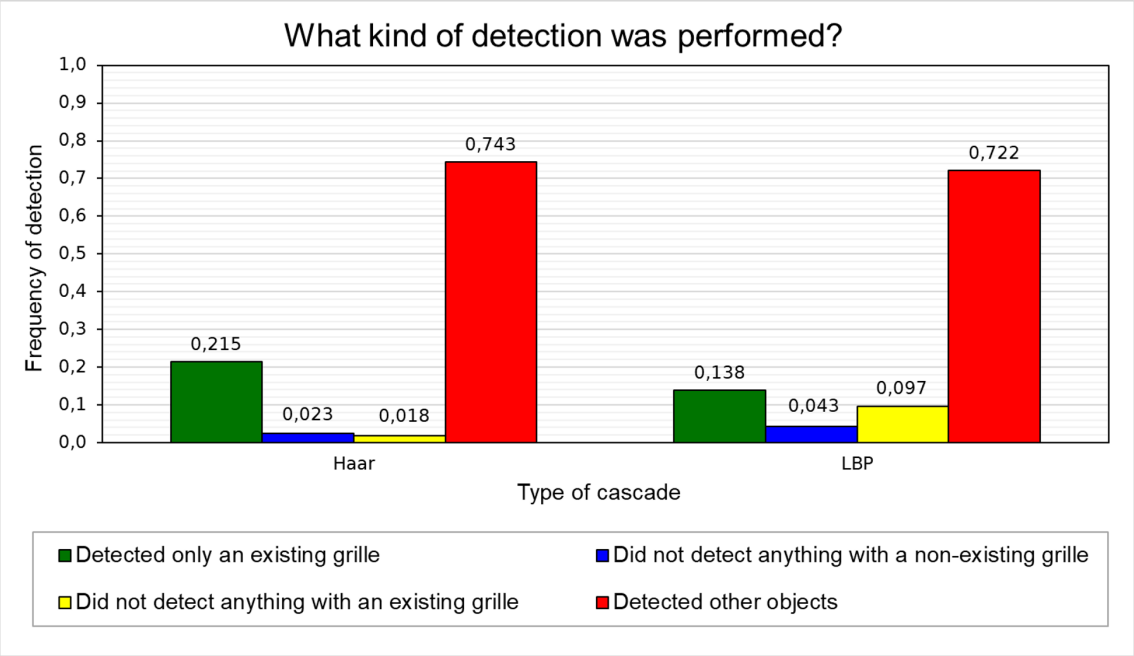


Figure 6.8: Results for the neighbours method (N_{big}), comparing Haar (left) and LBP (right) cascades.

Once again, the difference in negative samples had a significant impact on the results, with a smaller number of negative images producing more advantageous results. One possible explanation for this result is the quality of the data presented as negative. Since the N_{small} negative images were more carefully picked out, it is possible that the newer negative images introduced in N_{big} ended up being not as representative or informative to the construction of the cascade classifier, allowing some noise in the data that seems to have been less critical in the original negative sample pool. The problem in the approach may, therefore, lie not in the quantity of data used, but in its quality, and so more assays would have to be done to consistently verify any claims over the effect of the negative sample size on the quality of the produced cascade.

When it comes to these newly calculated cascades, it is possible to see that, although not satisfactory, these results are nonetheless more significant than the ones obtained for the intersections method. This was an expectable result, since it stems from the comparison of a tried and tested method with a new, experimental one.

For N_{big}, the most reliable cascade algorithm proved again to be Haar, with 23.8% correct predictions (against 18.1% for LBP). The differences experienced in these results corroborate the comparisons between Haar and LBP in the existing literature: whereas LBP is computationally faster, Haar produces more accurate results [182]. The same pattern occurred with the N_{small} negative samples, with the Haar cascade presenting an accuracy of 49.5 % whilst the LBP one achieved only 25.0%. Due to these results, the best cascade to adopt for further experiments would be the neighbours Haar cascade, N_{small} variant. An example of a result for a ROI obtained for this cascade and its LBP counterpart can be viewed in Figure 6.9.

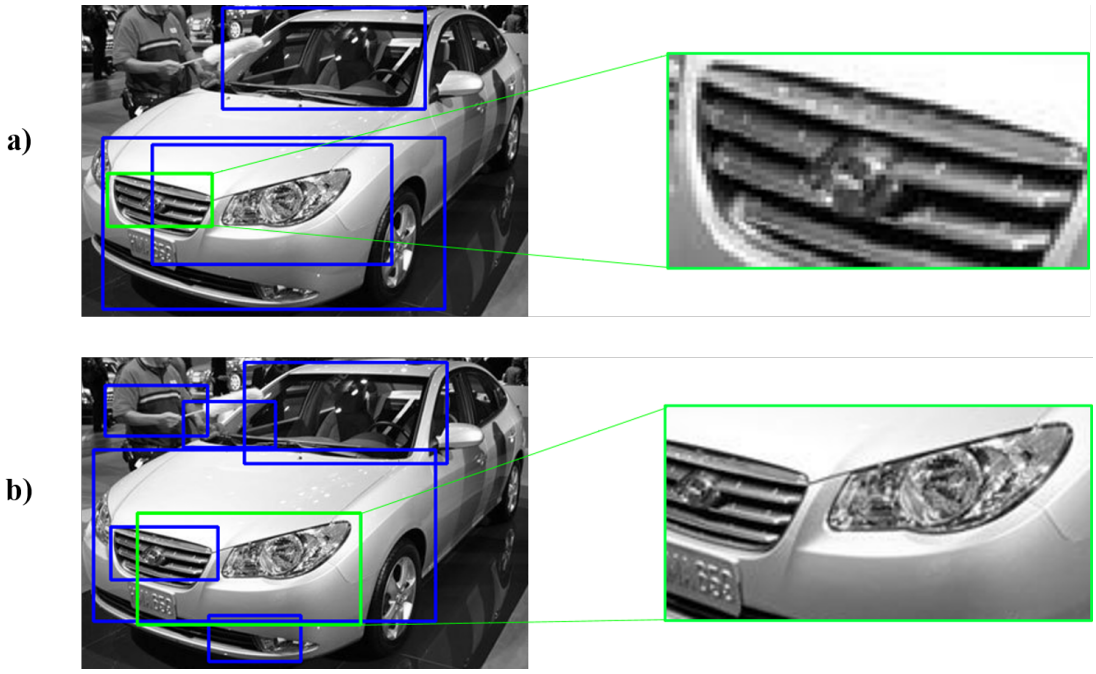


Figure 6.9: Example of ROIs obtained for the neighbours N_{small} method, for the Haar cascade (**a**)) and its LBP counterpart (**b**)). The ROI candidates are noted in the **blue** rectangles, with the selected final ROI being represented in **green**.

In spite of the better results obtained by the neighbours method, these assays proved inefficient, with very high false positive rates. Such results may be attributed to a number of different factors. Firstly, the dataset in itself may not have been specific enough, both in the distinction between positive and negative samples (e.g. car grilles vs. boots) and in the definition of said negative samples. Moreover, the parameters chosen in the sample creation stage may have been too lax, in that high rotation angles in all directions were allowed.

The image outputs were also evaluated in relation to Table 6.4, which sought to appraise the quality of the output in more general terms than those strictly defined by the metrics described above. The results for this analysis can be found in Table 6.13. It should be noted that metrics d), e), f) and g) are valid in relation to the cases where the original image actually contained a grille. The percentage of such images is given by metric a), which is presented at a factual level and will not be analysed in the discussion. Metric b) was the only one calculated given the total set of images.

Table 6.13: Grille image output evaluation.

		Intersections				**Neighbours**			
		Haar		**LBP**		**Haar**		**LBP**	
		small	*big*	*small*	*big*	*small*	*big*	*small*	*big*
a	Grille present	0.650							
b	Grille detected	0.290	0.262	0.330	0.265	0.527	0.372	0.322	0.252
d	Grille at least partially detected	0.638	0.612	0.680	0.618	0.875	0.720	0.670	0.600
e	Grille completely detected	0.157	0.145	0.155	0.137	0.410	0.205	0.097	0.110
f	Only grille detected	0.132	0.052	0.122	0.038	0.323	0.215	0.242	0.138
g	Grille perfectly detected	0.210	0.110	0.102	0.072	0.392	0.148	0.080	0.130

An initial analysis of Table 6.13 suggests that the use of the neighbours method used in conjunction with the Haar algorithm could yield potentially interesting results in regards to grille de-

tection. This superiority is demonstrated specially when using a smaller set of negative examples, with the grille being detected 81.1% of the times when it was present (0.527/0.650). However, a closer analysis indicates that, while this method might be able to pinpoint a general location of a vehicle's grille, it is not reliable enough, picking up a lot of noise along with its actual target. For instance, from the subset of images that depicted a car's grille, only 32.3% detected the grille with no other elements present in the image with enough prominence to warrant a different classification. On the other hand, the use of LBP with the bigger set of negative samples is confirmed to not be advantageous independently of the method being employed.

Given the unreliability of the grille detection methods discussed, the approach was discontinued, with no fingerprinting assays being performed on the detected ROIs to attempt MMR on them. This decision was taken due to the time constraints for the practical work not being compatible with producing a good grille detection cascade, fine-tuning it, and then proceeding to perfect a fingerprinting process on top of its results. Therefore, another, more powerful and complex technique was employed in order to try to achieve working results within the allotted time: convolutional neural networks.

6.2 Proprietary dataset

The Stanford cars dataset proved to be too diverse to be properly handled and analysed in a short timespan, and so TCoE provided a dataset comprising 105634 tolling images from one of its clients, with a resolution of 1392 x 1082 pixels, and captured in grayscale. Since these images belong to that company, representing actual drivers and vehicles, and are not in the public domain, said client's identity has been omitted.

Since this proprietary dataset contains images sourced from the tolling industry, there are a number of variables in MMR that are mitigated through its use - for instance, a vehicle cannot be captured sideways, since the angle variation is limited by the camera's placement relative to the road. The images are also already captured in grey-scale, eliminating the need for further image conversion steps. Altogether, the use of an industry-provided dataset would be helpful with the standardisation of the type of image to be processed, eliminating unwanted degrees of freedom. However, these images had to be labelled by hand, with 18 brands having been recognised. Therefore, in order not to introduce undue human error, only the vehicle's brand was considered when working with the dataset, and images that did not correspond to a frontal pose were removed.

Each brand was encoded with a number from 1 to 18, yielding the following list:

1 Audi
2 BMW
3 Citroen
4 Daihatsu
5 Ford
6 Honda
7 Hyundai
8 Mercedes-Benz
9 Mitsubishi
10 Nissan
11 Opel
12 Peugeot
13 Renault
14 Seat
15 Skoda
16 Subaru
17 Toyota
18 Volkswagen

The distribution of the used pictures in the dataset by brand is depicted in Figure 6.10.

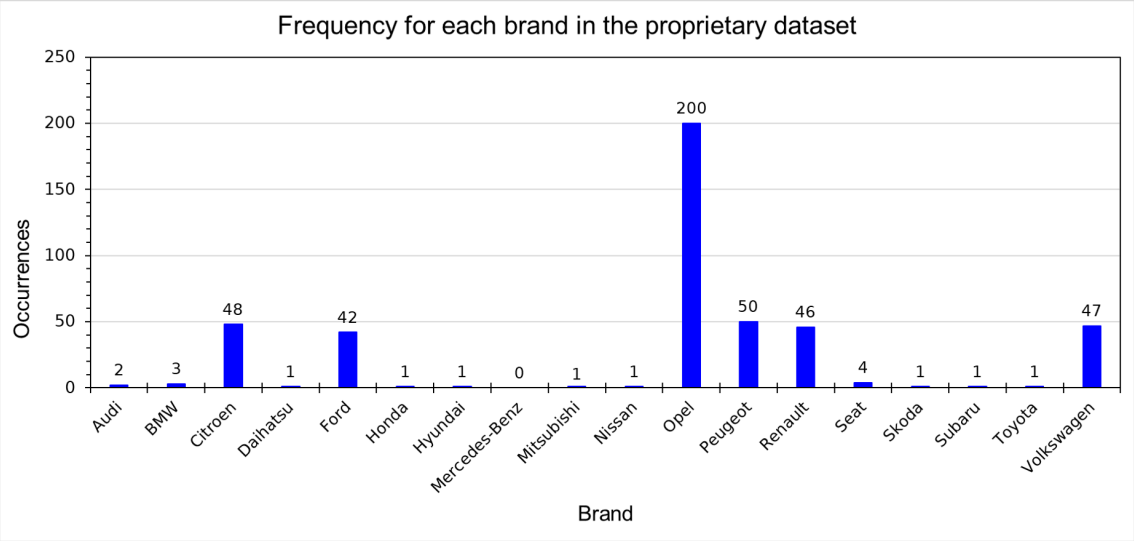


Figure 6.10: Distribution of the vehicles in the proprietary dataset according to the vehicle's brand.

A brief analysis of Figure 6.10 shows that there is an overwhelming prevalence of Opel cars, which amount to 200 of all 450 vehicles (44.4% of the dataset by vehicle). By a small margin, the second most prevalent brand is Peugeot, with 50 vehicles (11.1% of the dataset). It should be noted that there is a brand that does not contain any images (Mercedes-Benz). Its inclusion in the graphic shown in Figure 6.10 is due to all brands having been listed for the dataset prior to the exclusion of rear-end images. So as not to upset the nomenclature already established, Mercedes-Benz was not removed as a label in the front-only assays shown in this work, even though all the images depicting cars of this brand were rear-end ones.

6.2.1 Convolutional neural networks

The difficulty in determining the appropriate features to be used when classifying vehicle images spurred the decision to use a technique that worked as an automatic feature extractor in itself. The proprietary images were therefore used to train and test a set of CNNs, using Tensorflow. Moreover, in order to further diminish the complexity of the task, it was decided that specific brands were to be the focus of these new models, as opposed to a general model to predict every brand.

A number of attempts were performed, of which three will be the focus of this discussion. The first, an Opel-specific model, aimed to classify images as Opel (1) or not Opel (0). The second one worked analogously, but for the Peugeot brand. Finally, a new, more intricate model was developed in order to distinguish between Opel (0), Peugeot (1) and other vehicles (2). The choice in the brands to be tested was not arbitrary, taking into account the two brands with the highest vehicle frequency in the available data pool. The decision to use the brands with the most vehicles instead of those with more total images was made because, even though a brand like Renault possessed more photos than Opel, those photos had less variability, increasing the risk of overfitting the model.

Brand-specific models for Opel and Peugeot vehicles

When it comes to model analysis, the following metrics were calculated:

- Accuracy
- Precision

- Recall
- F_1 score
- Matthews correlation coefficient (MCC)
- Cohen's kappa
- Root-mean-square error (RMSE)
- Receiver operating characteristic (ROC) curve and its respective area under the curve (AUC)

Training and test distributions The Opel-specific model was trained using 300 images and tested with the remaining 150. The training image set was distributed as follows:

- Opel: 100 images
- Other brands: 200 images, of which:
  - Citroen: 45 images
  - Ford: 36 images
  - Peugeot: 36 images
  - Renault: 40 images
  - Volkswagen: 43 images

On the other hand, the test image set was comprised of the following:

- Opel: 100 images
- Other brands: 50 images, of which:
  - Audi: 2 images
  - BMW: 3 images
  - Citroen: 3 images
  - Daihatsu: 1 image
  - Ford: 6 images
  - Honda: 1 image
  - Hyundai: 1 image
  - Mitsubishi: 1 image
  - Nissan: 1 image
  - Peugeot: 14 images
  - Renault: 6 images
  - Seat: 4 images
  - Skoda: 1 image
  - Subaru: 1 image
  - Toyota: 1 image
  - Volkswagen: 4 images

Akin to the Opel model, the Peugeot-specific model was trained using 300 images and tested with the remaining 150. The training image set exhibited the following distribution:

- Peugeot: 100 images
- Other brands: 200 images, of which:
  - Citroen: 45 images
  - Ford: 36 images
  - Opel: 36 images
  - Renault: 40 images
  - Volkswagen: 43 images

Moreover, the test image set was established as follows:

- Peugeot: 100 images
- Other brands: 50 images, of which:
  - Audi: 2 images
  - BMW: 3 images
  - Citroen: 3 images
  - Daihatsu: 1 image
  - Ford: 6 images
  - Honda: 1 image
  - Hyundai: 1 image
  - Mitsubishi: 1 image
  - Nissan: 1 image
  - Opel: 14 images
  - Renault: 6 images
  - Seat: 4 images
  - Skoda: 1 image
  - Subaru: 1 image
  - Toyota: 1 image
  - Volkswagen: 4 images

Convolutional neural network For each model, the images were redimensioned into 128 x 128 pixels each and then two convolution kernels were applied, the first sized 7x7 pixels and the second 5x5. In each convolution layer, 64 filters were used. The convolution layers were interspersed with pooling layers, which used max-pooling to reduce sample sizes. The final layer in the applied CNNs was a fully-connected one with 1024 nodes and subject to a ReLU activation function, designed to extract as much information as possible from the output.

