

Faculty of Engineering and Environment

Department of Computer and Information Sciences

CM0645/KV6003 Individual Project

2018-2019

NAME: Abdulelah Alshehri 14035393

PROGRAMME: BSc Computer Science

PROJECT TYPE: Software Engineering Project

SUPERVISOR: Prof. Ahmed Bouridane

2ND MARKER: Dr. Longzhi Yang

PROJECT TITLE:

Vision-Based Detection of Parking Spaces Occupation and Violation

Declarations

I declare the following:

(1) that the material contained in this dissertation is the end result of my own work and that due acknowledgement has been given in the bibliography and references to **ALL** sources be they printed, electronic or personal.

(2) the Word Count of this Dissertation is 12,249

(3) that unless this dissertation has been confirmed as confidential, I agree to an entire electronic copy or sections of the dissertation to being placed on the eLearning Portal (Blackboard), if deemed appropriate, to allow future students the opportunity to see examples of past dissertations.  I understand that if displayed on eLearning Portal it would be made available for no longer than five years and that students would be able to print off copies or download.

(4) I agree to my dissertation being submitted to a plagiarism detection service, where it will be stored in a database and compared against work submitted from this or any other School or from other institutions using the service.

In the event of the service detecting a high degree of similarity between content within the service this will be reported back to my supervisor and second marker, who may decide to undertake further investigation that may ultimately lead to disciplinary actions, should instances of plagiarism be detected.

(5) I have read the Northumbria University/Engineering and Environment Policy Statement on Ethics in Research and Consultancy and I confirm that ethical issues have been considered, evaluated and appropriately addressed in this research.

**SIGNED:**

Abdulelah Alshehri 28/05/2019

Acknowledgements

First and foremost, I would like to thank my family, whom presence gave me the motivation and spirit to conquer all challenges faced.

Additionally, I would like to thank my supervisor, Prof. Ahmd Bouridane, for his guidance and enthusiasm throughout this project, which gave me the motivation to learn and challenge myself with new ideas and cutting-edge technology.

Finally, I would like to give special appreciation to my best friend, Abdullah Alshahrani, ttat; for his presence when I needed him the most, as without his support I would have been crushed under the stress and anxiety I faced throughout the year.

Abstract

Due to the sheer number of vehicles driving through cities and the increasing population in urban areas, the need to efficiently manage parking spaces has become apparent, to achieve the aim of giving driver the best possible experience with least delay. However, car park faces many issues where they are left unattended, especially, open car parks where entry is open for the public. This places tremendous stress on parking sites in highly populated areas, especially in peak times. One solution to this problem is to improve the efficiency of the existing parking spaces and how they are managed, by enabling site managers to be in direct contact with live information about the current occupation and violations of their chosen site.

This was achieved by using camera surveillance as an input for vehicle detection and classification in the developed system, which allows its user to assign parking spaces and customise their type, which can be used to raise alert in the case of incompatible parked vehicle with a parking space type. This stops cars from parking in spaces allocated for trucks or motorbikes.

The system implements state-of-art object detection and classification algorithm, YOLOv3, that can detect and classify objects in real-time. The domain of this application is not just parking management system but can easily be adapted to other application, such as people detection and tracking.

Contents

[Chapter 1) Introduction 8](#_Toc9938205)

[1.1 Aims and Objectives 8](#_Toc9938206)

[1.2 Overview – Project purpose, scope and main features 9](#_Toc9938207)

[1.3 Context of the Problem 10](#_Toc9938208)

[1.4 Implemented Tools and Techniques 10](#_Toc9938209)

[Chapter 2) Literature Review 11](#_Toc9938210)

[2.1 Overview 11](#_Toc9938211)

[2.2 Parking Space and Related Problems 11](#_Toc9938212)

[2.3 Existing Parking Space Detection Systems 12](#_Toc9938213)

[2.4 Challenges and Limitations in Implementation of Smart Parking Space Detection System 13](#_Toc9938214)

[2.5 Vision-Based object Detection with Convolutional Neural Networks 14](#_Toc9938215)

[2.6 Summary 15](#_Toc9938216)

[Chapter 3) Requirement Specification 17](#_Toc9938217)

[3.1 Introduction 17](#_Toc9938218)

[3.2 Commentary 17](#_Toc9938219)

[3.2.1 Overall characteristics 17](#_Toc9938220)

[3.2.2 YOLO Prediction framework 17](#_Toc9938221)

[3.2.3 Performance and Redeployment 19](#_Toc9938222)

[3.2.4 Storage and Privacy 19](#_Toc9938223)

[3.2.5 User Interface (11.2.2 & 11.2.6) 19](#_Toc9938224)

[3.3 Summary 19](#_Toc9938225)

[Chapter 4) System modelling & design 20](#_Toc9938226)

[4.1 Introduction 20](#_Toc9938227)

[4.2 Class Diagram 20](#_Toc9938228)

[4.3 Activity Diagram 20](#_Toc9938229)

[4.4 Dictionary and Data Transfer Objects (DTOs) 20](#_Toc9938230)

[4.5 YOLOv3 network Structure 21](#_Toc9938231)

[4.5.1 Configuration file 22](#_Toc9938232)

[4.5.2 Training dataset 23](#_Toc9938233)

[4.6 User Interface 23](#_Toc9938234)

[4.7 Summary 23](#_Toc9938235)

[Chapter 5) System development 24](#_Toc9938236)

[5.1 Introduction 24](#_Toc9938237)

[5.2 Main program 24](#_Toc9938238)

[5.3 Detection Module 24](#_Toc9938239)

[5.3.1 Video input and output 25](#_Toc9938240)

[5.3.2 Detector training and Viewer integration 27](#_Toc9938241)

[5.3.3 Introducing ROIs 30](#_Toc9938242)

[5.3.4 Integration 31](#_Toc9938243)

[5.4 User Interface 32](#_Toc9938244)

[5.5 Summary 34](#_Toc9938245)

[Chapter 6) System testing 35](#_Toc9938246)

[6.1 Introduction 35](#_Toc9938247)

[6.2 Test plan and Results 35](#_Toc9938248)

[6.3 Performance Testing 42](#_Toc9938249)

[6.4 Summary 42](#_Toc9938250)

[Chapter 7) Evaluation 43](#_Toc9938251)

[7.1 Product Evaluation 43](#_Toc9938252)

[7.1.1 Requirements evaluation 43](#_Toc9938253)

[7.1.2 Fitness for Purpose 43](#_Toc9938254)

[7.1.3 Build Quality 44](#_Toc9938255)

[7.1.4 Usability Evaluation 44](#_Toc9938256)

[7.2 Personal and Process Evaluation 45](#_Toc9938257)

[Chapter 8) Conclusion and Further Work 47](#_Toc9938258)

[8.1 Conclusion 47](#_Toc9938259)

[8.2 Recommendation for Further Work 47](#_Toc9938260)

[8.2.1 Use of Smart Parking Sensors 47](#_Toc9938261)

[8.2.2 Clustering Algorithms for the Calculation of Analytical Statistics 48](#_Toc9938262)

[Chapter 9) References 49](#_Toc9938263)

[Chapter 10) Bibliography 53](#_Toc9938264)

[Chapter 11) Appendices 55](#_Toc9938265)

[11.1 Terms of Reference 55](#_Toc9938266)

[11.2 Requirements Specifications 68](#_Toc9938267)

[11.2.1 General 68](#_Toc9938268)

[11.2.2 Dashboard Interface 68](#_Toc9938269)

[11.2.3 Video Input manager 69](#_Toc9938270)

[11.2.4 Training a model 69](#_Toc9938271)

[11.2.5 System Detection 70](#_Toc9938272)

[11.2.6 Detector Interface 70](#_Toc9938273)

[11.3 Design Documentation 71](#_Toc9938274)

[11.3.1 Class Diagram 71](#_Toc9938275)

[11.3.2 Activity Diagram 72](#_Toc9938276)

[11.3.3 System Use Case 73](#_Toc9938277)

[11.3.4 Interface wireframes 73](#_Toc9938278)

# Introduction

This project is to develop and implement a parking space occupation and violation detection system (SMART Parking), for parking site managers to help them identify issues quickly whether it is by inappropriate parking behaviour or simply the need to retrieve the occupation in real-time. The Aims and Objective, derived from the Terms of Reference (TOR, see Appendix [11.1](#_Requirements_Specifications)), are considered to be the guide by which the project’s quality as a whole is measured. The aim of the project is the high-level goal to be achieved at the completion of the project. The objectives are the lower-level goals/milestones that should be completed to fulfil the aim of the project.

## Aims and Objectives

**Aims:**

* “The development of a vehicle recognition and detection within parking spaces using a deep neural network to allow for parking occupation and violation detection”.

This will be enhanced by the inclusion of a literature review of current state-of-the-art systems and solution used to make managing parking spaces and increase its efficiency, and the current limitations of the implemented solutions.

**Objectives:**

1. **Review and evaluate existing work on smart parking systems.**

Current state-of-the-art research and implementation were reviewed discussing the limitations and recommendations. The review will indicate the feasibility of using a neural network-based approach, especially for real-time systems.

1. **Produce prioritised system specification requirements document.**

A requirement specification document will be produced detailing the requirements’ description, type and priority.

1. **Identifying and learning/improving knowledge required to complete the project (technical, academic and professionalism) - using online tutorials, labs from previous years.**

Further knowledge and research to be performed after concluding the Literature review and before starting the design phase. This was to enhance the design of the implemented system and include state-of-the-art technology.

1. **Produce system structure and behaviour documentation, detailing system’s classes (Class diagram), sub system interactions (Communication and Use Cases diagrams) and complex processes (Activity diagrams) and interface wireframes.**

The produced design documents needs to well-thought and accurate as they will be used as the blueprint, the system will be developed based upon. Naturally, unforeseen circumstances might arise limiting the implementation from following the design document. If that is the case, the design document will have higher priority to be updated first.

1. **Produce system skeleton structure design specification document.**

This will be the basic first block of code that will run all sub-systems running on top of it.

1. **Setup development environment.**

The development environment needed to develop the system. This will include Source control, integrated development environment (IDE) and framework used to train the prediction model.

1. **Development, testing and implementation of vehicle detection within parking spaces.**

The stages of system development. These will be recurrent throughout the development of the product, with a final product testing stage.

1. **Evaluate and analyse system to verify the implementation of all requirements and establish conclusion and recommendations.**

This will be the final stage after the system has passed all testing. It is evaluated to ensure that it accomplishes the set requirements.

1. **Reflection on project, product and personal management and performance.**

This will be a reflective writing on how good/bad choices made during all project stages and will be included in the evaluation section of the report.

1. **Production of project report chapters.**

This will progress throughout the year depending on the stage of the project until its completion.

## Overview – Project purpose, scope and main features

The proposed system’s main purpose is to help increase open-parking sites’ efficiency by allowing them to view their information in real-time. The information stored will consist of the current occupation rate and violations in the target parking lot.

The proposed system’s functionality will offer its users direct communication and visualisation of live video stream and allow flexible ways to set the regions of interest to be kept in check. This allows sites to not detect the full parking space occupation but just a small percentage where violations may occur and disrupt the use of the parking lot. Violations will be detected as car park administrators will be allowed to set the type of vehicle permitted in each parking space; once a different vehicle type is detected in an occupied parking space the manager will be informed.

The scope of the system will encompass the essential functionality that is required to for the detection and recognition of different vehicle types and match it to a parking space type where a violation needs to be prevented. Currently, the scope will be limited to the mentioned functionality and will not include any extensions that would allow the system to be used for other purposes, such as licence plate recognition or viewing information using a web-based interface.

## Context of the Problem

Nowadays, technology has become advanced and accessible enough that many drivers use it to be informed about their journeys’ congestion levels, closures, route recommendations, etc… all in the aim of having the best possible experience with least delay. However, many journeys end with the driver having to look for a parking space, which causes tremendous stress on parking sites in highly populated areas, especially in peak times. This is where this project comes in, with the aim to improve the how parking spaces are managed by keeping the site managers in direct contact with live information about the current occupation and violations in their parking site.

The ability to calculate the occupation rate and vehicles violations in the designated parking spaces and informing the parking administrators about them will help them find more solutions to utilise their spaces better or stop violations which disrupts the usage of other drivers. This can improve the overall experience of looking for a parking space, while reducing the time spent looking for one. As a result, improving the efficiency of the parking system and traffic flow in the area.

## Implemented Tools and Techniques

It is sufficient to briefly summarize how you approached the problem, by describing your methods and tools: you do not need to justify them here, as this will be dealt with in the analysis. You could describe the development tools and underpinning technologies that you used, the computational techniques, analysis and design techniques, etc.

The proposed system will be developed using C#, XML and C/C++ programming languages, and Windows Forms for the interface of the application. The written code will be kept in different files and included when necessary to loose coupling, improve code re-use and high modularity of the developed system. Additionally, the system will make use of configuration files to allow for quick and easy manipulations of the system settings.

The developed system will require two environments to be rebuilt; Darknet was used for the generation of the prediction model, and Visual Studio and OpenCV for the logic and integration of the prediction model into the system.

# Literature Review

## Overview

The technological advancements are being used for numerous aspects inclusive of making life easier and convenient through the integration of artificial intelligence in the daily routine. One of the neglected aspects of bringing convenience to the people is the issues faced by those driving cars, as they fail to find parking spaces easily. It has been verified that the detection of parking spaces has mainly been based on human cooperation, which can at times be frustrating for drivers due to excessive time-consumption (Abe, Ogitsu, and Mizoguchi 2016). Through this section of the research, literature is reviewed from a variety of sources. The aim is to highlight the problems that are associated with the parking space and its detection processes. Some of the methods used for automatic parking space detection are also discussed along with the challenges and limitations they present. Finally, future suggestions are made with respect to the methods that can be integrated to enhance the parking space detection systems.

## Parking Space and Related Problems

The increasing number of vehicles has generated the rising demand of parking spaces. It is found to be one of the most influential factors associated with safety of the traffic and also the traffic order in the urban areas. As discussed by (Iv et al, 2011), the lack of proper management of parking spaces leads to affected operational efficiency since the absence of fixed parking spots lead to traffic congestion and accidents in certain popular areas. In order to avoid traffic congestion and traffic related accidents it is important that rational planning and parking management is taken into picture (Iv et al., 2011). The problem is evident in the metropolitans and especially during the holiday season or on weekends, when almost all the families go out for recreational activities. The initiative taken by (Wang and He, 2011) led to the conclusion that periodic learning of the parking status brings forth the problem and requires a solution that is long-term and helps in avoiding chaotic situations. Furthermore, as highlighted by Taniguchi (2014), it is important to integrate the parking detection systems of automatic nature to provide sustainable living and convert urban metro cities into liveable ones. Thus, the application of innovative technologies corresponding to ICT and ITS is advised to bring forth the positive public-private partnerships that can promote city logistics policy measures.

In the thought-provoking research, it has been revealed that the drivers in the US spend almost 17 hours of their year in searching for a parking space (McCoy, 2017). The research carried forth and discussed by McCoy (2017) highlights the need to introduce powerful detecting methods for parking spaces so that the citizens do not have to waste their time in this regard. Moreover, this also results in overpaying, as the drivers are unaware of the estimated time that they will be parking the car for. Thus, identifying the solution to the situation is necessary, as the rate with which the number of cars is increasing on the road, there is bound to come a time that more parking would be required than the total area of recreational zones or food outlets. In the study conducted by Ionita et al. (2018), it is further highlighted that the free parking spots are difficult to be found in the cities that are unmonitored. In fact, in their research, Ionita et al. (2018) revealed that 30% of the traffic across the world is due to the cars that actively search for parking space for the cars. These results certify that there is a need to activate measures that can help curb the problem associated with parking spaces.

## Existing Parking Space Detection Systems

The smart parking system is a strategic issue and numerous studies have been conducted while new methods of detecting parking spaces are being developed. The research conducted by Lin, Rivano, and Le Mouel (2017) concluded that the inclusion of information and technology communications has made it possible to make smart parking possible. The focus on information collection, system deployment, and service dissemination forms the three-part macro-theme. Similarly, Tian et al. (2014) proceeded with the research of the intelligent parking systems based on the license plates and their recognition technology. The integration of Zigbee wireless technology has also been found associated with the process. The scholars Ma et al. (2017) discussed the automatic parking technology discussed parking space identification based on the parking scene recognition application. On the other hand, Bulan et al. (2013) devised the technology that assisted in parking detection through the video-based occupancy verification processed through real-time on-street parking. The system developed by them focused on detection of not just parking space through vehicle detection but also parking-angle violation and parking boundary violation.

Through their studies, Vlahogianni et al. (2015) highlighted that the implementation of a real-time parking prediction system for smart cities is yet another method of controlling chaos and traffic due to improper management of parking spaces. The two different methods are discussed of which the first is about analysing the probability associated with the availability of free space as free at subsequent intervals. The prediction of short-term parking occupancy in selected regions is identified as another possible way of systematised parking space detection. Vlahogianni et al. (2015) bring forth the data collection in the smart city of Santander in Spain through the use of wide network of on-street parking sensors, which favours the Weibull parametric models for determining availability of free parking space in the forthcoming intervals of time. The exploitation of 1-minute data by genetically optimised multilayer perceptron further predict the parking occupancy rates of up to 30 minutes in future, accurately (Vlahogianni et al., 2015). However, it is suggested by the scholars that the application of cutting-edge Deep Neural Network algorithms can further enhance the detection mechanism leading to enhanced accuracy and planning (Vlahogianni et al., 2015).

The process of detection of parking space based on artificial intelligence is under development and constant improvement is expected. The work of Luca et al. (2016) revealed that the software application that is based on RESTful Java and Google Cloud Messaging technologies is being used to alert the visitors about the available parking spaces. In the latest scenario, as discussed by Paidi et al. (2018), the application of magnetometers, ultrasonic sensors, and also machine vision has been helpful in the case of closed parking lots leading to successful detection of free parking spaces. Based on the studies conducted by Wang et al. (2014), the automatic parking of vehicles is a possibility and the extension of same process i.e. visual perception, ultrasonic sensors, and radar technology can be done to achieve detection of free parking spaces. In addition to this, True (2015) has suggested the integration of static images-based systems, which are an extension to exit and entry sensors that help in updating the number of available parking spaces. The identified methods of parking space detection suggest that the field of study is important and there are various parameters to be understood while developing new technology in the industry.

## Challenges and Limitations in Implementation of Smart Parking Space Detection System

Though there are some very impressive aspects of parking space detection that have been brought forth and applied, yet there are several challenges faced by the industry to make the parking space detection more efficient and convenient. The study conducted by Al-Kharusi and Al-Bahadly (2014) revealed that the intelligent system for parking processes the information through the application of sensor camera that develops rounded image of the parking lot. The understanding of the vacant parking spaces is drawn through the images processed by the camera, which is categorised as the machine learning technique. However, the application of this process has its own limitations, as it is a single layer-based plan that is bound to lose its efficiency in case of congestion and higher frequency entries and exists. In addition to this, as explained by Thonduri (2018), the current car parking space detection system cannot focus on the updated vacancy in the parking space, which limits the ability of the consumers to understand the real-time situation. Hence, the occurrence of arguments among different car drivers and also the car parking executives is common. The limitations of the systems associated with car parking space detection affect the time as well as the mood of the citizens and car drivers.

Yet another major challenge is the inability of the applied technology to predict the parking spaces in the real time. The works by Caicedo, Blazquez, and Miranda (2012) defined the correlation between the traffic and the poor management of the parking spaces. They also suggested that the reservation of parking spaces leads to futile queues, as almost 50% of the times the parties that reserve the place do not appear (Caicedo, Blazquez, and Miranda, 2012). Hence, the system of reservation is an obstacle that can impact the overall functioning of the effective parking mechanism. The system of sharing a fleet of cars across the specified stations is also observed and it is verified that the problems arise due to the number of available spaces and the number of customers coming up with their requests. Krumke et al. (2013) defined through their studies that the limitations of the car parking management systems are seen in the ability of the management to handle consumer inflow in proportion to the available spaces. The current systems and technologies are not feasible to cope up with the real-time detection of different aspects of parking space detection. Thus, there is a need to further modify and accelerate the measures that can ascertain the development of such methodologies that there are no long waiting queues, and neither are there time lags in information collection.

## Vision-Based object Detection with Convolutional Neural Networks

Due to the immense of recommendation of vision-based detection using Convolutional Neural Networks (CNNs) in the literature, some research went into the implementation their implementation and comparison between its current variations.