Both the Opel and the Peugeot brand models were subject to the same hyperparameter conditions, with a learning rate of 1×10^{-7} and a `softmax_cross_entropy` loss function. The optimizer chosen was Tensorflow's `GradientDescentOptimizer` and the dropout rate chosen for these assays was 0.4.

Results The first model to be built was the Opel-specific one, which yielded the testing results below.

- ACC = 0.889
- PRC = 0.879
- REC = 0.870
- F_1 score = 0.874
- AUC = 0.923

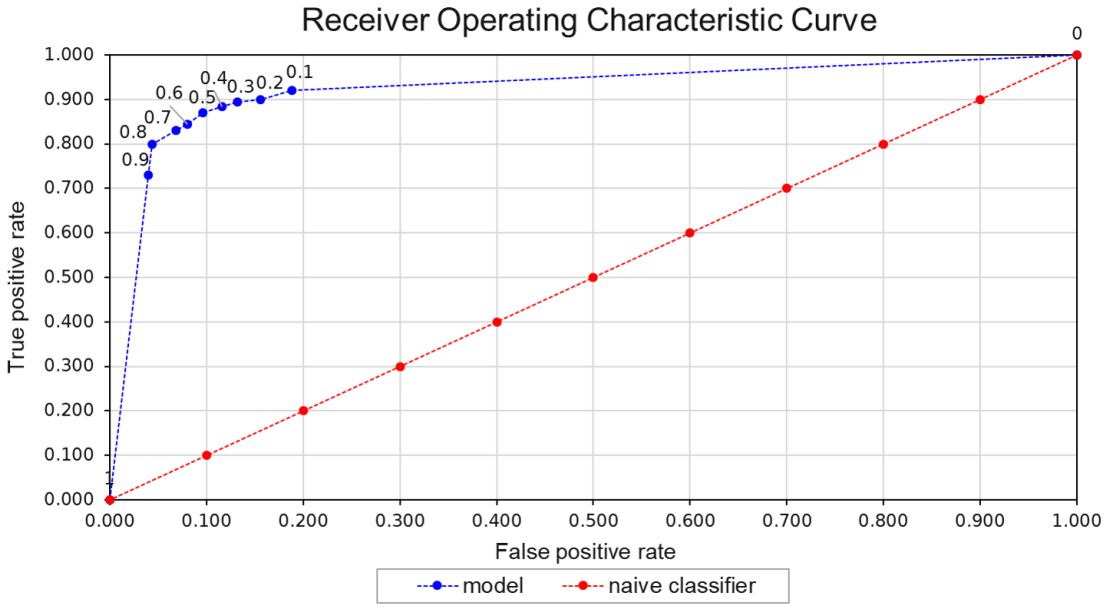


Figure 6.11: ROC curve for the Opel-detecting CNN attempt.

On a first analysis, these results appear very satisfactory, implying a very good generalisation power for this model in its classification of Opel images. However, the accuracy value for this testing set was much higher than the one obtained for the training set, which topped at 0.680. Allied with the nearly perfect results seemingly obtained for the test images, this disparity suggests that the images chosen to test the network may not have been the most suitable, since their classification proved to be too easy for the network. Another possible explanation would be the dropout rate being too large for the classification being performed, artificially making the task of calculating the training accuracy more difficult.

The Peugeot model was calculated afterwards, with no change in the input parameters when compared to the Opel one, and produced the following results.

- ACC = 0.953
- PRC = 0.969
- REC = 0.925
- F_1 score = 0.946
- κ = 0.905
- AUC = 0.974

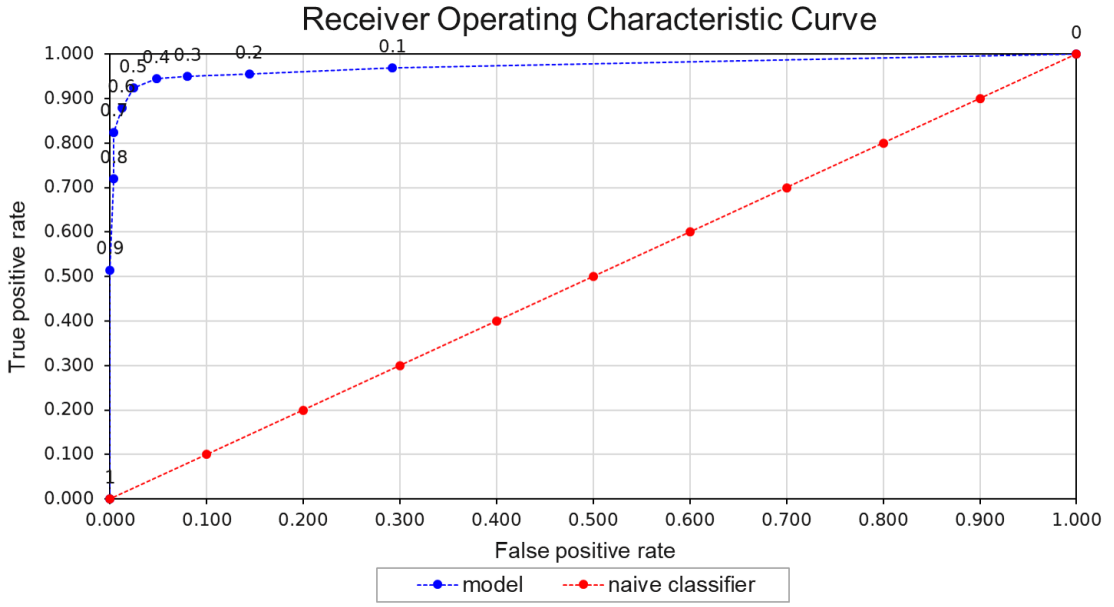


Figure 6.12: ROC curve for the Peugeot-detecting CNN attempt.

Once again, the accuracy obtained for the test image set was much higher than the training one (0.456), with this being an even more unequal case than what happened for the Opel scenario. These results seem to support the hypothesis of the testing set being unfit, and so they were set aside as unreliable and further validation was conducted on both models in order to assess their true classification potential for both brands.

Cross-testing

In order to validate the Opel and Peugeot models, cross-validation tests were carried out. These tests took the 450 images originally used to build the Peugeot model and fed them to the Opel model to be classified and vice-versa. Since not all the images were common in the construction of the two models, this allows for a fair attempt at estimating the generalisation power of each of the computed models.

In both cases, the negative (0) samples correspond to vehicles that do not belong to the brand (Opel or Peugeot) being considered by the model. Therefore, the original labels were adjusted, resulting in a total of 50 images that represented the considered brand (labeled 1) and 400 that did not (labeled 0) for each cross-validation test.

The disparity between the amounts of positive and negative images can be explained by the original composition of each dataset, described in detail above. Since, for instance the focus in the composition of the dataset for the Opel model was in choosing images of vehicles that belonged to the brand and contrasting those with images from other brands, the conversion of the dataset to consider Peugeot vehicles leaves all the Opel images, which represent approximately half of the dataset, as negative examples. On the other hand, the only positive samples are now given by the fraction of the non-Opel images in the dataset that belong to Peugeot vehicles. Consequently, the formerly balanced datasets are now skewed, consisting of 89% negative samples and only 11%

positive ones, which led to the need to evaluate not only the usual metrics (accuracy, precision and recall), but to introduce Cohen's kappa coefficient and Matthews correlation coefficient in an effort to provide normalised evaluations that attempt to minimise this bias.

The results obtained from these cross-validations are detailed separately below.

Cross-test 1: Peugeot images fed to Opel model

- Confusion matrix

		Predicted	
		1	**0**
Real	**1**	44	6
	0	87	313

- $ACC = 0.793$
- F_1 score $= 0.486$
- $RMSE = 0.455$
- $PRC = 0.336$
- $MCC = 0.458$
- $REC = 0.880$
- $\kappa = 0.388$

The first cross-validation consisted of feeding the 300 train and 150 testing images originally used to compute the Peugeot CNN model into the Opel model. For this purpose, each image was re-labeled according to whether or not it represented an Opel vehicle, yielding 50 Opel images (labeled 1) and 400 that represented other brands (labeled 0). It was, therefore, inevitable that the second row of the confusion matrix bore the bulk of the results.

This cross test resulted in an accuracy of 79.3%, meaning that 20.7% of the images were misclassified:

- Type I errors: the amount of false positives (F_P) was 87, meaning there were more non-Opel images being misclassified as Opel than there were actual Opel images in the set. Given this fact, the precision was predictably low, with only 33.6% of the images classified as being Opel vehicles actually belonging to the brand.
- Type II errors: there were 6 false negatives (F_N) in this cross-validation, corroborating the fact that the model has a high recall (0.880). This means that the model correctly identified almost every positive example in the provided set, only misclassifying 12% of the Opel images as belonging to another brand.

Given the misclassification rates, the RMSE for this set amounted to 0.455 and the F_1 score result was 0.486. Both these values point to a average to poor classifier that can only be counted on to determine which images do not belong to the Opel brand, which would be expected given the over-representation of the negative class. Cohen's kappa score also hints that, while these results are better than those produced by a random classifier, they are also not highly trustworthy when it comes to predicting positive values. According to Landis and Koch, this classifier would be a fair one ($0.21 \leq \kappa \leq 0.40$). In turn, the MCC for this test was 0.458, indicating a strong positive relationship between the values predicted by this CNN and the actual ones, but falling short of a satisfactorily correct classification.

Subsequently, the ROC curve for this model was computed. To do so, the output of the classifier that corresponded to the probability of a vehicle belonging to the positive class ($P(1)$) was taken and judged against different thresholds ranging from 0 to 1 in steps of 0.1, with 0.5 being the default one. For said image to be effectively considered as positive (1), the output should, therefore, be higher than, or equal to, the threshold. These results showed that the best outcomes were

located at the far end of the scale, at a threshold of 0.9. Given these conditions, the accuracy would be 0.89, the F_1 score 0.62 and the RMSE would drop to 0.34. For each scenario, the true positive rate ($TPR = \frac{TP}{TP+FN}$) and false positive rate ($FPR = \frac{FP}{FP+TN}$) were computed and the ROC curve calculated as shown in Figure 6.13. The complete results for this analysis can be viewed in the Appendix (Table A.3).

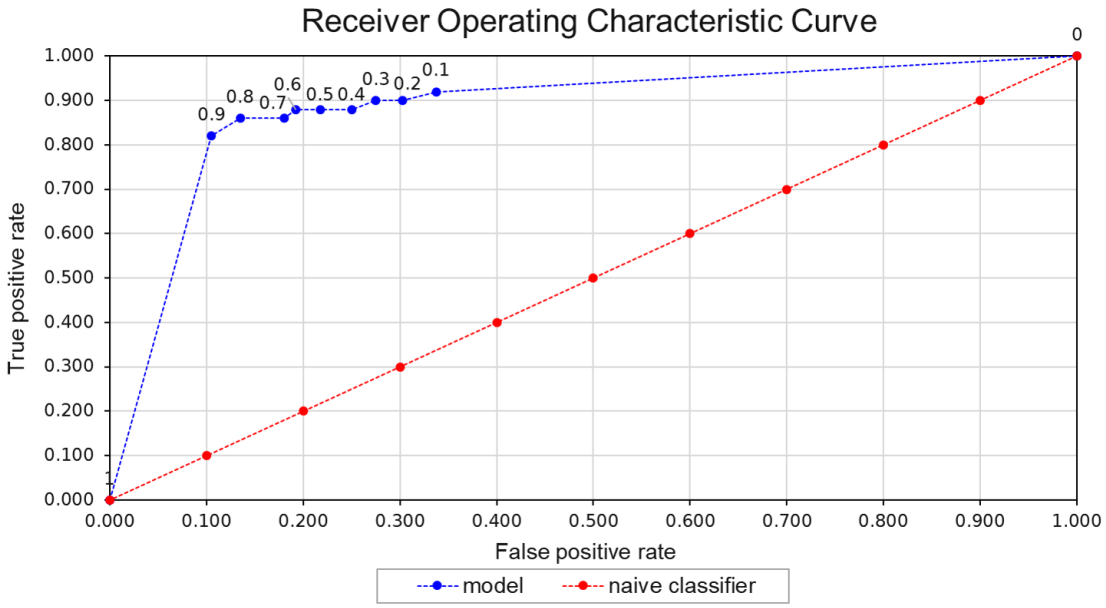


Figure 6.13: ROC curve for the cross test using Peugeot images against the Opel model.

The area under the curve (AUC) of the ROC curve described in Figure 6.13 was calculated using the Python `sklearn.metrics` library, yielding a value of 0.883. These results, along with the fact that the calculated accuracy was unexpectedly above the one returned by the CNN itself, suggested that the model was biased towards images it had seen before. Therefore, all the images that were common between the Opel and Peugeot datasets were discarded and the cross-validation was run anew with the remaining images and some more previously unseen ones. This process and its results are described in Section 6.2.1.

Cross-test 2: Opel images fed to Peugeot model

- Confusion matrix

		Predicted	
		1	**0**
Real	**1**	44	6
	0	62	338

- $ACC = 0.849$
- F_1 score $= 0.564$
- $RMSE = 0.389$
- $PRC = 0.415$
- $MCC = 0.537$
- $REC = 0.880$
- $\kappa = 0.487$

The second cross test took the images originally used to compute the Opel model and fed them to the Peugeot model. Once again, the images were divided into 300 training and 150

testing images, as when computing the model. Each one of the images was re-labeled according to whether or not it represented a Peugeot vehicle, yielding 50 Peugeot images (labeled 1) and 400 that represented other brands (labeled 0). Due to this fact, and similarly to what happened in the previous cross test, the second row of the matrix, which represents the negative images, is more densely populated than the first.

This test had an accuracy of 84.9%, meaning that 15.1% of the images were misclassified as follows:

- Type I errors: the amount of false positives (F_P) was 62, which denounces a lack of precision in the model (41.5%). Therefore, this model is not very good at detecting which images belong to Peugeot vehicles, picking wrong ones more than half the time.
- Type II errors: there were 6 false negatives (F_N) in this test, with the model having a high recall (0.880). This means that the model correctly identifies almost every positive example in the provided set, only misclassifying 12% of the Peugeot images.

The RMSE calculated for this model was 0.389, slightly lower than the one computed for the previous model, which hints at the Peugeot model being slightly more credible than the Opel one. This small advantage is confirmed by the higher F_1 score, at 0.564, and Cohen's kappa score (0.487), which is considered moderate in the Landis and Koch scale ($0.41 \leq \kappa \leq 0.60$), albeit on the lower end of the spectrum. Moreover, the MCC for this test was 0.537, which corresponds to a strong positive relationship between predicted and real values, lending credence to the conclusions stated above.