Currently CNN do much more than classifying an image based on the predominant object in it. These models can be trained to detect different objects in an image, classify and segment them. However, some of the most successful object detection approaches are extensions of object classification models. As an example of an object detector, we could have a window of a fixed size taken from all positions of the input image and feed it to an image classifier (Qu, Jin and Feng, 2013). Then, the classifier predicts if there is an object in the window or if it’s the background. However, there is a problem with this method. It must process thousands of object candidates, which makes it 9 Background and Literature Review extremely slow. One solution to this problem would be to run a scanning detector instead of a classifier. First, we scan the image for object candidates with a generic detector and then run a classification algorithm on the object candidates (Alexe, Deselaers and Ferrari, 2012). Nonetheless, this solution also presents a few problems. Objects can vary in size and can have a different aspect ratio, e.g. a person in a standing position is tall and cars viewed from the side are wider, we would have to scan with windows of many shapes. There are various methods for object detection in the literature and each one deals with these problems differently. R-CNN (Girshick et al, 2014) uses an algorithm called selective search (Uijlings et al., 2013) to scan the input image for possible objects, which generates approximately around 2000 region proposals. It then runs a CNN on these regions to extract features. These features are fed into a SVM to classify the region and to a linear regressor to better adjust the bounding box (if the object exists). Basically, this approach turns an object detection into a classification problem. This method is very intuitive but also very slow. One of the reasons R-CNN was slow is that runs the 2000 object proposals on the CNN. The authors in (He et al., 2015), improved on this solution by introducing the SPP Network. This network uses a CNN to extract the feature maps from the entire input image only once, and it uses these feature maps to extract the features corresponding to the region’s proposals generated by the selective search algorithm in the input image. This is done using a SPP to pool features in that section of the feature maps in the last convolutional layer. From the extracted features it generates its fix-length representations which are used to train the object classifier and the bounding box regressors. The authors in (Ren et al., 2017), proposed Fast R-CNN that is very similar to the SPP-Net, but they replaced the SPP layer with a "region of interest (ROI) pooling" layer (single-level SPP). Another difference in this network is that it uses a single softmax layer to output the class probabilities directly instead of training several SVM’s to classify each object class. In this way we have just one CNN to train instead of one network and several SVM’s. The performance of Fast R-CNN improved in terms of speed but there is a bottleneck still remaining which is the selective search algorithm to generate the region proposals. Faster R-CNN (Ren et al., 2017), replaced the selective search algorithm used by Fast R-CNN with a very small CNN called RPN to generate ROI’s. To handle the different size of objects and aspect ratio, it introduces the concept of anchor boxes. The network looks at each location in the last feature map and considers three bounding boxes and three aspect ratios. In total, we have nine boxes on which the RPN predicts the probability of being an object or background. The job of the RPN is to propose object regions and does not classify any objects. If an anchor box has an “objectness” score above a certain threshold, it gets forwarded as a region proposal. The region proposals are then fed into what is basically a Fast R-CNN. Altogether Faster R-CNN achieves better speeds and state-of-the-art accuracy. The authors in (Guo et al., 2017) proposed a Region-based Fully Convolutional Network (R-FCN) that share all the convolutional computations in the entire image. The authors propose position-sensitive score maps to address a dilemma between translation-invariance in image classification and translation-variance in object detection. Each of these scores’ maps represent one relative 10 Background and Literature Review section of an object (e.g. top-left corner of an image of a car). The authors in (Farhadi et al 2015) proposed You Only Look Once (YOLO) which was a new approach to object detection. While previous works repurposes classifiers to perform object detection, YOLO frames detection as a simple regression problem to spatially separate bounding boxes and associated classes’ probabilities. It takes an image as input and predicts the bounding boxes and their class probabilities in one evaluation. YOLO divides each image into an SXS grid cell. Each grid cell predicts B bounding boxes, each bounding box has one box confidence score (objectness) and the box coordinates, and it predicts C conditional classes’ probabilities. The key difference of YOLO is that it only looks once at the images what makes it extremely fast and can run detections in videos in real-time.

## Summary

Through the review of the literature it is concluded that parking is a major problem in the metro cities that leads to the disruption and causes chaos. The traffic system is also affected due to the lack of parking management while at the same time the citizens suffer from time wastage. A number of systems have been formulated and are in practice for determination of vacancy in parking spaces. Most commonly used systems include multiple image-based identification, computer-based detection, frequency-based conclusion of parking spaces and their vacancy possibilities. However, numerous limitations exist in the existing technology, for instance, single-layer detection through camera recordings and images limits the ability of the machine-based learning to give on the spot and current time status. Thus, there are several new technological integrations suggested for the development of systems that can lead the developers to have a more compact and efficient parking detection system. The most promising of the suggested systems inclusive of RFID, WSN, machine vision, etc. is the integration of vision-based algorithms. These algorithms allow specific data identification and regulation through the system input, which is why the same is being studied in detail further.

# Requirement Specification

## Introduction

Following the literature review, the nature of this project was reviewed and a requirements specification table was produced (see [Appendix 11.1](#_Requirements_Specifications)) based on use cases of the application. The requirements created targeted functional and non-functional (e.g. accessibility, extensibility, operational etc...) aspects. The table was segmented into the different parts, to cover all parts of the system in a clear manner, with each segment having; list of requirements, type and priority. The MoSCoW prioritisation rules were followed for the produced requirements (Waters, 2009). Additionally, every requirement has a small description of what is intended to be achieved after its completion.

The following commentary will only include specific requirements for its complexity, inter-dependability or Significance.

## Commentary

### Overall characteristics

Due to the nature of the proposed system, a number of processes have to be performed at the same time, i.e. the application, has to run the interface and the detector in the same time. As a result, a number of requirements were added to account:

* Error prevention ([11.1.1](#_General) #4 and #6), which is implemented by limiting the functionality in one module depending on the state of another. This is vital to ensure the minimisation of system failures in cases where the user performs invalid actions.
* Multi-threaded and thread-safe ([11.1.1](#_General) #3), the proposed system has UI and a detection module that will be running at the same time without affecting the performance of each other. Furthermore, the output of the detection module will update the displayed information on the UI, which has to be done in a thread-safe manner to avoid system failure.

### YOLO Prediction framework

YOLOv3 neural network prediction model was chosen by the writer to be used in the proposed system. As such, throughout the specification tables (All [11.1.4](#_Training_a_model) and [11.1.5](#_System_Detection) #1) there are requirements specific to the fact that system will implement the said algorithm. The system will allow of the detection of multiple vehicles types which will prove the system’s ability to differentiate between varieties of vehicle types.

The vehicle detection and classification tasks are the most important in this framework, as the applications for vehicle detection relies upon fast and accurate vehicle detection capabilities. As stated before in the literature review in the last years CNN have shown great advantages in object detection and classification. For this task in our framework, we used YOLOv3 (Redmon and Farhadi, 2018) a state-of-the-art, real-time object detection system. This model is extremely fast and accurate and 17 Methodological Approach has several advantages over classifier-based systems such as Faster R-CNN. At inference time this model looks at the whole image, so its predictions are informed by the global context in the image. For us to be able to use YOLO in the video processing module we needed a deep learning neural network framework to implement the model. After some research in some of the frameworks available, such as TensorFlow (Abadi et al, 2016), Caffe (Jia et al, 2014) and MXNet (Chen et al, 2015), we found some implementations of the YOLOv3 model that were publicly available. After running some tests to find which of the implementations delivered the best performance, in terms of fps, we chose to use the Darknet framework (Redmon, 2013). Darknet is a neural network framework written in C and Compute Unified Device Architecture (CUDA) by the authors of the YOLO model. Also, this version of YOLO (v3) is very recent and that could explain why the other implementations did not perform as well (considerable slower) as the Darknet framework. YOLOv3 uses Darknet-53 a new 53-layer CNN for feature extraction. To run the object detection model, we needed some weights for the feature extraction layers (convolutional layers). We used the weights of a pre-trained model trained on the Imagenet dataset with 1000 object classes and then fine-tuned on the COCO dataset (Lin et al, 2014) with 80 object classes. One of the advantages of YOLO is that it makes predictions with a single network evaluation. YOLO divides the input image into a SXS grid. Each grid cell predicts only one object, e.g. in Figure 3.3 the green cells try to predict the vehicles whose centre (orange dot) fall within the same. To locate the objects, each cell predicts a fixed number B of bounding boxes and each box includes a confidence score, four parameters for the bounding box and C classes’ probabilities but detects only one object regardless of the number of boxes B (Figure 3.4). YOLOv3 uses multi-label classification meaning that same objects can have more than one label e.g. the model may label a pick-up van with two labels (car and truck). As we are using a pre-trained model, we made an adaptation of the non-maximal suppression used in YOLOv2, so we remove the duplicated labels and bounding boxes with the lowest confidence score, as we don’t want to have more than one label per vehicle detected. On the video processing module once an image is forwarded to the vehicle detection and classification model, we will obtain the data of each detection as a result and not an annotated image. The data obtained from the model is the x and y coordinates of the top left corner of the bounding box, the width and the height of the bounding box, a class label and a confidence score for the class of each detection. YOLOv3 is a state-of-the-art object detector. It’s fast and accurate. It has been improved to detect smaller objects and is able to predict bounding boxes at three different scales using a similar concept to feature pyramid networks. Even using a pre-trained model on generic objects proved to be an accurate vehicle detector in images of traffic scenes

The implementation of YOLOv3 was decided based on multiple aspects:

* A pre-trained model can be used as a base for the new model. The main advantage is that, in the training phase the model already knows how to pick essential features in the training data. This allows to produce relatively, highly accurate model with relatively small number of data (Nurminen et al., 2019).
* Although accuracy-wise YOLOv3 does not offer any advantages compared to its competitors (SSD and R-FCN), it offers a faster prediction in real-time (Redmon and Farhadi 2018). This was an important aspect of the proposed system as Parking space occupancy change frequently, having a fast prediction rate allows for more accurate and current information.

### Performance and Redeployment

Although the proposed system aims to be lightweight and easy to run and re-deploy. As with any real-time processing system it requires certain amount of power to run smoothly ([11.2.5](#_System_Detection) #4 and #5). The proposed system targets CUDA development techniques to make use of extra performance provided by NVIDIA GPUs. However, to widen the compatibility range and offer better extensibility the program will also investigate supporting OpenCL included in all ARM and AMD central processing units (CPUs).

### Storage and Privacy

Proposed system’s logging functionality, [11.2.1](#_General) #10, will store information a completely anonymised manner to avoid any privacy liability, to adhere to the ethical approval of this project. Moreover, information will only be stored as changes in the created regions of interest appear.

### User Interface (11.2.2 & 11.2.6)

The implementation of flashy UI is not of vital importance to this project. Nonetheless, the UI maintaining validation, control of functionality and clarity of information displayed to the user will not be overlooked. Multiple requirements were added to ensure basic functionality to enable the demonstration of the proposed system. The UI will take human input to in order to control the system’s functionality.

## Summary

Produced requirement specifications was limited, as many of the tasks are time consuming to complete, i.e. data collection, data annotation and model training. However, the due to the time constraint of this project. However, the proposed system will allow for better extensibility to allow adding new feature in the future, [11.2.1](#_General) #8. Depending on the priority of the requirement some will be included before other and some will be ignored based on time constraints.

# System modelling & design

## Introduction

Before starting the development of the proposed system, a set of design documentations were produced for all system components. This will help give a better overview of what to expect during the development phase of the project. The design documents produced includes, class diagram, data dictionary diagram, activity diagrams and interface wireframes. The design documents were updated throughout the iteration of system development phase to ensure they contain final and latest updates and features.

The design documents are provided in full in the report’s appendices (see Appendix [11.3](#_Design_Documentation)).

## Class Diagram

The class diagram is produced with the intention of developing the system using Object-Oriented Programming (OOP). As a result, it was deemed appropriate to develop the class diagram using Unified Modelling Language (UML). Loose coupling and high cohesion were highly maintained to allow for better extensibility and easier code understanding (Clear, 1997). As shown in the diagram, the system consists of two main sub-systems; the User Interface (appUI: Form1) and the Detection Module (detectionModule : DetectionModule).

To show how the application functions and the properties, methods and data Types it needs to function. Additionally, structure of objects was carefully designed to not store unhelpful information which will minimise the memory consumed by the system. Noticing the lines between classes in the diagram represent association and are annotated by the field/property it is causing the association. Which shows that objects are very loose, and one can be changed without affecting more than one class.

## Activity Diagram

The activity diagram (see Appendix [11.3.2](#_Activity_Diagram)) details the main three activities the proposed system achieves, initialisation, add location and real-time detection. These activities were based upon the functional requirements of the proposed system. This displays a broken-down structure of system starting point, system actions, user actions and system ending point. It also details about each activity, certain conditions which must be checked, such as, whether there are any installed cameras, to ensure they was no disconnection or failure at any stage. This ensure the minimisation of system failures due to null references and unmanaged objects exceptions.

## Dictionary and Data Transfer Objects (DTOs)

To replace the need for using a local database. Multiple Data transfer objects and dictionaries were implemented to help move data from one part to another in the system. The designed objects can also be viewed in the product documentation and the produced class diagram (see Appendix [11.3.1](#_Class_Diagram)).

Additionally, a Dictionary derived class (BaseRepository<Tk,Tv>) was created with added features to help easily save local copies of the recorded information by the proposed system. This was designed and implemented at a later stage of this project, as the logging recorded information was a lower priority requirement. The dictionary is initialised by calling the constructor and providing a file name. Which is used at later stage to append all new information into the file or creating a new file if one does not exist. Appended information is saved in Json file format to allow for easy serialisation and deserialization, in turn, easy reuse of the data.

## YOLOv3 network Structure

To have the ability to train the prediction model, yolo requires a neural network configuration file. Although this file was continuously altered during the development of the project to increase the accuracy of the trained model, the number of classes it targeted have stayed the same in accordance with the requirement specifications (3; car, truck and motorbike).

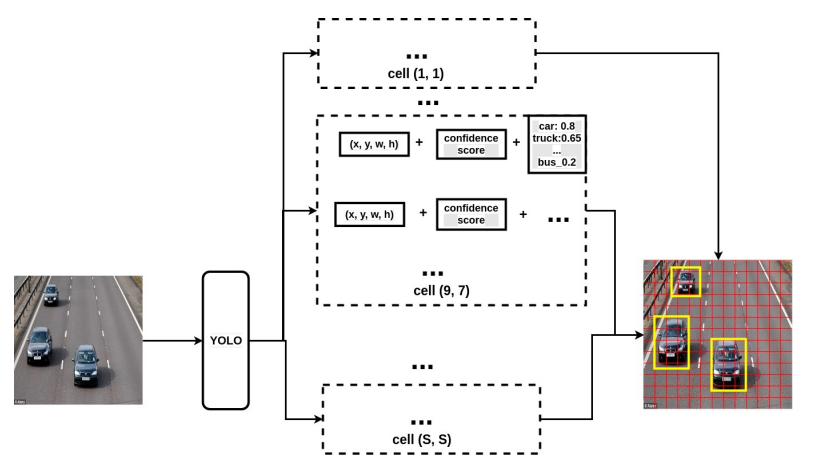


Figure YOLO system structure

### Configuration file

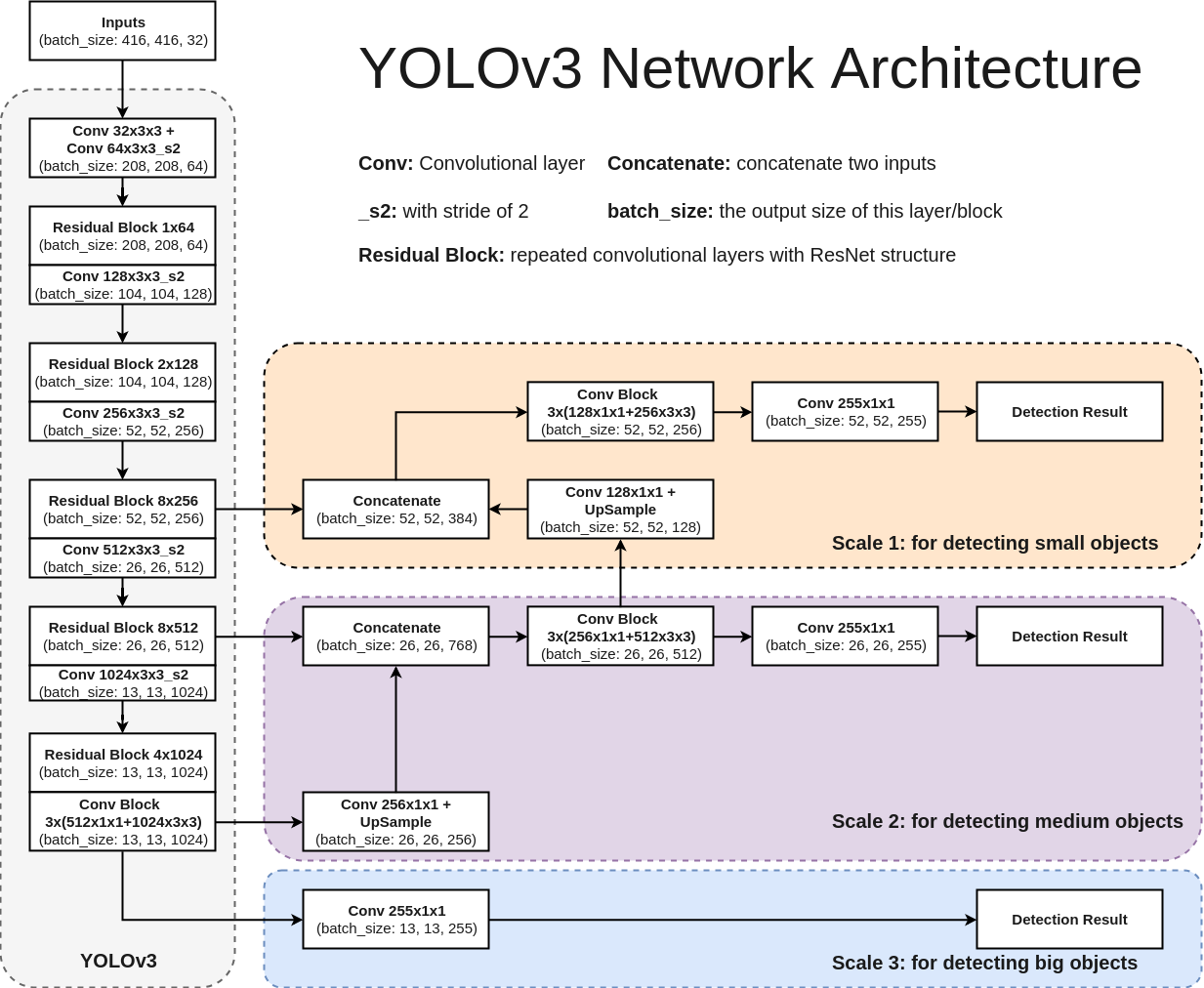


Figure : YOLOv3 structure (CyberAILab, 2018)

The structure of the network follows the default the Yolov3 structure discussed in the literature review.

As a result, the base configuration file was used, however, some changes were made to the configuration file before training the model:

* Changing the number of filters used by the last convolutional layer before the each of the three YOLO layers. Following this equation, filters = (classes + coordinates + 1) \*[number of mask]. Where the number of classes = 3; coordinates is the number of coordinates to be predicted for each object = 4; and the mask is index of the prediction layer, in this case = 3;
* Batch size and subdivisions were changed to a minimum to lower the load on the training workstation.
* Number of epochs was set to 6000.

Further amendments were also introduced to increase the accuracy of the model, which will be discussed the development chapter

### Training dataset

The first stage at this phase was to find an appropriate image dataset for the purpose of training the prediction model. Open Images Dataset v4, was chosen as an appropriate dataset due to the number of images it contains for each of the 3 target classes, the different conditions images were taken in and their resolution. 300 images were downloaded for each class and further segmented into 70% training, 20% testing and 10% validation to improve the accuracy of the model without biasing testing with images it was trained on.

The downloaded images were then annotated following YOLO’s syntax by using an appropriate annotation tool (VOTT).

## User Interface

Application was designed with a simple interface to display the processed information. As requirements were sorted through, it was decided that the interface will be designed to follow the Heuristic evaluation techniques to improve its aesthetics and usability. As shown in Appendix [11.3.4](#_Interface_wireframes), a dummy interface was create to estimate the look and feel of the different parts of the interface.

## Summary

The development of the system will strictly follow the outlined designs, although, if due to unforeseen constraints, designs will be amended and reviewed before performing any of the implementation.

# System development

## Introduction

After the completion of the design specification documents the system is starting the development phase. This phase will focus on the main functional and non-functional requirements and add-in more functionality as per the time constraint.

## Main program

This is the starting point of the system. The idea is that this sub-system will call the interface first once a location profile has been saved it will call the detection module based on the information received from the interface. This part could easily have been omitted; however, it was decided to be implemented as it allows for a much higher extensibility and modularity of the application which couldn’t be implemented due to the time constraint of this project. For example,

* Amendment on this part of the system can be allow it to be installed as a service running automatically as windows starts up.
* Can be used to hook up the detection module to a different or multiple interface.
* Allow for the addition of web APIs. This could easily be implemented by starting multiple service to be then called by the web project using simple http requests.

One aspect of note, is that although normally tasks started in different threads are started using the following syntax, Task.Run((() => "Do Work"));. On the other hand, starting an windows form application uses the [STAThread] keyword to inform the compiler to run GUI-based task.

[STAThread]

private static void RunForm()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

appUI = new Form1();

appUI.StartDeteBtn.Click += StartDetection;

appUI.StopDeteBtn.Click += StopDetection;

Application.Run(appUI);

}

## Detection Module

This is the main component of the developed system. Due to the complexity and length of this sub-system it was divided into further smaller system to allow for an easier and less coupled implementation. The smaller system is, video input and output, detector training, introducing Regions of Interest (ROI), and detection module integration.

### Video input and output

The first stage aimed simply to retrieve video input from workstation cameras and display it as an output to later be used to inspect how the system works.

This section took into considerations having multiple cameras running at once, so a locking mechanism was needed to stop an input being accessed more than once. As a result, (Key, Value) based dictionary was implemented were the key is the index of the camera and the value is its information. The system goes through the list of keys and initialises the capture objects using a for-loop making sure no indices have been entered more than once.

var keys = LocsRepo.GetKeys();

foreach (var key in keys) {

if (LocsRepo.GetById(key, out Location location))

{

capturesDict.Add(new Dtos.CapturingDevice(key,

location.Cameras.Device\_ID));

}

}

Once the call is made to the CapturingDevice object’s constructor with the relevant camera number, an instance is created containing the camera object, an empty frame that matches the width and height of the camera input and a viewer. The viewer is used to display the output of the video input coming from the camera object. The main two complications were the Camera input and viewer objects,

* Camera Object: the camera object uses OpenCV’s dll to run, which was approved by their development licence. However, OpenCV is written C/C++, which was quite problematic as this proposed system is developed in C#. As such, a wrapper function was created to allow the use of such dll, using C# environment.