The procedures for drawing the ROC and precision/recall curves for this test were identical to the ones described above for the first cross test. The resulting optimal threshold is similar, falling somewhere between 0.8 (which yields an accuracy of 0.92 and RMSE of 0.28) and 0.9 (where the F_1 score is 0.63). The ROC curve for this test can be visualised in Figure 6.14. The ROC AUC computed by the `sklearn.metrics` library was 0.900, which would make this classifier very promising. The full results for this analysis can be found in the Appendix, Table A.4.

When taken as a whole, these results hint that a problem similar to that thought to be happening in the first test might be occurring here as well. Therefore, the process of discarding all images known to the model was carried out, and the model was tested again using images it had not been introduced to in previous tests, in order to test its generalisation power.

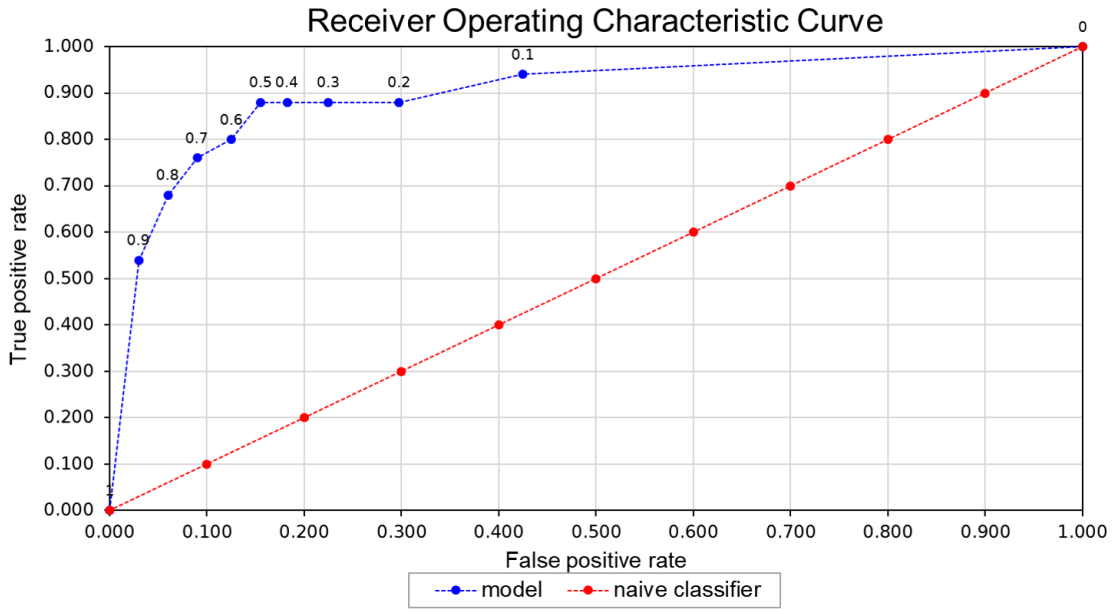


Figure 6.14: ROC curve for the cross test using Opel images against the Peugeot model.

Validation of the Opel and Peugeot models using new images Given the seemingly biased results obtained in the previous section, the Opel and Peugeot models were tested according to new image libraries. These libraries consisted of 151 images of vehicles not belonging to the brand being tested, which differed between the models, as well as 131 new Opel images (fed only to the Opel model) and 126 new Peugeot images (fed only to the Peugeot model). The images were picked so as to provide as little bias as possible when it came to lighting conditions, distance and position of the vehicles relative to the camera.

Validation 1: Opel model

- Confusion matrix

		Predicted	
		1	**0**
Real	**1**	69	62
	0	63	88

- $ACC = 0.557$
- F_1 score $= 0.525$
- $RMSE = 0.666$
- $PRC = 0.523$
- $MCC = 0.109$
- $REC = 0.527$
- $\kappa = 0.109$
- $AUC = 0.587$

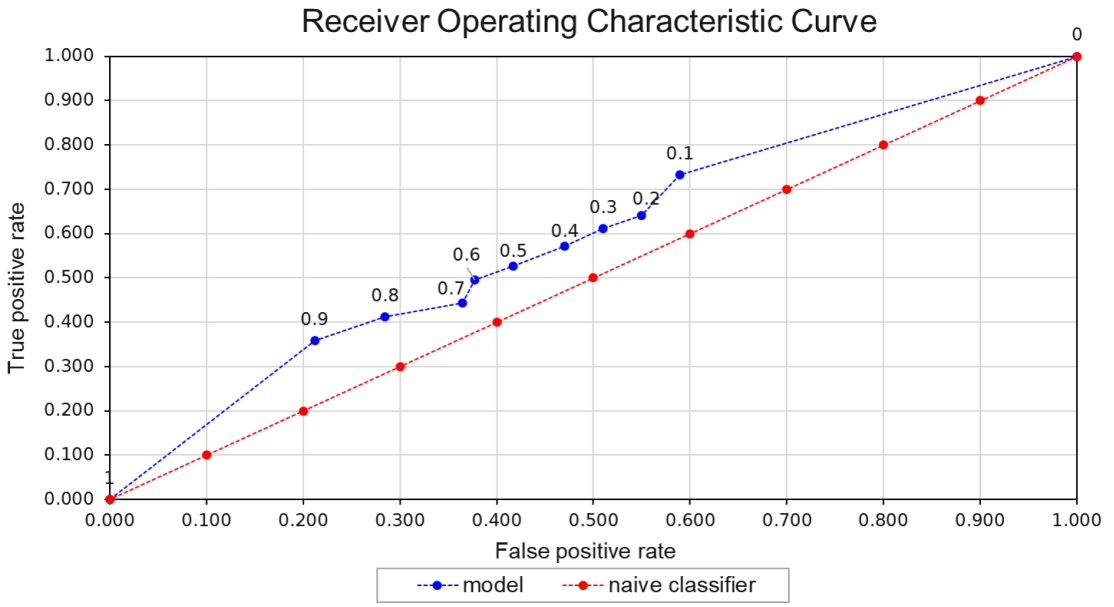


Figure 6.15: ROC curve for the test using previously unseen images against the Opel model.

The new assessment performed on the Opel model seems to confirm the previous assumption that the model was biased towards images it had already seen, with the accuracy dropping to 55.7% and inter-rater agreement now profiling itself as "slight". Further analysis of the area under the plotted curve shows that this model does not seem promising, performing only slimly better than a naïve classifier would.

The misclassification rate was of 44.3%, distributed as follows:

- **Type I errors**: the validation assays produced 63 false positive results, with a precision of 52.3%. This means that the model struggles to identify which images actually belong to Opel vehicles, with its predictions being marginally better than random estimates.
- **Type II errors**: similarly to what happened with the precision metric, the assays yielded 62 false negatives, bringing the recall to 52.7%. The model cannot, therefore, correctly identify positive samples much more efficiently than a naïve one.

Validation 2: Peugeot model

- Confusion matrix:

		Predicted	
		1	**0**
Real	**1**	65	62
	0	56	94

- $ACC = 0.574$
- F_1 score $= 0.524$
- $RMSE = 0.653$
- $PRC = 0.537$
- $MCC = 0.139$
- $REC = 0.512$
- $\kappa = 0.139$
- $AUC = 0.622$

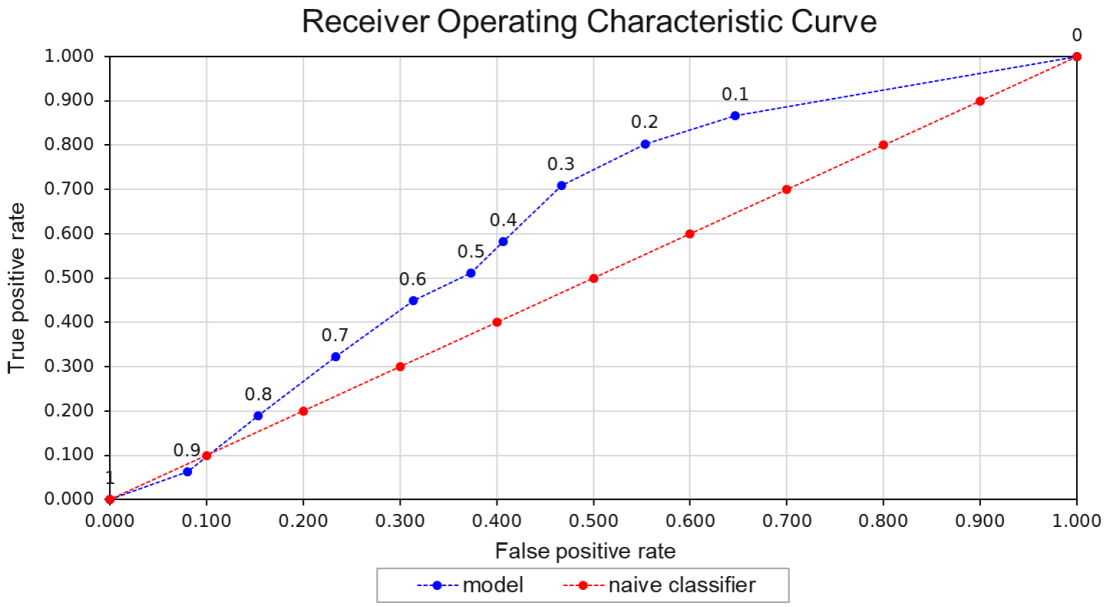


Figure 6.16: ROC curve for the test using previously unseen images against the Peugeot model.

The results for the validation of the Peugeot model are in line with those obtained for the Opel one, with a slightly better accuracy. The kappa statistic is also similar in scope, showing a "slight" level of inter-rater agreement. Moreover, the area under the curve for this assay set (0.622) lends further credence to the poor performance of the model, which can only classify instances 1.244 times better than a random one.

The analysis for type I and II errors is also corroborative of these conclusions.

- **Type I errors**: a total of 56 false positives were identified, with a precision of 53.7% for this model. In practice, this means that 53.7% of the images classified as belonging to Peugeot vehicles actually depict vehicles of the aforementioned brand.
- **Type II errors**: the test results produced 62 false negative results, with the recall being of 51.2%. Therefore, of all Peugeot images in the set, only little over half were correctly classified.

When taken as a whole, the Opel and Peugeot validation assays, which seemed initially promising, were shown to lack the generalisation power needed for the purpose this kind of classifier was meant to achieve. Due to time constraints, no more models were computed, however it can be speculated that better results could have been achieved through a combination of hyperparameter tuning and refinement of the chosen training and test sets.

Chapter 7

Discussion and conclusion

The MMR results presented in Chapter 6 establish the great difficulty inherent to this type of classification. This work saw the employment of three different methodologies: fingerprinting, cascade classification and convolutional neural networks. Given the experimental nature of this work, multiple variations were performed for each method, in order to ascertain the best course of action should any further development arise from these results.

The first technique to be attempted was the use of fingerprinting. These fingerprinting assays were performed with recourse to two different types of matchers, brute-force and FLANN, with only the former yielding any results. Moreover, two match-discarding techniques were attempted, Lowe's ratio test (at the ratios of 0.4 and 0.7) and OpenCV's cross-check. Additionally, three different ordering functions were used in conjunction with the aforementioned ratio test, namely, by maximum value (MV), minimum absolute difference between ratios (AD) and maximum match-to-keypoint ratio (MR).

The results for the fingerprinting assays showed the best combination of parameters to be the use of a brute-force matcher, ratio-testing (r=0.4) and the maximum ratio ordering function, for both brand and model. However, the overall accuracy for brand recognition was of 16.8%, and model results did not surpass 11.2% accuracy. Cohen's kappa correlation metric was likewise weak in both cases (12.5% agreement for the brand evaluation and 10.2% for the model one).

As a whole, these values cannot be considered satisfactory, and could benefit from a great deal of fine-tuning and refinement of the process as a whole. The main problem with this approach was the inadequacy of the samples relatively to what was meant to be achieved. Fingerprinting is a technique generally used to detect almost exact matches between images, but this work encompasses a much broader definition of what "matching" entails - for instance, different vehicles would have to be considered matches if they belonged to the same brand and model, independently of camera pose, visibility conditions or image resolution. The use of the Stanford dataset for this kind of assay was, therefore, ill-advised to begin with, hindering both make and model recognition. Another side-effect of the poor choice of dataset was the failure of tried and tested methods such as Lowe's ratio test. Theoretically, a ratio of 0.7 would produce interesting results, lowering the chance of false positives, but in practice these were so abundant that a lower ratio (0.4) proved better for these assays. This means that a lot of correctly identified positive matches were disregarded, lowering the overall accuracy.

The next step undertaken aimed to minimise the impact of this dataset choice by focusing only on the brand, through grille recognition. In theory, the image would be processed in order for the grille to be extracted, and this grille crop would then be subject to fingerprinting assays in order to determine the brand. Once the brand was known, a similar technique would then be applied to determine the model, using more vehicle sections such as mirrors or lights to help the recognition efforts.

For this purpose, two different families of cascade classifiers were designed, one based on Haar features and the other on LBP ones. The grille ROI was then determined from the output using two different metrics, the first being the intersection of potential ROIs and the other the min-neighbours (neighbours for short) built-in parameter in OpenCV. Two different sizes for the sets of negative samples were also tested (designed as N_{small} and N_{big}).

Considering the conjugation of all these factors, the combination that yielded the best results was that of a Haar cascade classifier using the neighbours technique and N_{small} as the set of negative samples while calculating the classifier. This approach resulted in an accuracy of 0.495, with a precision of 0.391 and a recall of 0.994. With false positives amounting to 50.3% of the results, this was also the assay that detected fewer extraneous objects instead of the desired grille. Although the near perfect recall indicates the model identified almost all the images that possessed a grille, the relatively low precision metric hints at the model's poor discriminating power, misclassifying too many negative samples as positive ones.

The remaining results showed a general tendency for the N_{small} cascades to perform better than the N_{big} ones, and for Haar cascades to perform better than LBP ones. This last result was expectable, given the nature of the features used in each case - LBP features are meant to result in lighter and faster cascade calculations when compared to the more robust Haar ones, but this tends to compromise their accuracy as a trade-off. However, given that the available literature features examples that use a number of negative samples substantially higher than the positive set, the first results proved more surprising. One possible explanation for this discrepancy was the difference in the dataset used for calculating the cascade, as opposed to the Stanford dataset, which was used as a test-set only. The N_{small} samples were the first to be chosen, being mainly concerned with setting car booths as negative samples, and throwing in some mirrors, wheels and front panes for further clarification, as well as off-road elements such as people. When the dataset was expanded, there was an attempt to balance all the negative samples in equal proportions, which may have contributed to the corresponding cascades' poorer performance by introducing elements the Stanford dataset did not focus as much on.

A consequence of these poor grille detection results was the abandonment of the integrated cascade and fingerprinting approach, which would undoubtedly result in very low accuracies. In fact, the choice of dataset proved to be a critical issue when dealing with make and model recognition, with the Stanford dataset possessing too many degrees of freedom and the subjectivity of image choice for the cascade recognition assays having had a significant impact in the results. The next step was, therefore, the adoption of a dataset based on images captured in tolling events, so as to better represent the reality of the intended target business.

A new approach was subsequently attempted through the use of convolutional neural networks (CNNs), applied to this new, more restricted dataset. Instead of focusing on the detection of every brand, this new approach also focused on producing binary models which were specific to each manufacturer, forgoing model recognition entirely at the starting point. To serve as proof of this concept, the two most prevalent brands in the new dataset, Opel and Peugeot, were chosen.