[DllImport(ExternLibrary, CallingConvention = CvInvoke.CvCallingConvention)]

internal static extern IntPtr cveVideoCaptureCreateFromDevice(int index);

* Viewer Object: the application required a second interface to display the output of the Camera object. Additionally, further functionality, such as pointer location, viewer’s width and height and manipulation methods were needed to improve the testing and development of the application. To that effect, another form was created containing an Imagebox capable of taking images and displaying them when needed was implemented, as seen in Figure 2.

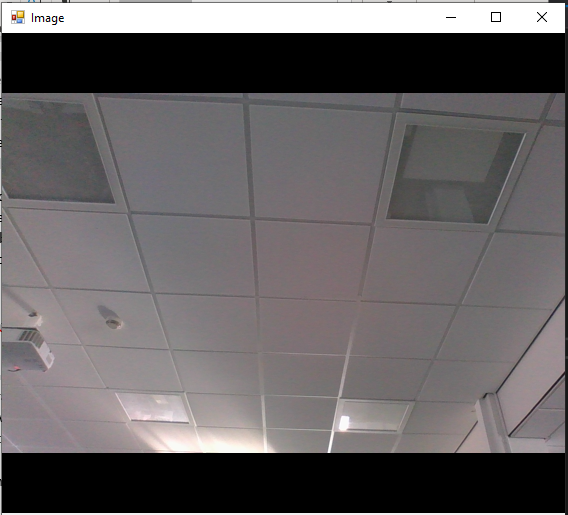


Figure Viewer Form

* The viewer was further advanced to show a child window displaying the properties of the viewer which will be vital to the development of the proposed system.

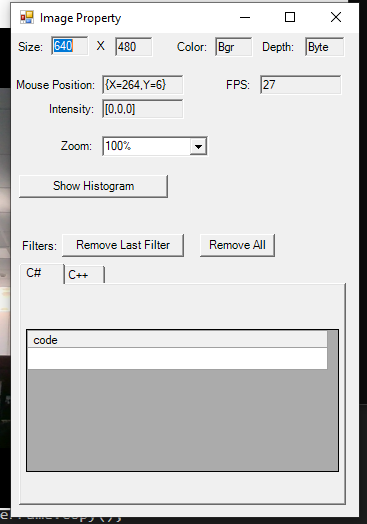


Figure Viewer further advancements

### Detector training and Viewer integration

The training of the YOLOv3 model was performed by cloning AlexyAB’s fork of Darknet, (<https://github.com/AlexeyAB/darknet>) and installing its requirements and pre-requisite drivers. Darknet takes in,

* YOLO configuration file, contains the structure of the network and all its modifications. The config file used for the system implemented modifications that were not in the base configuration files. Most important ones are,
  + The flag “random=1” was set to 1. This will increase precision by training YOLO for different resolutions, as it takes in different ratios of the same image to ensure accuracy on randomly selected resolutions.
  + The width and height of the images were adjusted to 320px to ensure faster training time and prediction speed.
  + Imbed cluster during the training of the network by changing indexes of anchors by amending the “masks=” keyword for each [yolo]-layer, so that 1st-[yolo]-layer has anchors larger than 60x60, 2nd larger than 30x30, 3rd remaining.
* classes.names, this file includes the names of classes to be detected; in this case, truck, car and motorbike
* classes.data, this file includes all the information the detector wants and is the one to be added in the training command of Darknet. It consists of following structure. Where the first line states the number of classes, the second is text file that includes all the paths for training images; the third, is contains the path for all testing images; the forth one is the names of the classes; and the last line indicate the directory where backup models will be save every 1000 epochs.

classes= 3

train = data/train.txt

valid = data/test.txt

names = data/obj.names

backup = backup/

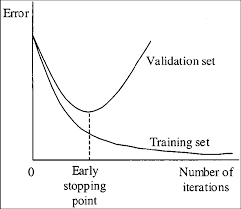
During the training phase of the model, it was improtant to keep an eye on the precision of the performed predictions. This is because training the model for too long introduces overfitting and decreases accuracy. As shown in Figure 4, overfitting is where the model’s predictions start to apply only on the training data. Essentially, the model starts memorising the training data and not anything else.

Figure Stopping training (Scardapane and Di Lorenzo, 2017)

After training was completed the model performed best on cars achieving 87% precision compared to 55% on trucks and 58% on motorbikes. The average is 15% higher than the trained model without the configuration changes. Although the percentages are still relatively low, this might be due to the limited number of training samples provided.

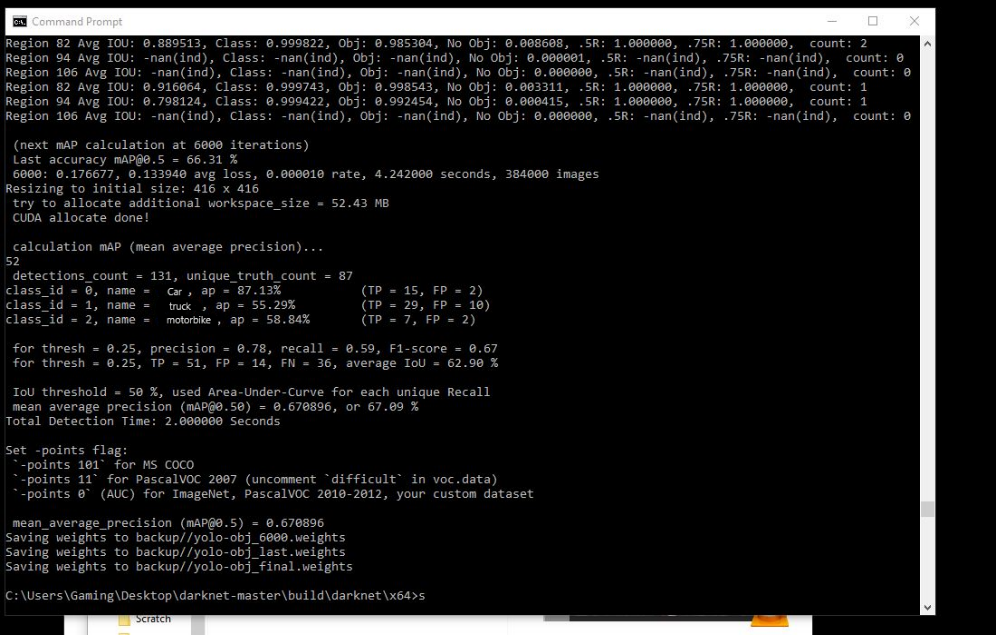


Figure Training final output

Furthermore, the application reached a loss < 0.5 and mean average precision of 73% at 4800 epochs, which is the model used for the final output of the application.

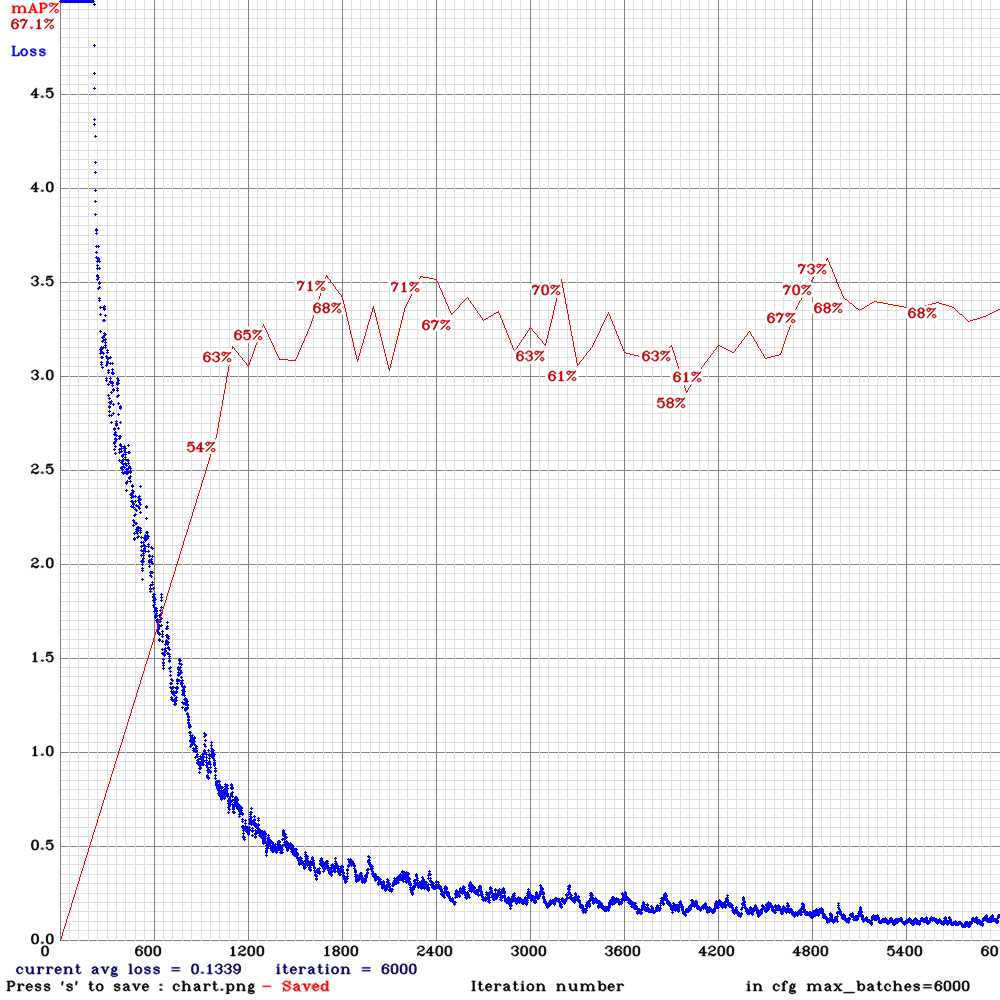


Figure Training Loss and Mean Average Precision

Once the model was finished it was tested against a simple picture of a parking lot and produced this output (shown in Figure 7).

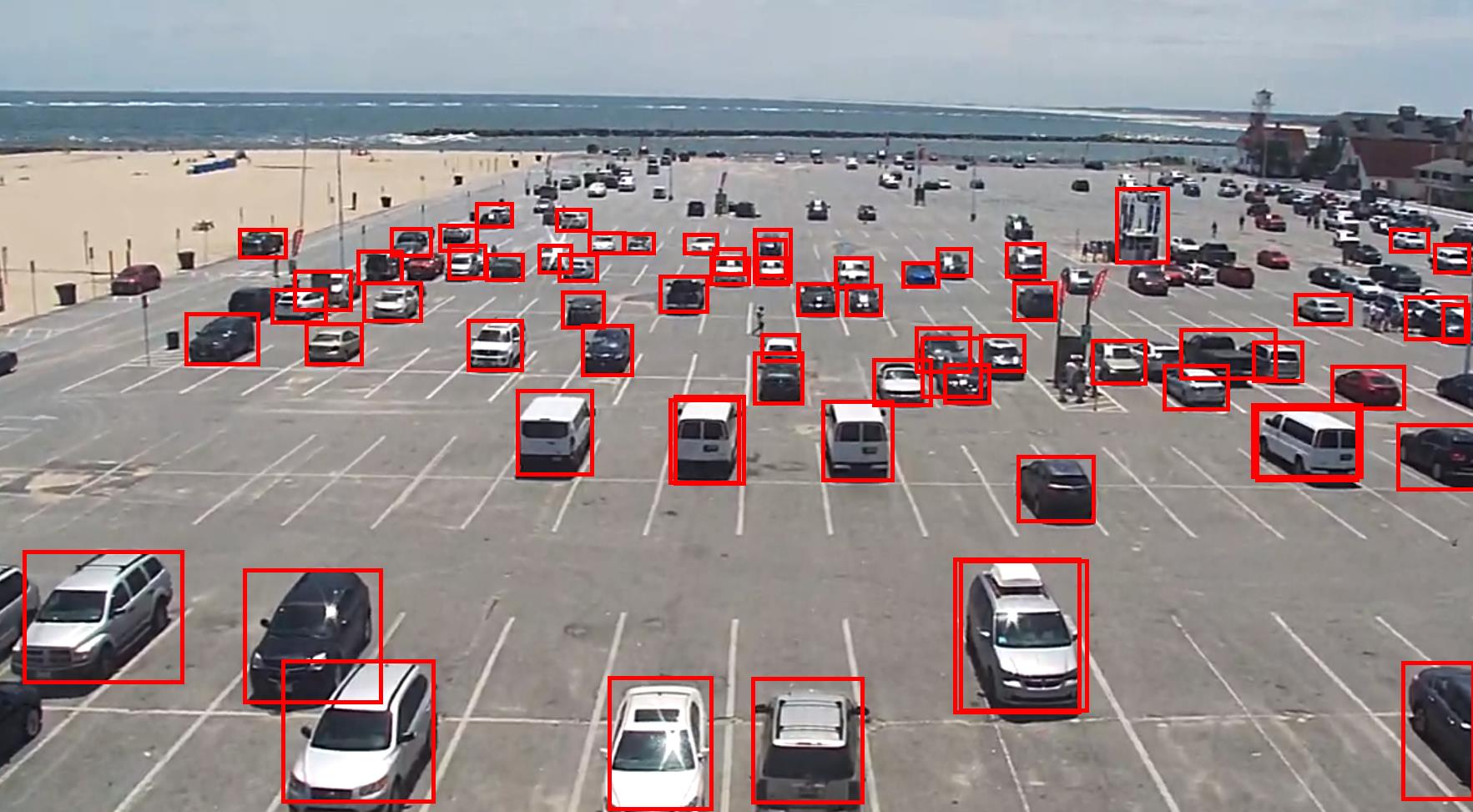


Figure Finished model output

Once the application was proven to be working, it was time to integrate it with viewer made in the previous step. This was achieved relatively easy once YOLOv3 wrapper was introduced to the application.

YOLOv3 wrapper works simply by running this command on the target frame converted to array of bytes and return a list of predictions and their confidence level.

//run the converted image through the detector

var yoloObjects = yoloWrapper.Detect(Frame);

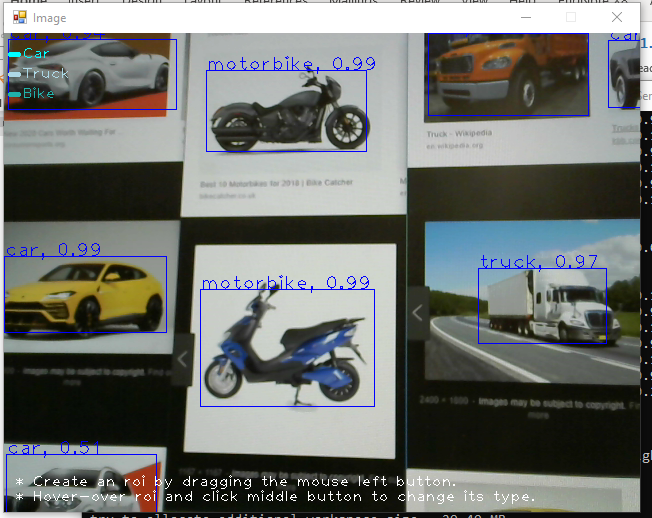


Figure Live prediction

### Introducing ROIs

Regions of interest is the approach implemented to detect the occupation and violation of parking spaces. Ultimately, the goal is to allow the user to create rectangular regions of interest, which are watched by the system and their state changes if prediction’s location intersects with them.

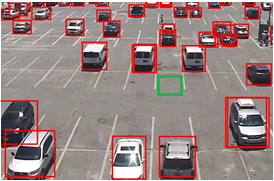


Figure Shows two different possible ROI location with one occupied and the other is not

This was implemented by capturing user input and recording the mouse location of the press and the release location. The first gives, X and Y of the top-left rectangle corner; the second returns the difference between the first and second point to output the width and height.

There was one main point of concern in this section,

1. Although the press and release input return two points that can be used as top-left and bottom-right coordinates. The user can make the press at point (50,50) and the release at (0,0) which would return negative values for width and height. As a solution, this piece of code was implemented guarding the release button coordinates. Where roiSelection is the X and Y of the press action. Essentially this code, replaces the x/y for the press action with release’s x/y to ensure the width and height are always positive. At this point the developed properties for the viewer were very vital as the x and Y coordinates can be viewed with ease.

var copyX = e.X;

var copyY = e.Y;

if (roiSelection.X > e.X)

{

copyX = roiSelection.X;

roiSelection.X = e.X;

}

if (roiSelection.Y > e.Y)

{

copyY = roiSelection.Y;

roiSelection.Y = e.Y;

}

roiSelection.Width = copyX - roiSelection.X;

roiSelection.Height = copyY - roiSelection.Y;

### Integration

During the process of development for the detection module was built in steps, which made it very easy to integrate them together. This was a relatively simple step with no complications as each sub-part was dependant on the other and was required to work before moving to the next one.

Unfortunately, due to the limited number of training images, quality of annotation and power of the training workstation, the detection of objects was not as accurate as intended. Due to the inaccuracy of the model, the bounding box of predictions covers not only the object but also surrounding area. As a result, objects might affect the state of an ROI even if they are not within it, as shown in Figure 10. As a solution, the predicted bounding box was cropped by 20% of its size and ran through the prediction algorithm again, which produces another bounding box to be used for finer detection.

Figure Advanced bounding box



## User Interface

This part was left till the very end of the development phase due to its relatively low importance. Nonetheless, the implemented interface’s aesthetics were optimal considering the produced wireframes in the design phase. Initially, this part of the system was set to have low importance, however, the development phase had more time than planned, which enhanced the looks of the designed interface.

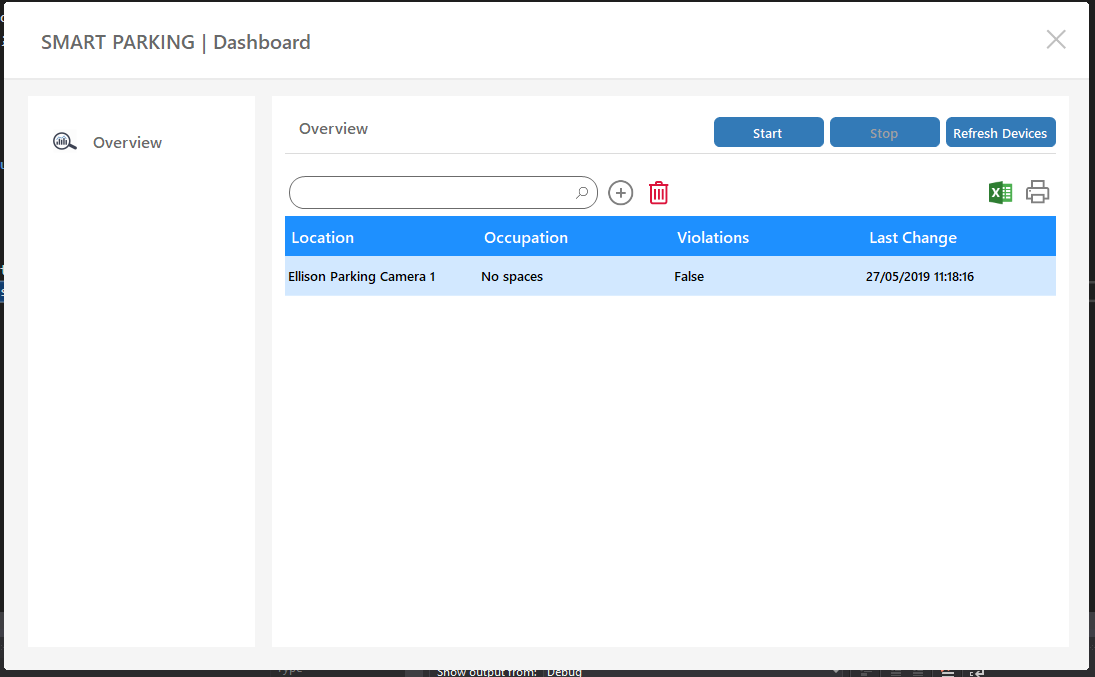


Figure System UI, Dashboard

The interface was designed using .NET windows forms and Bunifu wapper library was used purely to improve the looks of the interface. Nonetheless, the UI was developed considering various usability requirements to enhance user experience.

* Minimalistic design was implemented to ensure ease of use. Some action buttons were replaced with icons to allow for better aesthetics.
* 3-click rule was also taking into account that no action should take more than three clicks to be performed.
* UI elements: when developing the UI the contrast between the size of text, its colour and colour of the background were carefully chosen to allow for better usability. This application is designed to be used in a working environment where there’s appropriate lighting. As such, light colours background with dark text were chosen to enhance the readability of elements on the screen.

The developed interface strictly followed the design specifications, implementing approaches to limit human error by disabling buttons, displaying appropriate information and error messages, as well as minimalistic design to make the application as clear and easy to use as possible.

One main complexity was the automatic update functionality of the interface, as the two processors are working on different working threads as well as being of different types, information processor working on a Task thread; the interface on a STAThread. This introduced a “cross thread operation exception” when attempting to update the interface due to new predictions in the detection module. As a result, the implementation of a delegate method was necessary to request the STATthread to perform the update without an intrusion from the working thread in the detection module. The implementation required multiple amendments on how information is processed in the UI part of the system.