Each model was firstly calculated and evaluated against a test set that contained some images that overlapped with the training set, which proved to generate bias in the first results, with the Opel-specific model presenting an accuracy of 0.889, precision of 0.879 and recall of 0.870, while the Peugeot-specific model displayed an accuracy of 0.953, precision of 0.969 and recall of 0.925. In order to test the generalisation power of these models, a cross-test was performed, in which the images used in the calculation of the Peugeot model were fed to the Opel model for classification and vice-versa. The aforementioned bias was supported by these tests, since the accuracy dropped from 0.889 to 0.793 for the Opel model and from 0.953 to 0.849 for the Peugeot one. More tellingly, although recall stabilised at 0.880 for both models, their precisions dropped significantly, with the Opel model exhibiting a precision of 0.336 for this test, while the Peugeot model fared

slightly better, at 0.415. The models were, therefore, shown to misclassify many images not belonging to each brand as positive samples. Moreover, in order to completely eliminate any possible overlap between training and test sets, new images, chosen from the same dataset, were presented to each model. This time, the Opel model presented an accuracy of 0.557, precision of 0.523 and recall of 0.527, while the Peugeot model exhibited an accuracy of 0.574, precision of 0.537 and recall of 0.512. This shows that these models could do with further refinement, with these new images being potential candidates for network readjustment assays.

Overall, these results cannot be fully compared, since they not only hail from different datasets, but also have different metrics associated with them and different purposes. Some general conclusions can, nevertheless, be made for this work as a whole.

An important conclusion to be drawn is that, independently of the method used, the choice of dataset is crucial in determining whether an assay stands a chance of yielding satisfactory results. This issue was evidenced by the results obtained for the fingerprinting assays, which had their accuracy peak at 16.8% and produced no results when using a comparatively more refined matcher such as FLANN. Even though the use of this technique was most probably inadequate for MMR purposes, these efforts were likely also hindered by the extreme diversity present in the Stanford cars dataset, which comprises a wide range of angles when compared to those present in a standard open road tolling image capture, in which the cameras are set in a stationary position on the tolling gantry. Likewise, the grille detection results obtained through the use of cascade classifiers were probably subject to the same problem. A case could be made for applying these methods using a different dataset, such as the one later used when computing the CNN classifiers, which consisted of actual tolling images which were, therefore, subject to fewer degrees of freedom than those present in the Stanford dataset. Nevertheless, the existing literature suggests that pattern recognition methods such as the ones used in these convolutional neural networks would prove more reliable in make and model recognition efforts even if the exact same knowledge base had been used.

Although there were no expectations for relevant and definitive results, the work performed involved many different techniques in pursuit of its goal, which was to provide a proof of concept for conceiving and integrating a vehicle make and model recognition system into an already fully functioning application, enhancing its functionality. For that purpose, a putative architecture was established, but not implemented, and requisites were defined for this possible future implementation. Given the results described above, the best course for future development would be the employment of CNNs, with greater emphasis on hyperparameter tuning to achieve more consistent results. A possible implementation could reside on the use of different sets of neural networks, an initial one focusing on grille recognition in order to extract the brand, which would then forward the image to a neural network trained to discern between models of the most probable brand. Alternatively, in order to encompass the possibility of the initial brand classification being wrong, the image could also be forwarded to the set of n most probable brands, with the results then being merged and rearranged according to their probabilities. This type of solution would probably be costly, mainly in the development phase, given the amount of fine-tuning required and the extensive knowledge base that would have to be amassed. Therefore, a distributed system, with load balancing and inherent redundancy, would be required in order for a large-scale solution like this to be considered.

Appendix A

Supplementary information

A.1 Stanford dataset

A.1.1 Dataset analysis

Table A.1: Detailed analysis for the Stanford dataset.

Brand	Brand name	Model	Model name	Images				Keypoints			
				Train	Test	Total	Brand total	Train	Test	Total	Brand avg
1	AM General Hummer	1	AM General Hummer SUV 2000	45	44	89	89	22078	11769	33847	33847.0
2	Acura	2	Acura RL Sedan 2012	32	32	64	482	11633	11555	23188	34187.7
		3	Acura TL Sedan 2012	43	43	86		24482	25146	49628	
		4	Acura TL Type-S 2008	42	42	84		15044	15305	30349	
		5	Acura TSX Sedan 2012	41	40	81		11831	20853	32684	
		6	Acura Integra Type R 2001	45	44	89		20067	17798	37865	
		7	Acura ZDX Hatchback 2012	39	39	78		15448	15964	31412	
3	Aston Martin	8	Aston Martin V8 Vantage Convertible 2012	45	45	90	314	16720	16062	32782	34922.8
		9	Aston Martin V8 Vantage Coupe 2012	41	41	82		13326	16046	29372	
		10	Aston Martin Virage Convertible 2012	33	33	66		16888	26966	43854	
		11	Aston Martin Virage Coupe 2012	38	38	76		15239	18444	33683	
4	Audi	12	Audi RS 4 Convertible 2008	37	36	73	1169	24275	15711	39986	35413.7
		13	Audi A5 Coupe 2012	41	41	82		17567	14761	32328	
		14	Audi TTS Coupe 2012	43	42	85		25766	12485	38251	
		15	Audi R8 Coupe 2012	43	43	86		14765	16092	30857	
		16	Audi V8 Sedan 1994	44	43	87		9246	10679	19925	
		17	Audi 100 Sedan 1994	41	40	81		10169	16019	26188	
		18	Audi 100 Wagon 1994	43	42	85		8730	17124	25854	
		19	Audi TT Hatchback 2011	41	40	81		14346	10747	25093	
		20	Audi S6 Sedan 2011	46	46	92		31665	24949	56614	
		21	Audi S5 Convertible 2012	42	42	84		18558	12006	30564	
		22	Audi S5 Coupe 2012	43	42	85		14049	22830	36879	
		23	Audi S4 Sedan 2012	40	39	79		28720	15934	44654	
		24	Audi S4 Sedan 2007	45	45	90		24999	24871	49870	
		25	Audi TT RS Coupe 2012	40	39	79		20055	18674	38729	
5	BMW	26	BMW ActiveHybrid 5 Sedan 2012	34	34	68	1055	13829	18720	32549	34929.6
		27	BMW 1 Series Convertible 2012	36	35	71		15793	17049	32842	
		28	BMW 1 Series Coupe 2012	41	41	82		19973	16904	36877	
		29	BMW 3 Series Sedan 2012	43	42	85		20188	15857	36045	
		30	BMW 3 Series Wagon 2012	42	41	83		31716	12178	43894	
		31	BMW 6 Series Convertible 2007	44	44	88		12130	13381	25511	
		32	BMW X5 SUV 2007	42	41	83		18684	16900	35584	
		33	BMW X6 SUV 2012	42	42	84		14230	20456	34686	
		34	BMW M3 Coupe 2012	45	44	89		19697	16697	36394	
		35	BMW M5 Sedan 2010	41	41	82		19641	14303	33944	
		36	BMW M6 Convertible 2010	41	41	82		32861	11803	44664	
		37	BMW X3 SUV 2012	39	38	77		21845	10123	31968	
		38	BMW Z4 Convertible 2012	41	40	81		12745	16382	29127	
6	Bentley	39	Bentley Continental Supersports Conv. Convertible 2012	37	36	73	472	8261	11498	19759	41494.5
		40	Bentley Arnage Sedan 2009	39	39	78		57125	20958	78083	
		41	Bentley Mulsanne Sedan 2011	36	35	71		16596	17853	34449	
		42	Bentley Continental GT Coupe 2012	35	34	69		14112	14610	28722	
		43	Bentley Continental GT Coupe 2007	46	46	92		22522	21247	43769	
		44	Bentley Continental Flying Spur Sedan 2007	45	44	89		20170	24015	44185	
7	Bugatti	45	Bugatti Veyron 16.4 Convertible 2009	33	32	65	152	14318	12235	26553	32212.0
		46	Bugatti Veyron 16.4 Coupe 2009	44	43	87		16598	21273	37871	
8	Buick	47	Buick Regal GS 2012	35	35	70	314	15124	18033	33157	34279.3
		48	Buick Rainier SUV 2007	43	42	85		13518	19151	32669	
		49	Buick Verano Sedan 2012	38	37	75		13352	12398	25750	
		50	Buick Enclave SUV 2012	42	42	84		25970	19571	45541	
9	Cadillac	51	Cadillac CTS-V Sedan 2012	43	43	86	257	9773	13683	23456	28447.7
		52	Cadillac SRX SUV 2012	41	41	82		16313	14613	30926	
		53	Cadillac Escalade EXT Crew Cab 2007	45	44	89		16419	14542	30961	
10	Chevrolet	54	Chevrolet Silverado 1500 Hybrid Crew Cab 2012	40	40	80	1799	9403	25371	34774	40874.7
		55	Chevrolet Corvette Convertible 2012	40	39	79		21227	16034	37261	
		56	Chevrolet Corvette ZR1 2012	47	46	93		21036	25612	46648	
		57	Chevrolet Corvette Ron Fellows Edition Z06 2007	38	37	75		33242	18014	51256	
		58	Chevrolet Traverse SUV 2012	44	44	88		77296	20249	97545	
		59	Chevrolet Camaro Convertible 2012	45	44	89		15336	20429	35765	
		60	Chevrolet HHR SS 2010	37	36	73		10903	11353	22256	
		61	Chevrolet Impala Sedan 2007	43	43	86		31681	20043	51724	
		62	Chevrolet Tahoe Hybrid SUV 2012	37	37	74		11465	12491	23956	
		63	Chevrolet Sonic Sedan 2012	44	44	88		54366	18031	72397	
		64	Chevrolet Express Cargo Van 2007	30	29	59		9990	12952	22942	
		65	Chevrolet Avalanche Crew Cab 2012	45	45	90		11321	13264	24585	

		66	Chevrolet Cobalt SS 2010	42	41	83		31334	16091	47425	
		67	Chevrolet Malibu Hybrid Sedan 2010	39	38	77		12954	25083	38037	
		68	Chevrolet TrailBlazer SS 2009	40	40	80		15737	24752	40489	
		69	Chevrolet Silverado 2500HD Regular Cab 2012	38	38	76		10131	11364	21495	
		70	Chevrolet Silverado 1500 Classic Extended Cab 2007	43	42	85		20274	13562	33836	
		71	Chevrolet Express Van 2007	35	35	70		9042	11499	20541	
		72	Chevrolet Monte Carlo Coupe 2007	45	45	90		16310	19649	35959	
		73	Chevrolet Malibu Sedan 2007	45	44	89		11229	22779	34008	
		74	Chevrolet Silverado 1500 Extended Cab 2012	44	43	87		14308	19104	33412	
		75	Chevrolet Silverado 1500 Regular Cab 2012	44	44	88		35573	37359	72932	
11	Chrysler	76	Chrysler Aspen SUV 2009	44	43	87	516	14240	13948	28188	40497.8
		77	Chrysler Sebring Convertible 2010	41	40	81		19225	12030	31255	
		78	Chrysler Town and Country Minivan 2012	38	37	75		18978	22220	41198	
		79	Chrysler 300 SRT-8 2010	49	48	97		48990	16038	65028	
		80	Chrysler Crossfire Convertible 2008	43	43	86		9386	18610	27996	
		81	Chrysler PT Cruiser Convertible 2008	45	45	90		29240	20082	49322	
12	Daewoo	82	Daewoo Nubira Wagon 2002	45	45	90	90	10477	11249	21726	21726.0
13	Dodge	83	Dodge Caliber Wagon 2012	41	40	81	1253	13023	22381	35404	38606.7
		84	Dodge Caliber Wagon 2007	42	42	84		10688	19697	30385	
		85	Dodge Caravan Minivan 1997	44	43	87		17408	17264	34672	
		86	Dodge Ram Pickup 3500 Crew Cab 2010	43	42	85		23164	18278	41442	
		87	Dodge Ram Pickup 3500 Quad Cab 2009	44	44	88		23490	16061	39551	
		88	Dodge Sprinter Cargo Van 2009	40	39	79		4377	34822	39199	
		89	Dodge Journey SUV 2012	44	44	88		36128	17844	53972	
		90	Dodge Dakota Crew Cab 2010	41	41	82		16250	15761	32011	
		91	Dodge Dakota Club Cab 2007	39	38	77		14839	12100	26939	
		92	Dodge Magnum Wagon 2008	40	40	80		8381	22019	30400	
		93	Dodge Challenger SRT8 2011	39	39	78		19571	13001	32572	
		94	Dodge Durango SUV 2012	44	43	87		31637	25888	57525	
		95	Dodge Durango SUV 2007	46	45	91		23260	15461	38721	
		96	Dodge Charger Sedan 2012	41	41	82		32545	15999	48544	
		97	Dodge Charger SRT-8 2009	42	42	84		18083	19680	37763	
14	Eagle	98	Eagle Talon Hatchback 1998	46	46	92	92	31698	16773	48471	48471.0
15	Fiat	99	FIAT 500 Abarth 2012	28	27	55	122	22880	10833	33713	30883.0
		100	FIAT 500 Convertible 2012	34	33	67		13818	14235	28053	
16	Ferrari	101	Ferrari FF Coupe 2012	42	42	84	326	17732	12163	29895	27530.5
		102	Ferrari California Convertible 2012	39	39	78		12231	18472	30703	
		103	Ferrari 458 Italia Convertible 2012	40	39	79		14230	14822	29052	
		104	Ferrari 458 Italia Coupe 2012	43	42	85		9900	10572	20472	
17	Fisker	105	Fisker Karma Sedan 2012	44	43	87	87	16046	24838	40884	40884.0
18	Ford	106	Ford F-450 Super Duty Crew Cab 2012	42	41	83	1035	19675	14909	34584	37891.3
		107	Ford Mustang Convertible 2007	45	44	89		16158	40068	56226	
		108	Ford Freestar Minivan 2007	44	44	88		9932	16534	26466	
		109	Ford Expedition EL SUV 2009	45	44	89		11073	13125	24198	
		110	Ford Edge SUV 2012	43	43	86		29044	16418	45462	
		111	Ford Ranger SuperCab 2011	42	42	84		10708	15681	26389	
		112	Ford GT Coupe 2006	46	45	91		49260	23227	72487	
		113	Ford F-150 Regular Cab 2012	43	42	85		16458	11425	27883	
		114	Ford F-150 Regular Cab 2007	45	45	90		17278	14575	31853	
		115	Ford Focus Sedan 2007	45	45	90		14752	34515	49267	
		116	Ford E-Series Wagon Van 2012	38	37	75		10183	20399	30582	
		117	Ford Fiesta Sedan 2012	43	42	85		15395	13904	29299	
19	GMC	118	GMC Terrain SUV 2012	42	41	83	473	13557	20032	33589	40088.8
		119	GMC Savana Van 2012	68	68	136		22072	31554	53626	
		120	GMC Yukon Hybrid SUV 2012	43	42	85		21916	19090	41006	
		121	GMC Acadia SUV 2012	45	44	89		13018	27634	40652	
		122	GMC Canyon Extended Cab 2012	40	40	80		15663	15908	31571	
20	Geo	123	Geo Metro Convertible 1993	45	44	89	89	9409	18664	28073	28073.0
21	Hummer	124	HUMMER H3T Crew Cab 2010	39	39	78	165	16632	12330	28962	38520.5
		125	HUMMER H2 SUT Crew Cab 2009	44	43	87		29959	18120	48079	
22	Honda	126	Honda Odyssey Minivan 2012	42	42	84	321	13781	15262	29043	27056.8
		127	Honda Odyssey Minivan 2007	41	41	82		17484	15049	32533	
		128	Honda Accord Coupe 2012	39	39	78		7516	16978	24494	
		129	Honda Accord Sedan 2012	39	38	77		8825	13332	22157	
23	Hyundai	130	Hyundai Veloster Hatchback 2012	41	41	82	871	21644	14675	36319	40263.7
		131	Hyundai Santa Fe SUV 2012	42	42	84		15458	23352	38810	
		132	Hyundai Tucson SUV 2012	44	43	87		29174	18914	48088	
		133	Hyundai Veracruz SUV 2012	42	42	84		28868	24137	53005	
		134	Hyundai Sonata Hybrid Sedan 2012	34	33	67		10620	16925	27545	
		135	Hyundai Elantra Sedan 2007	42	42	84		14184	18134	32318	
		136	Hyundai Accent Sedan 2012	24	24	48		3448	28390	31838	