On the detection module, each location calls this at the end of processing each frame,

private void UpdateUIRepo(CapturingDevice captureObj)

{

Program.appUI.Invoke((MethodInvoker)delegate { Program.appUI.UpdateRepository(captureObj); });

}

On the user interface,

public void UpdateRepository (Dtos.CapturingDevice locObj)

{

LocationsList.GetById(locObj.LocationID, out Location location);

var states = locObj.OccupationState();

location.TruckActive = states.Item2[0];

location.CarActive = states.Item2[1];

location.McActive = states.Item2[2];

location.ViolationActive = states.Item2[3];

location.OccupiedROIs = states.Item1[0];

location.TotalROIs = states.Item1[1];

location.Timestamp = DateTime.Now;

LocationsList.Add(locObj.LocationID, location);

this.overviewTab.PopulateDatatoDatagrid();

}

Following the update of information on the interface the local information logger checks the location object. Only if the object was changed since the last time it was logged will the system run the following method from the created repository object, which writes it into the local file chosen when initialising the LocationsList object. This is to reduce the size of the created file and saving only new information where a occupation has been changed.

LocationsList.WriteDictionaryToDisk();

## Summary

Overall, the developed application accomplishes most of the functional requirements in the produced requirements specification document. The structure of the system was highly maintained to avoid any unnecessary dependencies and ensure the application was loosely coupled to allow for future extensibility and updates. The trained prediction model was not as accurate as intended, however, due to the current model taking 40 hours for the training process, other approaches were implemented to improve the prediction accuracy. Further, the interface was developed was highly functional, smooth and aesthetically pleasing, which was given low priority in the design stage but was picked during the final stages of the development.

# System testing

## Introduction

Following the completion of implementation phase of the proposed parking occupation and violations system, a test plan based on the same structure as a Factory Acceptance Testing (FAT), was produced to measure out whether the functionality implemented was appropriately achieved.

## Test plan and Results

For the purpose of testing the system, a test plan was created to test each part of the system separately to confirm its functionality, followed by integration tests to ensure successful communication and functionality across the system as a whole. Additionally, the way the test plan is designed allows for quick identification of faulty items, for example, failed expected results can be identified as, (test 1, a, 4, 5); the first column is the test number, the second is the test performed and following columns are the failed expected results.

*The system has passed all the following tests.*

1. **Test Target:** User interface, Dashboard
2. Tests the UI’s controls on start-up if the system has no cameras attached. This test ensures that all controls are in their correct state when starting the system with no camera attached.

* **Tasks**
  + Disable/unplug all cameras.
  + Run application
  + Inspect user interface
* **Expected**
  1. “No video input devices …” Message box appeared.
  2. “Start” button is enabled.
  3. “Stop” is disabled.
  4. Refresh button is enabled.
  5. “Add” button is disabled.
  6. “Delete” button is disabled.
  7. Side “overview” tab is enabled

1. Tests the UI’s controls on start-up if the system has no cameras attached. Ensures controls are updated when refresh cam button is clicked.

* **Tasks**
  + Do not plug any input devices.
  + Click “refresh cameras” button
  + Inspect user interface
* **Expected**

1. **“No video input devices …” Message box appeared.**
2. “Start” button is enabled.
3. “Stop” is disabled.
4. Refresh button is enabled.
5. **“Add” button is disabled.**
6. **“Delete” button is disabled.**
7. Side “overview” tab is enabled
8. Tests the UI’s controls on start-up if the system has no cameras attached. Ensures controls are updated when refresh cam button is clicked.

* **Tasks**
  + Plug any video input devices.
  + Click “refresh cameras” button
  + Inspect user interface
* **Expected**

1. **“N video input devices found …” Message box appeared.**
2. “Start” button is enabled.
3. “Stop” is disabled.
4. Refresh button is enabled.
5. **“Add” button is enabled.**
6. **“Delete” button is enabled.**
7. Side “overview” tab is enabled
8. Tests the UI’s controls on healthy start-up if the system has cameras attached.

* **Tasks**
  + Restart the application.
  + Inspect user interface
* **Expected**

1. “Start” button is enabled.
2. “Stop” is disabled.
3. Refresh button is enabled.
4. **“Add” button is enabled.**
5. **“Delete” button is enabled.**
6. Side “overview” tab is enabled
7. Tests the UI’s controls on start-up. Start button Functionality with no location added

* **Tasks**
  + Click “Start” button
  + Inspect user interface
* **Expected**

1. **“No locations are added …” Message box appeared.**
2. “Start” button is enabled.
3. “Stop” is disabled.
4. Refresh button is enabled.
5. “Add” button is enabled.
6. “Delete” button is enabled.
7. Side “overview” tab is enabled
8. Tests the UI’s controls on start-up. Delete button Functionality with no location added

* **Tasks**
  + Click “Delete” icon button
  + Inspect user interface
* **Expected**

1. **“No locations are added …” Message box appeared.**
2. “Start” button is enabled.
3. “Stop” is disabled.
4. Refresh button is enabled.
5. “Add” button is enabled.
6. “Delete” button is enabled.
7. Side “overview” tab is enabled
8. **Test Target:** User interface, Location Form
9. Tests the Location form on start-up if the system’s cameras were detached. This test ensures that all controls are in their correct state when the system with no camera attached.

* **Tasks**
  + Disable/unplug all cameras.
  + Click “add” icon button
  + Inspect user interface
* **Expected**

1. **“No video input devices …” Message box appeared.**
2. “Start” button is enabled.
3. “Stop” is disabled.
4. Refresh button is enabled.
5. **“Add” button is disabled.**
6. **“Delete” button is disabled.**
7. Side “overview” tab is enabled
8. Tests the Location form on start-up if the system’s cameras were attached. This test ensures that all controls are in their correct state when the system with cameras attached.

* **Tasks**
  + Enable/plug all cameras.
  + Click “add” icon button
  + Inspect user interface
* **Expected**

1. **Add Location form appeared**
2. **Name field is empty**
3. **Description button empty**
4. **Drop-down list includes all cameras**
5. Tests the Location form on when adding a location. This test ensures that a location is added to the database.

* **Tasks**
  + Choose a name or leave blank.
  + Choose a description or leave blank
  + Choose a camera input device
  + Click the add button
* **Expected**

1. **Operation successful indicator appeared**
2. **Name field is reset**
3. **Description field is reset**
4. Tests the Location form on when adding a location. This test ensures that a location is added to the database.

* **Tasks**
  + Close the window
* **Expected**

1. **Form is closed**
2. **Dashboard is displayed**
3. **Added item is added to the dashboard list of locations**
4. **List items display “No spaces”, “false” and “<timestamp of creation>” on new addition**
5. **Test Target:** Detection, Camera input and output
6. Tests the detection module of the system. This test ensures that video input is received from the chosen camera devices.

* **Tasks**
  + Click the “Start” button.
  + Inspect the system
* **Expected**

1. **The system’s output the network being built for detection in the background.**
2. **“Start” button is disabled.**
3. **“Start” button’s text changed to “running”.**
4. “Stop” is enabled.
5. **Camera Viewer is displayed**
6. **Camera input is displayed**
7. **Test Target:** Detection, YOLOv3 detection and recognition
8. Tests the detection module of the system. This test ensures that video input is recognised by the trained model.

* **Tasks**
  + Get an image of a car, truck and motorbike (real view or on a computer screen).
  + Put the three images in the view of the camera
  + Inspect the Viewer
* **Expected**

1. **The system shows two boxes for each prediction, Blue and yellow, and a confidence level.**
2. **“Start” button is disabled.**
3. **“Start” button’s text changed to “running”.**
4. “Stop” is enabled.
5. **Camera Viewer is displayed**
6. Tests the detection module of the system. This test ensures that video input is recognised by the trained model even in different light conditions, resolution and orientation.

* **Tasks**
  + Get an image of a car, truck and motorbike (real view or on a computer screen).
  + Lower the brightness of the screen.
  + Rotate the image
  + Use lower resolution images
  + Use all of the above settings at once
  + Put the images in the view of the camera
  + Inspect the Viewer
* **Expected**

1. **The system shows two boxes for each prediction, Blue and yellow, and a lower confidence level.**
2. **“Start” button is disabled.**
3. **“Start” button’s text changed to “running”.**
4. “Stop” is enabled.
5. **Camera Viewer is displayed**
6. **Test Target:** Detection module, Viewer’s regions of interest
7. Tests the detection module of the system. This test ensures that video input viewer accepts user action to produce ROIs.

* **Tasks**
  + Right click using the mouse on the viewer
  + Click the last option “properties”
  + Inspect the Viewer
* **Expected**

1. **New child window appears**
2. **Viewer information is displayed**
3. Tests the detection module of the system. This test ensures that video input viewer accepts user action to produce ROIs.

* **Tasks**
  + Left click and drag the mouse at any place in the viewer’s window.
  + Release the button at the desired location
  + Inspect the Viewer
* **Expected**

1. **A Cyan coloured box appears**
2. **Unique ID is printed above the region of interest**
3. Tests the detection module of the system. This test ensures that video input viewer accepts user action to change ROIs type.

* **Tasks**
  + Press the middle button on the ROI
  + Inspect the viewer
* **Expected**

1. **The ROI changed colour to Blue, which indicate the type**
2. Tests the detection module of the system. This test ensures that video input viewer accepts user action to delete ROIs.

* **Tasks**
  + Press the control key + the left mouse button on the ROI
  + Inspect the viewer
* **Expected**

1. **ROI is deleted**
2. **Test Target:** Parking System, Integration
3. Tests the system as a whole. This test ensures that video input is displayed correctly on the viewer, and the viewer accepts user action to produce ROIs, the prediction is displayed accurately on the viewer, ROIs react to the prediction bounding box and that the alert is displayed on the dashboard for that location.

* **Tasks**
  + Create multiple ROIs each with different type.
  + Force the created ROI to intersect with the prediction bounding boxes
  + Inspect the viewer
* **Expected**

1. **ROIs have different colours matching the legend on the top left corner of the viewer**
2. **ROI’s colour change to orange as soon as it has intersected with a prediction.**
3. **After few seconds, its colour changes to red confirming the occupation.**
4. **Dashboard reflects the output of the viewer, by amending the occupation percentage, violations and latest timestamp.**
5. **Test Target:** Parking System, Logger
6. Tests the system logging functionality.

* **Tasks**
  + Open the system’s exe directory
  + Find the “LocationProfiles.json”
  + Open it using text editor
* **Expected**

1. **File exists.**
2. **Events are populated in the file with different timestamps.**

## Performance Testing

The system supports the use of NVIDIA CUDA drivers to allow the model more processing power, in turn, faster prediction speed. The system was tested on two different machines and four configurations, each with a different GPU.

The following table was produced by calculating the amount of time taken to process one frame and calculating the average for 1000 frames, using the simple piece of code,

//Initialise the Stopwatch

var stopWatch = new System.Diagnostics.Stopwatch();

//Start the stopwatch

stopWatch.Start();

//Feed the new frame array to the model

var yoloObjects = yoloObject.Detect(newframe);

//Stop the stopwatch

stopWatch.Stop();

//Add value to the list be calculated

benchmarkValues.Add(stopWatch.Elapsed.TotalMilliseconds);

|  |  |
| --- | --- |
| Graphical processing Unit (GPU) | Average Time per frame (ms) |
| Intel i5-4670K | 1506ms |
| Intel i7-7700HQ | 2238ms |
| NVIDIA GeForce GTX 1050 TI | 151ms |
| NVIDIA GeForce GTX 1070 | 30ms |

## Summary

Although, the overall functionality of the system was tested multiple times and improved after each run to allow for a smooth-running system, with minimal failures. The list of tests created was maintained during the development of the application to ensure all functionality and possible variations are tested.