		137	Hyundai Genesis Sedan 2012	44	43	87		21746	16006	37752	
		138	Hyundai Sonata Sedan 2012	40	39	79		18104	17113	35217	
		139	Hyundai Elantra Touring Hatchback 2012	43	42	85		17766	16754	34520	
		140	Hyundai Azera Sedan 2012	42	42	84		50609	16880	67489	
24	Infiniti	141	Infiniti G Coupe IPL 2012	34	34	68	133	10036	31503	41539	35817.0
		142	Infiniti QX56 SUV 2011	33	32	65		11102	18993	30095	
25	Isuzu	143	Isuzu Ascender SUV 2008	40	40	80	80	9013	21789	30802	30802.0
26	Jaguar	144	Jaguar XK XKR 2012	47	46	93	93	31566	25347	56913	56913.0
27	Jeep	145	Jeep Patriot SUV 2012	44	44	88	438	63057	21892	84949	69935.6
		146	Jeep Wrangler SUV 2012	43	43	86		37065	15261	52326	
		147	Jeep Liberty SUV 2012	45	44	89		54759	17612	72371	
		148	Jeep Grand Cherokee SUV 2012	45	45	90		18707	22721	41428	
		149	Jeep Compass SUV 2012	43	42	85		72441	26163	98604	
28	Lamborghini	150	Lamborghini Reventon Coupe 2008	36	36	72	319	10929	18238	29167	28572.8
		151	Lamborghini Aventador Coupe 2012	44	43	87		10855	20042	30897	
		152	Lamborghini Gallardo LP 570-4 Superleggera 2012	36	35	71		10866	10389	21255	
		153	Lamborghini Diablo Coupe 2001	45	44	89		22246	10726	32972	
29	Land Rover	154	Land Rover Range Rover SUV 2012	43	42	85	170	40560	20077	60637	49053.0
		155	Land Rover LR2 SUV 2012	43	42	85		16321	21148	37469	
30	Lincoln	156	Lincoln Town Car Sedan 2011	39	39	78	78	8405	18095	26500	26500.0
31	Mini	157	MINI Cooper Roadster Convertible 2012	37	36	73	73	13887	12524	26411	26411.0
32	Maybach	158	Maybach Landaulet Convertible 2012	29	29	58	58	8644	9022	17666	17666.0
33	Mazda	159	Mazda Tribute SUV 2011	36	36	72	72	13996	14136	28132	28132.0
34	McLaren	160	McLaren MP4-12C Coupe 2012	44	44	88	88	20215	18823	39038	39038.0
35	Mercedes-Benz	161	Mercedes-Benz 300-Class Convertible 1993	48	48	96	518	14492	21701	36193	44095.0
		162	Mercedes-Benz C-Class Sedan 2012	46	45	91		20676	26431	47107	
		163	Mercedes-Benz SL-Class Coupe 2009	37	36	73		18091	12359	30450	
		164	Mercedes-Benz E-Class Sedan 2012	44	43	87		62853	13407	76260	
		165	Mercedes-Benz S-Class Sedan 2012	45	44	89		20824	16114	36938	
		166	Mercedes-Benz Sprinter Van 2012	41	41	82		20903	16719	37622	
36	Mitsubishi	167	Mitsubishi Lancer Sedan 2012	48	47	95	95	62516	21276	83792	83792.0
37	Nissan	168	Nissan Leaf Hatchback 2012	42	42	84	341	26216	18108	44324	53935.8
		169	Nissan NV Passenger Van 2012	39	38	77		24625	13621	38246	
		170	Nissan Juke Hatchback 2012	44	44	88		72664	24752	97416	
		171	Nissan 240SX Coupe 1998	46	46	92		16567	19190	35757	
38	Plymouth	172	Plymouth Neon Coupe 1999	44	44	88	88	29991	27061	57052	57052.0
39	Porsche	173	Porsche Panamera Sedan 2012	44	43	87	87	28082	19584	47666	47666.0
40	Ram	174	Ram C/V Cargo Van Minivan 2012	41	41	82	82	7026	14783	21809	21809.0
41	Rolls-Royce	175	Rolls-Royce Phantom Drophead Coupe Convertible 2012	31	30	61	226	6573	7133	13706	30377.0
		176	Rolls-Royce Ghost Sedan 2012	39	38	77		34755	13164	47919	
		177	Rolls-Royce Phantom Sedan 2012	44	44	88		9884	19622	29506	
42	Scion	178	Scion xD Hatchback 2012	42	41	83	83	18152	10392	28544	28544.0
43	Spyker	179	Spyker C8 Convertible 2009	45	45	90	175	28569	27177	55746	47661.0
		180	Spyker C8 Coupe 2009	43	42	85		28271	11305	39576	
44	Suzuki	181	Suzuki Aerio Sedan 2007	38	38	76	333	12290	10616	22906	49613.3
		182	Suzuki Kizashi Sedan 2012	46	46	92		38835	43195	82030	
		183	Suzuki SX4 Hatchback 2012	42	42	84		43280	14228	57508	
		184	Suzuki SX4 Sedan 2012	41	40	81		20157	15852	36009	
45	Tesla	185	Tesla Model S Sedan 2012	39	38	77	77	14767	17591	32358	32358.0
46	Toyota	186	Toyota Sequoia SUV 2012	39	38	77	332	13998	21236	35234	37978.5
		187	Toyota Camry Sedan 2012	44	43	87		26250	20875	47125	
		188	Toyota Corolla Sedan 2012	44	43	87		13499	23635	37134	
		189	Toyota 4Runner SUV 2012	41	40	81		19120	13301	32421	
47	Volkswagen	190	Volkswagen Golf Hatchback 2012	43	43	86	263	6839	18148	24987	46617.0
		191	Volkswagen Golf Hatchback 1991	46	46	92		47582	20160	67742	
		192	Volkswagen Beetle Hatchback 2012	43	42	85		27055	20067	47122	
48	Volvo	193	Volvo C30 Hatchback 2012	42	41	83	260	21854	20945	42799	51237.3
		194	Volvo 240 Sedan 1993	46	45	91		34864	41096	75960	
		195	Volvo XC90 SUV 2007	43	43	86		19377	15576	34953	
49	Smart	196	smart fortwo Convertible 2012	40	40	80	80	19147	21782	40929	40929.0

A.1.2 Fingerprinting

Model sample

Table A.2: Stanford dataset vehicle models used for fingerprinting.

Chevrolet

54 Chevrolet Silverado 1500 Hybrid Crew Cab 2012
55 Chevrolet Corvette Convertible 2012
56 Chevrolet Corvette ZR1 2012
57 Chevrolet Corvette Ron Fellows Edition Z06 2007
58 Chevrolet Traverse SUV 2012
59 Chevrolet Camaro Convertible 2012
60 Chevrolet HHR SS 2010
61 Chevrolet Impala Sedan 2007
62 Chevrolet Tahoe Hybrid SUV 2012
63 Chevrolet Sonic Sedan 2012
64 Chevrolet Express Cargo Van 2007
65 Chevrolet Avalanche Crew Cab 2012
66 Chevrolet Cobalt SS 2010
67 Chevrolet Malibu Hybrid Sedan 2010
68 Chevrolet TrailBlazer SS 2009
69 Chevrolet Silverado 2500HD Regular Cab 2012
70 Chevrolet Silverado 1500 Classic Extended Cab 2007
71 Chevrolet Express Van 2007
72 Chevrolet Monte Carlo Coupe 2007
73 Chevrolet Malibu Sedan 2007
74 Chevrolet Silverado 1500 Extended Cab 2012
75 Chevrolet Silverado 1500 Regular Cab 2012

Dodge

83 Dodge Caliber Wagon 2012
84 Dodge Caliber Wagon 2007
85 Dodge Caravan Minivan 1997
86 Dodge Ram Pickup 3500 Crew Cab 2010
87 Dodge Ram Pickup 3500 Quad Cab 2009
88 Dodge Sprinter Cargo Van 2009
89 Dodge Journey SUV 2012
90 Dodge Dakota Crew Cab 2010
91 Dodge Dakota Club Cab 2007
92 Dodge Magnum Wagon 2008
93 Dodge Challenger SRT8 2011
94 Dodge Durango SUV 2012
95 Dodge Durango SUV 2007
96 Dodge Charger Sedan 2012
97 Dodge Charger SRT-8 2009

Ford

106 Ford F-450 Super Duty Crew Cab 2012
107 Ford Mustang Convertible 2007
108 Ford Freestar Minivan 2007
109 Ford Expedition EL SUV 2009
110 Ford Edge SUV 2012
111 Ford Ranger SuperCab 2011
112 Ford GT Coupe 2006
113 Ford F-150 Regular Cab 2012
114 Ford F-150 Regular Cab 2007
115 Ford Focus Sedan 2007
116 Ford E-Series Wagon Van 2012
117 Ford Fiesta Sedan 2012

GMC

118 GMC Terrain SUV 2012
119 GMC Savana Van 2012
120 GMC Yukon Hybrid SUV 2012
121 GMC Acadia SUV 2012
122 GMC Canyon Extended Cab 2012

Honda

126 Honda Odyssey Minivan 2012
127 Honda Odyssey Minivan 2007
128 Honda Accord Coupe 2012
129 Honda Accord Sedan 2012

Hyundai

130 Hyundai Veloster Hatchback 2012
131 Hyundai Santa Fe SUV 2012
132 Hyundai Tucson SUV 2012
133 Hyundai Veracruz SUV 2012
134 Hyundai Sonata Hybrid Sedan 2012
135 Hyundai Elantra Sedan 2007
136 Hyundai Accent Sedan 2012
137 Hyundai Genesis Sedan 2012
138 Hyundai Sonata Sedan 2012
139 Hyundai Elantra Touring Hatchback 2012
140 Hyundai Azera Sedan 2012

Jeep

145 Jeep Patriot SUV 2012
146 Jeep Wrangler SUV 2012
147 Jeep Liberty SUV 2012
148 Jeep Grand Cherokee SUV 2012
149 Jeep Compass SUV 2012

Land Rover

154 Land Rover Range Rover SUV 2012
155 Land Rover LR2 SUV 2012

Mercedes-Benz

161 Mercedes-Benz 300-Class Convertible 1993
162 Mercedes-Benz C-Class Sedan 2012
163 Mercedes-Benz SL-Class Coupe 2009
164 Mercedes-Benz E-Class Sedan 2012
165 Mercedes-Benz S-Class Sedan 2012
166 Mercedes-Benz Sprinter Van 2012

Nissan

168 Nissan Leaf Hatchback 2012
169 Nissan NV Passenger Van 2012
170 Nissan Juke Hatchback 2012
171 Nissan 240SX Coupe 1998

Ram

174 Ram C/V Cargo Van Minivan 2012

Toyota

186 Toyota Sequoia SUV 2012 186
187 Toyota Camry Sedan 2012 187
188 Toyota Corolla Sedan 2012 188
189 Toyota 4Runner SUV 2012 189

A.2 Proprietary dataset

Table A.3: Evaluation of the Opel model with the images used to build the Peugeot model.

	Prediction at different thresholds										
	0	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1**
TP	50	46	45	45	44	44	44	43	43	41	0
TN	0	265	279	290	300	313	323	328	346	358	400
FP	400	135	121	110	100	87	77	72	54	42	0
FN	0	4	5	5	6	6	6	7	7	9	50
Accuracy	0.111	0.691	0.720	0.744	0.764	0.793	0.816	0.824	0.864	0.887	0.889
Precision	0.111	0.254	0.271	0.290	0.306	0.336	0.364	0.374	0.443	0.494	0.000
Recall (TP rate)	1.000	0.920	0.900	0.900	0.880	0.880	0.880	0.860	0.860	0.820	0.000
Fallout (FP rate)	1.000	0.338	0.303	0.275	0.250	0.218	0.193	0.180	0.135	0.105	0.000
F_1 **score**	0.200	0.398	0.417	0.439	0.454	0.486	0.515	0.521	0.585	0.617	0.000
MCC	0.000	0.373	0.389	0.413	0.424	0.458	0.487	0.490	0.554	0.539	0.000
Cohen's kappa	0.000	0.271	0.297	0.326	0.346	0.388	0.424	0.433	0.514	0.555	0.000
RMSE	0.943	0.556	0.529	0.506	0.485	0.455	0.429	0.419	0.368	0.337	0.333
ROC AUC	0.883										
Precision/ recall AUC ratio	2.290										

Table A.4: Evaluation of the Peugeot model with the images used to build the Opel model.

	Prediction at different thresholds										
	0	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1**
TP	50	47	44	44	44	44	40	38	34	27	0
TN	0	230	281	310	327	338	350	364	376	388	400
FP	400	170	119	90	73	62	50	36	24	12	0
FN	0	3	6	6	6	6	10	12	16	23	50
Accuracy	0.111	0.616	0.722	0.787	0.824	0.849	0.867	0.893	0.911	0.922	0.889
Precision	0.111	0.217	0.270	0.328	0.376	0.415	0.444	0.514	0.586	0.692	0.000
Recall (TP rate)	1.000	0.940	0.880	0.880	0.880	0.880	0.800	0.760	0.680	0.540	0.000
Fallout (FP rate)	1.000	0.425	0.298	0.225	0.183	0.155	0.125	0.090	0.060	0.030	0.000
F_1 **score**	0.200	0.352	0.413	0.478	0.527	0.564	0.571	0.613	0.630	0.607	0.000
MCC	0.000	0.324	0.381	0.450	0.500	0.537	0.530	0.568	0.581	0.570	0.000
Cohen's kappa	0.000	0.209	0.293	0.378	0.440	0.487	0.500	0.554	0.579	0.564	0.000
RMSE	0.943	0.620	0.527	0.462	0.419	0.389	0.365	0.327	0.298	0.279	0.333
ROC AUC	0.900										
Precision/ recall AUC ratio	3.585										

Table A.5: Evaluation of the Opel model with previously unused images.

	Prediction at different thresholds										
	0	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1**
TP	131	96	84	80	75	69	65	58	54	47	0
TN	0	62	68	74	80	88	94	96	108	119	151
FP	151	89	83	77	71	63	57	55	43	32	0
FN	0	35	47	51	56	62	66	73	77	84	131
Accuracy	0.465	0.560	0.539	0.546	0.550	0.557	0.564	0.546	0.574	0.589	0.535
Precision	0.465	0.519	0.503	0.510	0.514	0.523	0.533	0.513	0.557	0.595	0.000
Recall (TP rate)	1.000	0.733	0.641	0.611	0.573	0.527	0.496	0.443	0.412	0.359	0.000
Fallout (FP rate)	1.000	0.589	0.550	0.510	0.470	0.417	0.377	0.364	0.285	0.212	0.000
F_1 **score**	0.634	0.608	0.564	0.556	0.542	0.525	0.514	0.475	0.474	0.448	0.000
MCC	0.000	0.151	0.093	0.101	0.102	0.109	0.119	0.080	0.134	0.163	0.000
Cohen's kappa	0.000	0.140	0.090	0.099	0.102	0.109	0.119	0.079	0.130	0.151	0.000
RMSE	0.732	0.663	0.679	0.674	0.671	0.666	0.660	0.674	0.652	0.641	0.682
ROC AUC	0.587										
Precision/ recall AUC ratio	0.936										

Table A.6: Evaluation of the Peugeot model with previously unused images.

	Prediction at different thresholds										
	0	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1**
TP	127	110	102	90	74	65	57	41	24	8	0
TN	0	53	67	80	89	94	103	115	127	138	150
FP	150	97	83	70	61	56	47	35	23	12	0
FN	0	17	25	37	53	62	70	86	103	119	127
Accuracy	0.458	0.588	0.610	0.614	0.588	0.574	0.578	0.563	0.545	0.527	0.542
Precision	0.458	0.531	0.551	0.563	0.548	0.537	0.548	0.539	0.511	0.400	0.000
Recall (TP rate)	1.000	0.866	0.803	0.709	0.583	0.512	0.449	0.323	0.189	0.063	0.000
Fallout (FP rate)	1.000	0.647	0.553	0.467	0.407	0.373	0.313	0.233	0.153	0.080	0.000
F_1 **score**	0.629	0.659	0.654	0.627	0.565	0.524	0.494	0.404	0.276	0.109	0.000
MCC	0.000	0.252	0.264	0.244	0.175	0.139	0.139	0.100	0.047	-0.033	0.000
Cohen's kappa	0.000	0.209	0.241	0.237	0.175	0.139	0.137	0.092	0.037	-0.018	0.000
RMSE	0.736	0.642	0.624	0.622	0.642	0.653	0.650	0.661	0.674	0.688	0.677
ROC AUC	0.622										
Precision/ recall AUC ratio	1.100										

Glossary

Access audit:
Quality criterion (McCall). Those attributes of the software that provide for an audit of the access of software and data. [183]

Access control:
Quality criterion (McCall). Ability of the system to determine what resources each user should be able to access. [184]

Accessibility:
Primitive construct (Boehm). Code possesses accessibility in the sense that it facilitates selective use of its parts. [185]

Accountability:
Primitive construct (Boehm). Code possesses the characteristic accountability to the extent that its usage can be measured. [185]

Accuracy:
Quality criterion (McCall), primitive construct (Boehm), quality sub-characteristic (FURPS). Attributes of the software that provide the required precision in calculations and outputs. [129, 183, 185]

Adaptability:
Quality sub-characteristic (FURPS). Degree to which an application can accommodate changes to its external environment. [186]

As-is utility:
High-level quality characteristic (Boehm). Addresses how well, easily, reliably and efficiently the software product can be used as-is. [130] Intermediate-level quality characteristics associated (Boehm): reliability, efficiency, human engineering. [130]

Augmentability:
Primitive construct (Boehm). Code possesses augmentability to the extent that it can easily accommodate expansion in component computational functions or data storage requirements. [185]

Central Limit Theorem:
Statistics Theorem. The properly normalised sum of a large enough number of independent random variables tends toward a normal distribution even if the variables themselves are not normally distributed. [187, 188]

Communication commonality:

Quality factor (McCall). Software attribute that provides the use of standard protocols and interface routines. [183]

Communicativeness:
Quality factor (McCall), primitive construct (Boehm). Code possesses communicativeness to the extent that it facilitates the specification of inputs and provides outputs whose form and content are easy to assimilate and useful. [183, 185]

Compatibility:
Quality sub-characteristic (FURPS). Degree to which an application can be integrated into or used with other applications, products or systems. [186]

Completeness:
Quality criterion (McCall), primitive construct (Boehm). Complete information is defined by: (1) No information is left unstated or to be determined, (2) The information does not contain any undefined objects or entities and (3) No information is missing. [185, 189]

Conciseness:
Quality criterion (McCall), primitive construct (Boehm). Software attribute that allows for the implementation of a function using a minimal amount of code and with no excess of information. [183, 185]

Configurability:
Quality sub-characteristic (FURPS) Degree to which an application can be easily set up or configured for a particular application or intended use. [186]

Consistency:
Quality criterion (McCall), primitive construct (Boehm), quality sub-characteristic (FURPS). Refers to situations where a specification contains no internal contradictions. [129, 189] Comprises internal consistency, to the extent in which it contains uniform notation, terminology and symbology within itself, and external consistency, in that its content is traceable to its requirements. [185]

Contextual:
High-level product property (Dromey). Deals with the external influences on the use of a component. [132] Quality attributes associated (Dromey): maintainability, reusability, portability, reliability. [132]

Correctness:
Quality factor (McCall), high-level product property (Dromey). Extent to which the software conforms to its specifications and to its declared objectives, with no violation of basic principles. [132, 190] Quality criteria associated (McCall): traceability, completeness, consistency. [129, 130] Quality attributes associated (Dromey): functionality, reliability. [132]

Data commonality:
Quality factor (McCall). Software attribute that provides the use of standard data representations. [183]

Descriptive:
High-level product property (Dromey). Measures the descriptiveness of a component. [132] Quality attributes associated (Dromey): maintainability, reusability, portability, usability. [132]

Device efficiency:
Primitive construct (Boehm). Efficiency applied at the device level. Fulfilling the stipulated purpose without wasting a device's resources. [185]

Documentation:
Quality sub-characteristic (FURPS) Documentation quality is the degree to which an application external documentation (e.g. user manuals, reference guides, online help) possesses a desired set of characteristics. [186]

Efficiency:
Quality factor (McCall), quality attribute (Dromey), quality sub-characteristic (FURPS). Extent to which the software is able to do more with less system resources (e.g. hardware, operating system, communications). [129, 190] Quality criteria associated (McCall): execution efficiency, storage efficiency. [129, 130] Primitive constructs associated (Boehm): accountability, device efficiency and accessibility. [130]

Error tolerance:
Quality criterion (McCall). Attributes of the software that provide continuity of operation under nonnominal conditions. [183]

Execution efficiency:
Quality criterion (McCall). Efficiency pertaining to the execution of a system or part of it, which is measured by the time it takes a system, or part of a system, to perform a computation. [191]

Expandability:
Quality criterion (McCall). Relative effort required to expand software capabilities and/or performance by enhancing current functions or by adding new functionality. [190]

Extensibility:
Quality sub-characteristic (FURPS) The degree to which an application's capabilities can be extended by modifying current features or adding new features. [186]

Flexibility:
Quality factor (McCall). Ease of effort for changing the software's mission, functions or data to meet changing needs and requirements. [190] Quality criteria associated (McCall): self-descriptiveness, expandability, generality. [129, 130]

Functionality:
Quality attribute (Dromey). Includes the system's feature sets, capabilities and security. [132]

Generality:
Quality criterion (McCall). Degree to which a system or part of it can be used for a wide range of intended uses. [186]

Human engineering:
Intermediate-level quality characteristic (Boehm). A code possesses human engineering to the extent in which it fulfills its purpose without wasting the user's time and energy or degrading their morale. [185] Primitive constructs associated (Boehm): robustness/integrity, accessibility, communicativeness. [129, 130, 132]

Installability:

Quality sub-characteristic (FURPS) The degree to which an application can easily be prepared for use. [186]

Instrumentation:
Quality factor (McCall). Software attributes that allow for the measurement of usage or identification of errors. [183]

Integrity:
Quality factor (McCall), primitive construct, grouped with robustness (Boehm). Extent to which the software is able to withstand intrusion by unauthorised users or software within a specified time period. [190] Quality criteria associated (McCall): access control, access audit. [129, 130]

Internal:
High-level product property (Dromey). Measures how well a component has been deployed according to its intended use. [132] Quality attributes associated (Dromey): maintainability, efficiency, reliability. [132]

Interoperability:
Quality factor (McCall). Relative effort needed to couple the software on one platform to another software and/or another platform. [190] Quality criteria associated (McCall): modularity, communication commonality, data commonality. [129]

Legibility:
Primitive construct (Boehm). Code possesses legibility to the extent that its function is easily discerned by reading the code. [185]

Localisability:
Quality sub-characteristic (FURPS) The degree to which an application can easily be adopted, preferably via preferences or options, to satisfy the needs of languages other than English and to local standards such as decimal separator, currency symbol, time zone, calendar, etc. [186]

Machine independence:
Quality criterion (McCall), primitive construct, referred to as device independence (Boehm). Attributes of the software that provide as much independence of the hardware environment and configurations as possible (e.g. physical machine the system operates in). [183, 185]

Maintainability:
Quality factor (McCall), high-level quality characteristic (Boehm), quality attribute (Dromey), quality sub-characteristic (FURPS). Ease of effort for locating and fixing a software failure within a specified time period. How easy it is to understand, modify and retest the software product. [130, 190] Quality criteria associated (McCall): simplicity, conciseness, self-descriptiveness, modularity. [129, 130] Intermediate-level quality characteristics associated (Boehm): testability, understandability, modifiability. [130, 132]

Modifiability:
Intermediate-level quality characteristic (Boehm). Code possesses the characteristic modifiability to the extent that it facilitates the incorporation of changes, once the nature of the desired change has been determined. [185] Primitive constructs associated (Boehm): structuredness, augmentability. [130]

Modularity:

Quality criterion (McCall). Attributes of the software that provide a structure of highly independent modules. Concept based on each function of the system being implemented in a unique module. [183]

Operability:
Quality factor (McCall). Software attributes that determine operation and procedures concerned with the operation of the software. [183]

Portability:
Quality factor (McCall), high-level quality characteristic (Boehm), quality attribute (Dromey). Ease of effort to transport software to another environment or platform. [132, 190] Quality criteria associated (McCall): self-descriptiveness, software system independence, machine independence. [129] Primitive constructs associated (Boehm): device independence, self-containedness. [130]

Predictability:
Quality sub-characteristic (FURPS) Degree to which each execution will potentially yield the same results. [192]

Recoverability:
Quality sub-characteristic (FURPS) An application is recoverable if it has the capability to reestablish required performance levels and recover affected data after an application or system failure [193]

Reliability:
Quality factor (McCall), intermediate-level quality characteristic (Boehm), quality attribute (Dromey). Extent to which the software will perform according to its stated objectives within a specified time period. The system's ability not to fail. [130, 190] Quality criteria associated (McCall): consistency, accuracy, error tolerance. [129, 130] Primitive constructs associated (Boehm): self-containedness, accuracy, completeness, robustness/integrity, consistency [130]

Reusability:
Quality factor (McCall), quality attribute (Dromey). Ease of effort to use the software (or its components) in other software systems and applications. [190] Quality criteria associated (McCall): self-descriptiveness, generality, modularity, software system independence, machine independence. [129]

Robustness:
Primitive construct, grouped with integrity (Boehm). Code possesses robustness to the extent that it can continue to perform despite some violation of the assumptions in its specification. [185]

Security:
Quality sub-characteristic (FURPS). The ability of the system to protect itself against accidental or deliberate intrusion. [194]

Self-containedness:
Primitive construct (Boehm). Code possesses self-containedness to the extent that it performs all its explicit and implicit (e.g. initialization, input checking, diagnostics, etc.) functions within itself. [185]

Self-descriptiveness:

Quality criterion (McCall), primitive construct (Boehm). A software product possesses self-descriptiveness to the extent that it contains enough information for a reader to determine or verify its objectives, assumptions, constraints, inputs, outputs, components, and revision status. [185]

Serviceability:
Quality sub-characteristic (FURPS). Aspects of a system design contributing to ease of diagnostics and repair. [195]

Simplicity:
Quality criterion (McCall). Implementation of functions in the most understandable manner, avoiding practices that increase complexity. [183]

Software as a service:
Software paradigm. The delivery of software as an internet-based service via a web browser, rather than as a product that must be purchased, installed, and maintained. Process in which a third party (usually a vendor) assumes all risk for providing the software application to the enterprise or organisation. [196, 197]

Software system independence:
Quality criterion (McCall). Attributes of the software that provide as much independence of the software environment as possible (e.g. operating systems, utilities, input and output routines, etc.). [183]

Storage efficiency:
Quality criterion (McCall). Efficiency pertaining to the storage of data, which is measured by the number of bytes needed to store data. [191]

Structuredness:
Primitive construct (Boehm). Code possesses structuredness to the extent it possesses a definite pattern of organization of its interdependent parts. [185]

Testability:
Quality factor (McCall), intermediate-level quality characteristic (Boehm), quality sub-characteristic (FURPS). Ease of testing the program to verify that it performs a specified function. [190] Quality criteria associated (McCall): simplicity, instrumentation, self-descriptiveness, modularity. [129, 130] Primitive constructs associated (Boehm): accountability, communicativeness, self-descriptiveness, structuredness. [129, 130, 132]

Traceability:
Quality criterion (McCall). Software engineering term that refers to documented links between software engineering work products (e.g., requirements and test cases). [128]

Training:
Quality factor (McCall). Attributes of the software that provide transition from current operation or initial familiarization. [183]

Understandability:
Intermediate-level quality characteristic (Boehm). A code possesses the characteristic understandability to the extent that its purpose is clear to the inspector. [185] Primitive constructs associated (Boehm): consistency, structuredness, conciseness, legibility. [129, 130, 132]

Usability:
Quality factor (McCall), quality attribute (Dromey), quality characteristic (FURPS). Relative ease of learning the operation of the software. [190] Quality criteria associated (McCall): operability, training, communicativeness. [129, 130] Sub-characteristics associated (FURPS): consistency, documentation.

Bibliography

[1] T. D. Hau, "Economic fundamentals of road pricing: a diagrammatic analysis, Part I — Fundamentals," *Transportmetrica*, vol. 1, no. 2, pp. 81–117, 2005.

[2] P. Blythe, "RFID for road tolling, road-use pricing and vehicle access control," in *IEE Colloquium on RFID Technology*. IET, 1999.

[3] H. M. Al-Deek, A. Radwan, A. Mohammed, and J. Klodzinski, "Evaluating the improvements in traffic operations at a real-life toll plaza with electronic toll collection," *Journal of Intelligent Transportation Systems*, vol. 3, no. 3, pp. 205–223, 1996.

[4] F. Dionori, L. Manzi, R. Frisoni, J. M. Vasallo, J. G. Sánchez, L. O. Rendueles, J. L. P. Iturriaga, and N. Patchett, "Technology Options for the European Electronic Toll Service," European Parliament, Doctorate-General for Internal Policies, Policy Department B, Structural and Cohesion Policies, Transport and Tourism, Tech. Rep., 2014.

[5] J. Klodzinski, E. Gordin, and H. Al-Deek, "Evaluation of impacts of open road tolling on main-line toll plaza," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2012, pp. 72–83, 2007.

[6] N. J. Patil and M. N. Hedaoo, "Comparative Study of Various Road Toll Collection Systems - A Review," *International Journal of Scientific Research in Science and Technology*, vol. 3, no. 1, pp. 87–94, 2017.

[7] M. A. M. Abuzwidah, "Evaluation and Modeling of the Safety of Open Road Tolling System," Ph.D. dissertation, College of Engineering and Computer Science, University of Central Florida, 2011.

[8] J. Benda, J. Hochmuth, R. Kowshik, and L. Medgyesy, "Open road tolling signing studies," in *TRB 88th Annual Meeting Compendium of Papers CD-ROM, Transportation Research Board, Washington, DC*, 2009.

[9] R. K. Carrier and M. B. Greenbaum, "Tackling enforcement and collection in today's tolling environment," *Tollways*, vol. 4, no. 2, 2007.

[10] J. Alves, J. Wolff, and C. Melton, "Keeping an eye on revenues," *Tolltrans*, pp. 52–53, 2017.

[11] L. Smith, "Life Story," *Traffic Technology International*, pp. 31–36, 2012.

[12] C. Self, "The True Costs of Manual Image Review in Electronic Toll Collection," Perceptics LPR White Paper, 2017.

[13] S. Moseley and M. Cantelli, "Violation Image Review Processing: Flexible, modular service offering for short-term or long-term needs," Xerox Corporation White Paper, 2012.

[14] L. Dlagnekov, “Video-based Car Surveillance: License Plate, Make, and Model Recognition,” Master’s thesis, University of California, San Diego, 2005.

[15] Accenture LLS, “Accenture: the growth of a global leader,” 2017. [Online]. Available: https://www.accenture.com/us-en/company-growth-global-leader

[16] ——, “Accenture timeline,” 2017. [Online]. Available: https://www.accenture.com/us-en/accenture-timeline

[17] ——, “What we do,” 2017. [Online]. Available: https://www.accenture.com/us-en/company

[18] M. Vanderschuren, “Facts and fiction about e-tolling,” *Civil Engineering*, no. September, pp. 11–13, 2012.

[19] 4icom and Steer Davis Gleave, “Study on "State of the Art of Electronic Road Tolling",” European Commission Directorate-General for Mobility and Transport, Directorate D – Logistics, maritime and land transport and passenger rights, Tech. Rep. October, 2015. [Online]. Available: https://ec.europa.eu/transport/sites/transport/files/modes/road/road_charging/doc/study-electronic-road-tolling.pdf

[20] KPMG International, “An evolution of tolling,” *KPMG Toll Benchmarking Study 2015*, 2015. [Online]. Available: https://assets.kpmg.com/content/dam/kpmg/pdf/2015/06/kpmg-toll-benchmarking-study-2015-v2.pdf

[21] A. R. Algonda, R. R. Sonar, and S. N. Bhutada, “Hybrid Solution for E-Toll Payment,” *Advances in Intelligent Systems and Computing*, no. 673, pp. 405–414, 2017.

[22] M. Negnevitsky, *Artificial Intelligence: a Guide to Intelligent Systems*, 2nd ed. Addison-Wesley, 2005.

[23] J. G. Carbonell, R. S. Michalski, T. M. Mitchell, J. Anderson, R. Banerji, G. Bradshaw, T. Dietterich, N. Haas, F. Hayes-Roth, G. Hendrix, P. Langley, D. Lenat, J. Mostow, B. Nudel, M. Rychener, R. Quinlan, H. Simon, D. Sleeman, R. Stepp, and P. Utgoff, “An Overview of Machine Learning,” in *Machine Learning: an Artificial Intelligence Approach*, J. G. Carbonell, R. S. Michalski, and T. M. Mitchell, Eds. Palo Alto, California: TIOGA Publishing Co., 1983.

[24] S. B. Kotsiantis, “Supervised Machine Learning: A Review of Classification Techniques,” *Informatica*, vol. 31, pp. 249–268, 2007.

[25] L. P. Kaelbing, M. L. Littman, and A. W. Moore, “Reinforcement Learning: A Survey,” *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[26] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, pp. 436–444, 2015.

[27] C. Sammut and G. I. Webb, *Encyclopedia of Machine Learning*, 1st ed. Springer US, 2010.

[28] G. Ericson, W. A. Rohm, A. Jenks, J. Martens, and B. Rohrer, “How to choose algorithms for Microsoft Azure Machine Learning,” Microsoft Azure Machine Learning Studio Documentation, 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/machine-learning/studio/algorithm-choice

[29] M. Swamynathan, "Step 2: Introduction to Machine Learning," in *Mastering Machine Learning with Python in Six Steps*, 2nd ed. Berkeley, CA: Apress, 2019, ch. 2, pp. 65–144.

[30] ——, "Step 6: Deep and Reinforcement Learning," in *Mastering Machine Learning with Python in Six Steps*, 2nd ed. Berkeley, CA: Apress, 2019, ch. 6, pp. 383–442.

[31] S. Gollapudi, "Chapter 1: Artificial Intelligence and Computer Vision," in *Learn Computer Vision using OpenCV: With Deep Learning CNNs and RNNs*, 1st ed. Berkeley, CA: Apress, 2019, ch. 1, pp. 1–30.

[32] ——, "Chapter 3: Deep Learning for Computer Vision," in *Learn Computer Vision using OpenCV: With Deep Learning CNNs and RNNs*, 1st ed. Berkeley, CA: Apress, 2019, ch. 3, pp. 51–70.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances In Neural Information Processing Systems*, pp. 1–9, 2012.

[34] A. Deshpande, "A Beginner's Guide To Understanding Convolutional Neural Networks," 2016. [Online]. Available: https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/

[35] M. Nielsen, *Neural Networks and Deep Learning*, 2017. [Online]. Available: http://neuralnetworksanddeeplearning.com/index.html

[36] S. I. Serengil, "A Gentle Introduction to Convolutional Neural Networks," 2017. [Online]. Available: http://sefiks.com/2017/11/03/a-gentle-introduction-to-convolutional-neural-networks/

[37] U. Michelluci, "Chapter 8: Convolutional and Recurrent Neural Networks," in *Applied Deep Learning: A Case-Based Approach to Understanding Deep Neural Networks*, 1st ed. Berkeley, CA: Apress, 2018, ch. 8, pp. 323–364.

[38] D. Chicco, "Ten quick tips for machine learning in computational biology," *BioData Mining*, vol. 10, no. 1, pp. 1–17, 2017.

[39] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation," in *Australasian Joint Conference on Artificial Intelligence*, 2006, pp. 1015–1021.

[40] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing and Management*, vol. 45, no. 4, pp. 427–437, 2009.

[41] T. Byrt, J. Bishop, and J. B. Carlin, "Bias, prevalence and kappa," *Journal of Clinical Epidemiology*, vol. 46, no. 5, pp. 423–429, 1993.

[42] L. Flight and S. A. Julious, "The disagreeable behaviour of the kappa statistic," *Pharmaceutical Statistics*, vol. 14, no. 1, pp. 74–78, 2015.

[43] B. Engelman, E. Hayden, and D. Tasche, "Measuring the Discriminative Power of Rating Systems," Discussion paper, Series 2: Banking and Financial Supervision, Deutsche Bundesbank and University of Vienna, Tech. Rep. 01, 2003.

[44] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets," *PLoS ONE*, vol. 10, no. 3, pp. 1–21, 2015.

[45] S. García, A. Fernández, J. Luengo, and F. Herrera, "A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability," *Soft Computing*, vol. 13, pp. 959–977, 2009.

[46] C. E. Metz, "Basic principles of ROC analysis," *Seminars in Nuclear Medicine*, vol. 8, no. 4, pp. 283–298, 1978.

[47] D. Powers, "Evaluation: From Precision, Recall and F-Measure To ROC, Informedness, Markedness & Correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[48] J. V. White, S. Steingold, and C. G. Fournelle, "Performance Metrics for Group-Detection Algorithms," in *Proceedings of Interface 2004*, 2004.

[49] Data School, "Simple guide to confusion matrix terminology," 2018. [Online]. Available: http://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/

[50] J. Parker, *Algorithms for Image Processing and Computer Vision*, 2nd ed. Wiley Publishing, Inc., 2011.

[51] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[52] T. Saito and M. Rehmsmeier, "Classifier evaluation with imbalanced datasets," 2017. [Online]. Available: https://classeval.wordpress.com/

[53] Y. Sasaki, "The truth of the F-measure," *Teach Tutor mater*, vol. 1, no. 5, 2007.

[54] D. M. W. Powers, "What the F-measure doesn't measure: Features, Flaws, Fallacies and Fixes," Tech. Rep., 2015.

[55] J. Davis and M. Goadrich, "The relationship between Precision-Recall and ROC curves," *Proceedings of the 23rd international conference on Machine learning - ICML '06*, pp. 233–240, 2006.

[56] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[57] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[58] S.-T. Yeh, "Using Trapezoidal Rule for the Area under a Curve Calculation," in *Proceedings of the 27th Annual SAS® User Group International (SUGI'02)*, 2002.

[59] D. M. W. Powers, "The problem with kappa," *Conference of the European Chapter of the Association for Computational Linguistics*, pp. 345–355, 2012.

[60] J. Sim and C. C. Wright, "The Kappa Statistic in Reliability Studies : Use, Interpretation, and Sample Size Requirements," *Physical Therapy*, vol. 85, no. 3, pp. 257–268, 2005.

[61] P. Baldi, S. Brunak, Y. Chauvin, C. A. F. Andersen, and H. Nielsen, "Assessing the accuracy of prediction algorithms for classification: an overview." *Bioinformatics*, vol. 16, no. 5, pp. 412–424, 2000.

[62] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance," *Climate Research*, vol. 30, no. 1, pp. 79–82, 2005.

[63] J. S. Armstrong and F. Collopy, "Error Measures For Generalizing About Forecasting Methods: Empirical Comparisons," *International Journal of Forecasting*, vol. 8, no. 1, pp. 69–80, 1992.

[64] T. S. Huang, "Computer Vision: Evolution and Promise," *University of Illinois*, p. 5, 1997.

[65] J. Gage, "Introduction to Computer Vision," Algorithmia Inc., 2018. [Online]. Available: https://blog.algorithmia.com/introduction-to-computer-vision/

[66] H. Ling, H. Cheng, Q. Ma, F. Zou, and W. Yan, "Efficient Image Copy Detection Using Multi-Scale Fingerprints," IEEE MultiMedia, 2012.

[67] D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital data," in *Annual International Cryptology Conference*. Springer, 1995, pp. 452–465.

[68] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 6, pp. 1092–1104, 2009.

[69] S. Srinivasan and N. Sawant, "Finding Near-duplicate Images on the Web using Fingerprints," *Proceeding of the 16th ACM international conference on Multimedia MM 08*, pp. 881–884, 2008. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1459359.1459512

[70] S. Gollapudi, "Chapter 5: Object Detection and Recognition," in *Learn Computer Vision using OpenCV: With Deep Learning CNNs and RNNs*, 1st ed. Berkeley, CA: Apress, 2019, ch. 5, pp. 1–30.

[71] A. L. Kabade and V. G. Sangram, "Canny edge detection algorithm," *International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)*, vol. 5, no. 5, pp. 1292–1295, 2016.

[72] N. Senthilkumaran and R. Rajesh, "Edge detection techniques for image segmentation – a survey of soft computing approaches," *International Journal of Recent Trends in Engineering and Technology*, vol. 1, no. 2, pp. 250–254, 2009.

[73] S.Vijayarani and M.Vinupriya, "Performance Analysis of Canny and Sobel Edge Detection Algorithms in Image Mining," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 1, no. 8, pp. 1760–1767, 2013.

[74] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[75] G. T. Shrivakshan and C. Chandrasekar, "A Comparison of various Edge Detection Techniques used in Image Processing," *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 5, pp. 269–276, 2012.

[76] R. Maini and H. Aggarwal, "Study and comparison of various image edge detection techniques," *International Journal of Image Processing*, vol. 3, no. 1, pp. 1–11, 2009.

[77] A. L. Kabade and V. G. Sangram, "Canny edge detection algorithm," *International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)*, vol. 5, no. 5, pp. 1–7, 2002.

[78] M. Trajković and M. Hedley, "Fast corner detection," *Image and Vision Computing*, vol. 16, no. 2, pp. 75–87, 1998.

[79] E. Rosten and T. Drummond, "Machine Learning for High Speed Corner Detection," *Computer Vision – ECCV 2006*, vol. 1, pp. 430–443, 2006.

[80] H. Wang and M. Brady, "Real-time corner detection algorithm for motion estimation," *Image and Vision Computing*, vol. 13, no. 9, pp. 695–703, 1995.

[81] C. Schmid, R. Mohr, and C. Bauckhage, "Evaluation of interest point detectors," *International Journal of Computer Vision*, vol. 37, no. 2, pp. 151–172, 2000.

[82] P.-L. Shui and W.-C. Zhang, "Corner detection and classification using anisotropic directional derivative representations," *IEEE Transactions of Image Processing*, vol. 22, no. 8, pp. 3204–3218, 2013.

[83] H. P. Moravec, "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover," Stanford Artificial Intelligence Laboratory, Computer Science Department, Stanford University, Tech. Rep., 1980.

[84] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," in *Procedings of the Alvey Vision Conference 1988*. Alvey Vision Club, 1988, pp. 147–152.

[85] J. Corso, "Local Image Features," University of Michigan, College of Engineering, Electrical Engineering and Computer Science, EECS 598-08, Foundations of Computer Vision, 2014.

[86] K. Jeong and H. Moon, "Object detection using FAST corner detector based on smartphone platforms," in *Proceedings of the First ACIS/JNU International Conference on Computers, Networks, Systems, and Industrial Engineering, CNSI 2011*, 2011, pp. 111–115.

[87] S. Seth, P. Upadhyay, R. Shroff, and R. Komatwar, "Review of Content Based Image Retrieval Systems," *International Journal of Engineering Trends and Technology (IJETT)*, vol. 19, no. 4, pp. 178–181, 2015.

[88] F. P. Yang and M. L. Hao, "Effective image retrieval using texture elements and color fuzzy correlogram," *Information (Switzerland)*, vol. 8, no. 1, 2017.

[89] J. E. C. Cruz, E. H. Shiguemori, and L. N. F. Guimarães, "A comparison of Haar-like, LBP and HOG approaches to concrete and asphalt runway detection in high resolution imagery," *Journal of Computational Interdisciplinary Sciences*, vol. 6, pp. 121–136, 2015.

[90] T. Sikora, "The MPEG-7 visual standard for content description - An overview," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 6, pp. 696–702, 2001.

[91] S. Belongie, J. Malik, and J. Puzicha, "Shape context: A new descriptor for shape matching and object recognition," *Advances in Neural Information Processing Systems*, vol. 546, pp. 831–837, 2000.

[92] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 2001.

[93] R. Lienhart and J. Maydt, "An Extended Set of Haar-like Features for Rapid Object Detection," in *Proceedings of the 2002 Iternational Conference on Image Processing*, 2002.

[94] B. Vidakovic and P. Mueller, "Wavelets for kids: a tutorial introduction," Institute of Statistics and Decision Sciences, Duke University, 1991.

[95] D. S. Khadtare, M. M. Jadhav, and M. Khadtare, "A robust watermarking approach for raw video and its DSP implementation," *International Journal of Advanced Engineering Research and Studies*, vol. 1, no. 1, pp. 198–208, 2011.

[96] C. P. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," in *IEEE International Conference on Computer Vision*, 1998, pp. 555–562.

[97] S.-K. Pavani, D. Delgado, and A. F. Frangi, "Haar-like features with optimally weighted rectangles for rapid object detection," *Pattern Recognition*, vol. 43, no. 1, pp. 160–172, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2009.05.011

[98] S. G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, 1989.

[99] S. Sivaraman and M. M. Trivedi, "A Review of Recent Developments in Vision-Based Vehicle Detection," in *IEEE Intelligent Vehicles Symposium*, no. IV, Gold Coast, Australia, 2013.

[100] G. Bradski and A. Kaehler, *Learning OpenCV*. O'Reilly, 2008.

[101] T. Ojala, M. Pietikäinen, and D. Harwood, "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions," *Proceedings of 12th International Conference on Pattern Recognition*, vol. 1, pp. 582–585, 1994.

[102] ——, "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[103] L. Wang and D.-C. He, "Texture classification using texture spectrum," *Pattern Recognition*, vol. 23, no. 8, pp. 905–910, 1990.

[104] S. L. Happy, A. George, and A. Routray, "A real time facial expression classification system using Local Binary Patterns," *4th International Conference on Intelligent Human Computer Interaction (IHCI)*, pp. 1–5, 2012.

[105] A. Mordvintsev and A. Rahman, "Face Detection using Haar Cascades," OpenCV - Python Tutorials, 2013. [Online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html

[106] D. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 1999, pp. 1150–1157 vol.2.

[107] D. G. Lowe, "US Patent No. US006711293B1: Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image," The University of British Columbia, Vancouver (CA), pp. 1–12, 2004.

[108] J. Li and N. M. Allinson, "A comprehensive review of current local features for computer vision," *Neurocomputing*, vol. 71, pp. 1771–1787, 2008.

[109] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[110] Y. Ke and R. Sukthankar, "PCA-SIFT: A More Distinctive Representation for Local Image Descriptors," *Computer Vision and Pattern Recognition*, vol. 2, pp. 506–513, 2004.

[111] E. N. Mortensen, H. Deng, and L. Shapiro, "A SIFT Descriptor with Global Context," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, 2005, pp. 184–190.

[112] T. Lindeberg, "Scale Invariant Feature Transform," *Scholarpedia*, vol. 7, no. 5, p. 10491, 2012, revision #153939.

[113] J. Wu, Z. Cui, V. S. Sheng, P. Zhao, D. Su, and S. Gong, "A comparative study of SIFT and its variants," *Measurement Science Review*, vol. 13, no. 3, pp. 122–131, 2013.

[114] W. Cheung and G. Hamarneh, "n-SIFT: n-Dimensional scale invariant feature transform," *IEEE Transactions on Image Processing*, vol. 18, no. 9, pp. 2012–2021, 2009.

[115] L. Juan and O. Gwun, "A comparison of SIFT, PCA-SIFT and SURF," *International Journal of Image Processing (IJIP)*, vol. 3, no. 4, pp. 143–152, 2009.

[116] A. E. Abdel-Hakim and A. A. Farag, "CSIFT: A SIFT Descriptor with Color Invariant Characteristics," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, 2006, pp. 1978–1983.

[117] J.-M. Geusebroek, R. van den Boomgaard, A. W. M. Smeulders, and H. Geerts, "Color Invariance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 12, pp. 1338–1350, 2001.

[118] J.-M. Morel and G. Yu, "ASIFT : A New Framework for Fully Affine Invariant Image Comparison," *SIAM Journal on Imaging Sciences*, vol. 2, no. 2, pp. 438–469, 2009.

[119] R. Funayama, H. Yanagihara, L. Van Gool, T. Tuytelaars, and H. Bay, "US Patent No. US008165401B2: Robust Interest Point Detector and Descriptor," Toyota Motor Europe NV, Brussels (BE); K.U. Leuven Research & Development, Leuven (BE); Eidgenoessiche Technische Hochschule Zurich, Zurich (CH), pp. 1–13, 2012.

[120] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF : Speeded Up Robust Features," in *European Conference on Computer Vision*, 2006, pp. 404–417.

[121] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary Robust Independent Elementary Features," in *European Conference on Computer Vision*, 2010, pp. 778–792.

[122] J. Liu and X. Liang, "I-BRIEF: A fast feature point descriptor with more robust features," *Proceedings - 7th International Conference on Signal Image Technology and Internet-Based Systems, SITIS 2011*, pp. 322–328, 2011.

[123] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2564–2571, 2011.

[124] H. J. Chien, C. C. Chuang, C. Y. Chen, and R. Klette, "When to use what feature? SIFT, SURF, ORB, or A-KAZE features for monocular visual odometry," *International Conference Image and Vision Computing New Zealand*, no. 1, pp. 0–5, 2017.

[125] P. Fernández Alcantarilla, A. Bartoli, and A. J. Davison, "KAZE Features," in *ECCV 2012, Part VI, LNCS 7577*, 2012, pp. 214–227.

[126] Z. Saleem, "A Comparative Analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK," in *International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, 2018.

[127] D. Galin, *Software Quality Assurance - From theory to implementation*, 1st ed. Addison-Wesley, 2004.

[128] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practicioner's Approach*, 8th ed. McGraw Hill Education, 2015.

[129] D. Samadhiya, S.-H. Wang, and D. Chen, "Quality models: Role and value in software engineering," *International Conference on Software Technology and Engineering (ICSTE)*, vol. 1, pp. 320–324, 2010.

[130] R. E. Al-Qutaish, "Quality models in software engineering literature: an analytical and comparative study," *Journal of American Science*, vol. 6, no. 3, pp. 166–175, 2010.

[131] A. B. Al-Badareen, M. H. Selamat, M. A. Jabar, J. Din, and S. Turaev, "Software Quality Models: A Comparative Study," in *ICSECS (1)*, no. June, 2011, pp. 46–55.

[132] D. Jamwal, "Analysis of Software Quality Models for Organizations," *International Journal of Latest Trends in Computing (E-ISSN: 2045-5364)*, vol. 1, no. 2, pp. 19–23, 2010.

[133] D. Firesmith, "Journal of Object Technology - Prioritizing Requirements," *Journal of Object Technology*, vol. 3, no. 8, pp. 35–47, 2004. [Online]. Available: http://www.jot.fm/issues/issue_2004_09/column4

[134] E. Miranda, "Time boxing planning: Buffered Moscow rules," *ACM SIGSOFT Software Engineering Notes*, vol. 36, no. 6, pp. 1–5, 2011.

[135] P. Achimugu, A. Selamat, R. Ibrahim, and M. N. Naz, "A systematic literature review of software requirements prioritization research," *Information and Software Technology*, vol. 56, no. 6, pp. 568–585, 2014.

[136] J. Karlsson, C. Wohlin, and B. Regnell, "An evaluation of methods for prioritizing software requirements," *Information and Software Technology*, vol. 39, pp. 939–947, 1998.

[137] R. Baran, A. Glowacz, and A. Matiolanski, "The efficient real- and non-real-time make and model recognition of cars," *Multimedia Tools and Applications*, vol. 74, no. 12, pp. 4269–4288, 2015.

[138] A. Minich and C. Li, "Vehicle Logo Recognition and Classification: Feature Descriptors vs. Shape Descriptors," *Spring*, pp. 1–6, 2011. [Online]. Available: http://www.stanford.edu/class/ee368/Project_11/Reports/Burkhard_Li_Minich_Automated_Vehicle_Identification.pdf

[139] Y. Wang, Z. Liu, and F. Xiao, "A fast coarse-to-fine vehicle logo detection and recognition method," in *2007 IEEE International Conference on Robotics and Biomimetics, ROBIO*, no. 1, 2007, pp. 691–696.

[140] K.-T. Sam and X.-L. Tian, "Vehicle Logo Recognition Using Modest AdaBoost and Radial Tchebichef Moments," in *Proceedings of 2012 4th International Conference on Machine Learning and Computing IPCSIT*, vol. 25, 2012, pp. 91–95.

[141] Y. Huang, R. Wu, Y. Sun, W. Wang, and X. Ding, "Vehicle logo recognition system based on convolutional neural networks with a pretraining strategy," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 1951–1960, 2015.

[142] A. Psyllos, C. N. Anagnostopoulos, and E. Kayafas, "Vehicle model recognition from frontal view image measurements," *Computer Standards and Interfaces*, vol. 33, no. 2, pp. 142–151, 2011.

[143] C. N. E. Anagnostopoulos, I. E. Anagnostopoulos, V. Loumos, and E. Kayafas, "A License Plate-Recognition Algorithm for Intelligent Transportation System Applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 3, pp. 377–392, 2006.

[144] J. Sauvola and M. Pietikäinen, "Adaptive document image binarization," *Pattern Recognition*, vol. 33, pp. 225–236, 2000.

[145] G. Lazzara and T. Géraud, "Efficient Multiscale Sauvola's Binarization," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 17, no. 2, pp. 105–123, 2014.

[146] K. Suzuki, I. Horiba, and N. Sugie, "Linear-time connected-component labeling based on sequential local operations," *Computer Vision and Image Understanding*, vol. 89, pp. 1–23, 2003.

[147] C. N. Anagnostopoulos, "Artificial vision and computational intelligence techniques for industrial applications and quality control," Ph.D. dissertation, Electrical and Computer Engineering Department., National Technical University of Athens, 2002.

[148] G. Pearce and N. Pears, "Automatic Make and Model Recognition from Frontal Images of Cars," in *8th IEEE International Conference on Advanced Video and Signal-Based Surveillance*, 2011, pp. 373–378.

[149] D. T. Munroe and M. G. Madden, "Multi-class and single-class classification approaches to vehicle model recognition from images," *Proceedings of IEEE AICS*, 2005.

[150] V. S. Petrović and T. F. Cootes, "Analysis of Features for Rigid Structure Vehicle Type Recognition," in *British Machine Vision Conference (BMVC'04)*, 2004, pp. 587–596.

[151] J. A. Noble, "Descriptions of image surfaces," Ph.D. dissertation, University of Oxford, 1989.

[152] K. Ramnath, S. N. Sinha, R. Szeliski, and E. Hsiao, "Car make and model recognition using 3D curve alignment," in *2014 IEEE Winter Conference on Applications of Computer Vision, WACV 2014*, 2014, pp. 285–292.

[153] E. Hsiao, S. N. Sinha, K. Ramnath, S. Baker, L. Zitnick, and R. Szeliski, "Car make and model recognition using 3D curve alignments," Tech. Rep. MSR-TR-2014-9, 2014.

[154] R. Newcombe, "From Client to Project Stakeholders: a Stakeholder Mapping Approach," *Construction Management and Economics*, vol. 21, no. 8, pp. 841–848, 2003.

[155] L. Chung and J. C. S. d. P. Leite, "On Non-Functional Requirements in Software Engineering," *Conceptual modeling: Foundations and applications*, pp. 363–379, 2009.

[156] M. Glinz, "On Non-Functional Requirements," *15th IEEE International Requirements Engineering Conference (RE 2007)*, pp. 21–26, 2007.

[157] EasyGo, "Roadside and on board equipment: Annex 2.2 to Joint Venture Agreement/Toll Service Provider Agreement," 2013.

[158] TransportPolicy.net, "US: Vehicle Definitions," Collaboration between the International Council on Clean Transportation and DieselNet, 2018. [Online]. Available: https://www.transportpolicy.net/standard/us-vehicle-definitions/

[159] Electronic Transaction Consultants Corporation, "ETC Has Your Solution," 2017. [Online]. Available: http://www.etcc.com/solutions/

[160] R. A. S. G. de Almeida, A. R. R. Pinto, and R. R. Figueira, "US Patent No. US201703722161A1: Intelligent Automatic License Plate Recognition for Electronic Tolling Environments," Accenture Global Solutions Limited, Dublin, Ireland, 2017.

[161] H. Kaur, "Automating Static Code Analysis for Risk Assessment and Quality Assurance of Medical Record Software," Master's thesis, University of Victoria, 2017.

[162] SonarSource S.A., "Code Quality," 2018. [Online]. Available: https://www.sonarsource.com/why-us/code-quality/

[163] RedHat, "Arquillian – So you can rule your code. Not the bugs." 2018. [Online]. Available: http://arquillian.org/

[164] S. Chacon and B. Straub, *Pro Git*, 2nd ed. Apress, 2014.

[165] Atlassian, "SourceTree: Simplicity and power in a beautiful Git GUI," 2018. [Online]. Available: https://www.sourcetreeapp.com/

[166] Jenkins Project, "Jenkins: Build great things at any scale," 2018. [Online]. Available: https://jenkins.io/

[167] RedHat, "JBoss Middleware: Red Hat JBoss Operations Network," 2018. [Online]. Available: https://www.redhat.com/en/technologies/jboss-middleware/operations-network

[168] Sonatype, "Nexus Repository: Expert flow control for binaries, build artifacts, and release candidates," 2018. [Online]. Available: https://www.sonatype.com/nexus-repository-sonatype

[169] Phacility, "A powerful software development platform," 2018. [Online]. Available: https://www.phacility.com/

[170] Z. Antolić, "An Example of Using Key Performance Indicators for Software Development Process Efficiency Evaluation," R&D Center Ericsson Nikola Tesla, 6, 2008.

[171] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3D object representations for fine-grained categorization," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 554–561, 2013. [Online]. Available: https://ai.stanford.edu/~jkrause/cars/car_dataset.html

[172] W. Clarke, "Top 10 Best-Selling Vehicles of 2006," Edmunds Inc. (retrieved via Wayback Machine), 2008. [Online]. Available: https://web.archive.org/web/20080417180145/http://www.edmunds.com/reviews/list/top10/120637/article.html

[173] B. Zhang, "The 20 best-selling cars and trucks in America," Business Insider, 2017. [Online]. Available: http://www.businessinsider.com/best-selling-cars-trucks-vehicle-america-2016-2017-1/

[174] P. F. Alcantarilla, J. Nuevo, and A. Bartoli, "Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces," in *British Machine Vision Conference (BMVC)*, 2013.

[175] OpenCV Dev Team, "Common Interfaces of Descriptor Matchers," OpenCV API Reference, 2018. [Online]. Available: https://docs.opencv.org/2.4/modules/features2d/doc/common_interfaces_of_descriptor_matchers.html

[176] E. Bostanci, "Is Hamming distance the only way for matching binary image feature descriptors?" *Electronics Letters*, vol. 50, no. 11, pp. 806–808, 2014.

[177] A. Mordvintsev and A. Rahman, "Feature Matching," OpenCV – Python Tutorials, 2013. [Online]. Available: http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html

[178] T. Ball and R. Mehner, "Train your own OpenCV Haar classifier," Coding Robin, 2013. [Online]. Available: https://coding-robin.de/2013/07/22/train-your-own-opencv-haar-classifier.html

[179] T. Ball, "Train your own OpenCV Haar classifier: Instructions," GitHub, 2017. [Online]. Available: https://github.com/mrnugget/opencv-haar-classifier-training

[180] OpenCV Dev Team, "Cascade Classifier Training," 2018. [Online]. Available: https://docs.opencv.org/2.4/doc/user_guide/ug_traincascade.html

[181] N. Seo, "Tutorial: OpenCV haartraining (Rapid Object Detection With A Cascade of Boosted Classifiers Based on Haar-like Features)," 2012. [Online]. Available: http://note.sonots.com/SciSoftware/haartraining.html

[182] K. Kadir, M. K. Kamaruddin, H. Nasir, S. I. Safie, and Z. A. K. Bakti, "A comparative study between LBP and Haar-like features for Face Detection using OpenCV," *2014 4th International Conference on Engineering Technology and Technopreneuship, ICE2T 2014*, vol. 2014-Augus, no. August, pp. 335–339, 2014.

[183] J. A. McCall, P. K. Richards, and G. F. Walters, "Factors in Software Quality - Volume 1 - Concept and Definitions of Software Quality," *Defense Technical Information Center*, vol. 1, 1977.

[184] G. McGraw and J. Viega, "Building secure software: How to avoid security problems the right way," in *RTO/NATO Real-Time Intrusion Detection Symposium*, 2002.

[185] B. W. Boehm, J. R. Brown, and M. Lipow, "Quantitative evaluation of software quality," *Proceedings of the 2nd international conference on Software engineering*, pp. 592–605, 1976.

[186] O. Balci, "Quality assessment, verification, and validation of modeling and simulation applications," *Proceedings of the 36th Winter Simulation Conference, 2004*, pp. 122–129, 2004.

[187] A. Ghasemi and S. Zahediasl, "Normality Tests for Statistical Analysis: A Guide for Non-Statisticians," *International Journal of Endocrinology and Metabolism*, vol. 10, no. 2, pp. 486–489, 2012.

[188] A. C. Bovik, *The Essential Guide to Image Processing*, 1st ed. Academic Press, Elsevier, 2009.

[189] D. Zowghi and V. Gervasi, "The Three Cs of Requirements: Consistency, Completeness, and Correctness," *Proceedings of 8th International Workshop on Requirements Engineering: Foundation for Software Quality*, no. March, pp. 155–164, 2002.

[190] N. Ashrafi, "The impact of software process improvement on quality: in theory and practice," *Information & Management*, vol. 40, no. 7, pp. 677–690, 2003.

[191] W. Frakes, "Introduction to information storage and retrieval systems," *Space*, vol. 14, no. 10, 1992.

[192] M. Harman, "The current state and future of search based software engineering," in *FoSE 2007: Future of Software Engineering*, 2007, pp. 342–357.

[193] I. Gorton, "Software Quality Attributes," in *Essential Software Architecture*, 2006, pp. 1–283.

[194] L. Chung, "Non-Functional Requirements," University of Texas at Dallas, Department of Computer Science, CS6361/501, (Advanced) Requirements Engineering, 2006.

[195] A. M. Johnson Jr. and M. Malek, "Survey of Software and Serviceability Tools for Evaluating Reliability, Availability, and Serviceability," *ACM Computing Surveys*, vol. 20, no. 4, 1988.

[196] G. St. Clair, "Software-as-a-Service (SaaS): Put the Focus on the KM/Knowledge Services Core Function," SaaS White Paper, EOS International, Tech. Rep., 2008.

[197] C. Spence, J. Devoys, and S. Chahal, "Architecting Software as a Service for the Enterprise," IT@Intel White Paper, Cloud Computing, Intel Information Technology, Intel Corporation, Tech. Rep., 2009.