# Evaluation

## Product Evaluation

The developed software product as part of this dissertation, SmartParking, and its deliverables meets most of the requirements set in the Terms of Reference (TOR, see Appendix [11.1](#_Requirements_Specifications)) and the design specifications in [Chapter 4](#_System_modelling_&). Moreover, some of the low priority requirements were included as part of the extra time provided for the development phase.

### Requirements evaluation

**General Requirements**

All requirement in this table were fully met.

**Dashboard Interface**

Requirement from 1-4 were fully met. Fifth and sixth requirements were not met. They were given a low priority and were deemed not important enough during the development phase.

**Video Input Manager**

All requirement in this table were fully met.

**Trained Model**

All requirement in this table were fully met.

**Detection system**

All requirement in this table were fully met.

**Detection interface**

All requirement in this table were fully met.

### Fitness for Purpose

The initial idea of the project was to use a pre-trained SVM model and library to perform detection and display the processed information on a web-based interface. After discussing this idea further with colleagues, the writer deemed it too elementary and not as challenging as desired, which inspired the idea of developing the same proposed system with few amendments.

1. More focus on the detection algorithm.
2. Have a local interface with more attention to occupations and violations in the location profile in the system
3. Increased extensibility to allow for a web-based interface as originally intended. By storing real-time information in a Json format which can easily be reserialised and used with a web API.

The implementation of a cutting-edge deep neural network algorithm was one of the main objectives of the new initiative, which appeared during the literature review process of current limitations of smart parking solutions. As such, the YOLOv3 framework was chosen and a model was trained from scratch. These changes were reflected upon throughout this report, as many design specification requirements had to be re-written after the adoption of YOLOv3 and before starting the development phase of the project.

Additionally, the system successfully identifies vehicles within parking spaces, and has the ability to differentiate different types of vehicles and raises alerts in the case of parking type violations, which is the main purpose of the developed system. However, this application could easily be extended to allow for more features and functionality.

One main goal in the original TOR, that is not applicable anymore due to the use of a local interface was that third-party stakeholders (i.e. drivers), will not be able to view the current state of the parking space.

### Build Quality

Build quality was given a lot of focus during the development of the system. A set of good coding practices were implemented as stated in the TOR, including KISS (Keep It Simple Stupid) and SOLID design principles. Additionally, DTOs were used which helped the functionality of the system greatly. These good practices allow for easy extensibility, code readability and code re-use.

Other approaches were also used to enhance build quality including:

* Semantic structure of the written code was formatted and organised appropriately.
* Code was written and commented throughout the system files to preserve readability.
* Different functionalities were kept in separate files and only included when needed.
* Thread-safe methods were used to eliminate cross-thread exceptions and failures.
* Global variables were used only when necessary to minimise dependability of code sections on others.

### Usability Evaluation

User interface issues will be evaluated using the Heuristics evaluation method (Fung et al., 2016). This framework was introduced by Jacob Nielsen in the 1990s to evaluate the user interface of applications, and the same has been used on products designed by Apple, Google, and Adobe. To conduct a comprehensive Heuristic evaluation of the system’s interface, the writer assumed the role of an experienced role to successfully critically evaluate the produced UI.

**The 9 evaluation steps include:**

* Visibility of system status- the application informs users of system operations within a reasonable time to provide user with ease.
* Match between the real world and the system- this application may have fallen short on this part, as some aspects have their inherent limitations, such as, ROIs can only support rectangular shaping, which is not practical unless the camera has a direct line to target parking space.
* User freedom and control- the application allows users to redo actions, such as, adding and deleting location profiles, and adding and deleting ROIs.
* Standards and consistency- there is no hierarchy on the font sizes and vital information that should occupy large space, on the home screen is allowed less space as compared to secondary information. Additionally, the button placements, colours, and shapes are not standardized.
* Prevention of errors- buttons and icons are appropriately named which will reduce the number of errors committed.
* Recognition rather than recall- only few commands were implemented which helps the user remember the workflow of the system.
* Flexibility and use efficiency- the app’s visual is disconnected from the user’s environment. The loading of camera input and neural network could be faster as it takes a few seconds.
* Aesthetic and minimalist design- minimalistic design was implemented, but lacks more features and colours
* Help users recognise, diagnose and recover from some errors- the application does provide with message boxes to help recover from errors users. Additionally, users it is not possible to change some of the location settings.

## Personal and Process Evaluation

Due to the nature of this project, including the technologies used and size, it is without a doubt, one of the hardest challenges the author has faced, and in turn, the most rewarding.

Knowledge gained from this project about machine learning and cutting-edge Deep Neural Network (DNN) algorithms, windows forms development and threading and concurrency, as a result of this project are considered invaluable to the author. At the start of this project, the author had very limited knowledge about the field of machine learning and DNN. As a result, this knowledge has provided the author with great appreciation to the advances made in this fields and its current limitations.

The project management aspect enabled the author to further their knowledge in the subject areas by allowing the freedom to use whatever means to implement the developed system, which the author did not posse much familiarity in. DNN have provided more accurate solutions that surpassed the threshold of acceptability in many application using speech recognition and computer vision, which makes the author very confident in his ability to keep up with contemporary research in this field.

The particular difficulty this project introduces was the ability to manage time properly. A project plan and timeline were produced as part of the TOR, but unfortunately the author did not adhere to the stated deadlines. While the having the freedom to plan, design and execute one’s own project and dictate the direction of work been enjoyable, the open-endedness aspect of the project has been a blessing and a curse.

Nonetheless, these issues highlighted the need for improvement in project management skills including action planning and initiative, as well as time management skills and accountability.

The final and key advantage of this project was that it has combined knowledge and skills learnt over the past few years in the author’s academic study; in particular, system analysis, software design and development, and programming. Ultimately this project has been the output of the knowledge accumulated over the last 4 years of this course.

# Conclusion and Further Work

## Conclusion

The produced requirements specifications earlier in this project outlined all essential, desirable and cosmetic requirements for the proposed system which was structured, designed and developed base on. Each of the project phases aimed to encompass as much detail as possible to ensure the quality and robustness of the produced project deliverables. Additionally, the produced documentation has provided a basis and plan for the implementation process of the system which was rigorously tested at a later stage.

The evaluation process covered aspects unrelated to the product; including how the author felt and performed during the project’s lifespan. It also aimed to assess the finished product’s build quality as well as its fitness for purpose. Moreover, it gave a good reflection of what went wrong and right during, and the author’s strengths and weaknesses which is vital for self-development.

In conclusion, the aims and objectives set at the beginning of the project were met and the use of YOLOv3 as a detection algorithm was proven to be very accurate and efficient even with relatively small number of training samples.

## Recommendation for Further Work

### Use of Smart Parking Sensors

The process of parking a vehicle in traffic dense environments is a struggle and results in time loss along with the congestion and resultant environmental pollution. In the studies conducted by Paidi et al. (2018) it is concluded that the smart parking sensors and technologies facilitate the drivers through the guidance to free parking spaces. The process of smart integrations can be helpful in bringing about an improvement in the parking process. In their study, the authors highlight that open parking lots are more difficult to manage as compared to the closed parking lots and thus, a combination of possible algorithms is required to enhance efficiency (Paidi et al., 2018). Thus, a combination of machine vision and that of the convolutional neural network that consists of multi-agent systems is suggested in order to facilitate open parking space management. Ahad, Khan, and Ahmad (2016) identified the application of intelligent parking systems and they suggest that the web-based app system can be developed that allow the drivers to be interconnected helping them to find the parking space easily. The application is bound to make use of the web-based interface that keeps a log of all the entries as well as exits while displaying the free parking space at each junction.

### Clustering Algorithms for the Calculation of Analytical Statistics

It is important to note the functionality of clustering algorithms in the discovery of the relationships that exist between parking space detection based on the occupancy rates throughout the year. The study conducted by Aggarwal and Zhai (2012) highlights that the development of the cluster algorithms is beneficial in studying large data, which is why collaborative filtering can be achieved through its application. Moreover, as discussed by Wilson (2010), the process of data clustering is beneficial in application of models that can help in yielding better integrated processes. The clustering algorithms are therefore identified as the alternative and powerful meta-analysis tools that can analyse the massive volumes of data. The study perfected by Fahad et al. (2014) points to the availability of vast pools of knowledge and data, which needs to be organised and directed in the specific zones. Hence, clustering algorithms can be used as an option to achieve preferred results that can promote distinction of useful data from the non-related content. Furthermore, Boyinbode et al. (2010) have conducted a survey on clustering algorithms being used for wireless sensor networks and revealed that gathering data from diverse kinds of environment is possible through its application. Hence, it is possible to achieve positive results in the detection of free parking spaces too. Thus, the same is being planned in future research related to the parking space detection using web-based interface.

# References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. and Kudlur, M., (2016). Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)* (pp. 265-283).

Abe, Y., Ogitsu, T. and Mizoguchi, H. (2016). Design and Evaluation of Parking Position Detection with Human Cooperation for Automatic Parking. Journal of the Institute of Industrial Applications Engineers, 4(2), pp.100-105.

Aggarwal, C. and Zhai, C. (2012). A Survey of Text Clustering Algorithms. Mining Text Data, pp.77-128.

Ahad, A., Khan, Z. and Ahmad, S. (2016). Intelligent Parking System. World Journal of Engineering and Technology, 04(02), pp.160-167.

Al-Kharusi, H. and Al-Bahadly, I. (2014). Intelligent Parking Management System Based on Image Processing. World Journal of Engineering and Technology, 02(02), pp.55-67.

Baroffio, L., Bondi, L., Cesana, M., Redondi, A. and Tagliasacchi, M. (2015). A visual sensor network for parking lot occupancy detection in Smart Cities. 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT).

Boyinbode, O., Le, H., Mbogho, A., Takizawa, M. and Poliah, R. (2010). A Survey on Clustering Algorithms for Wireless Sensor Networks. 2010 13th International Conference on Network-Based Information Systems.

Bulan, O., Loce, R., Wu, W., Wang, Y., Bernal, E. and Fan, Z. (2013). Video-based real-time on-street parking occupancy detection system. Journal of Electronic Imaging, 22(4), p.041109.

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C. and Zhang, Z., 2015. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*.

Clear, T. (1997). Coupling and cohesion among disciplines. ACM SIGCSE Bulletin, 29(4), pp.14-16.

CyberAILab. (2018). CyberAILab | AI | Machine Learning | Computer Vision | Demos. [online] Available at: <https://www.cyberailab.com/home/a-closer-look-at-yolov3> [Accessed 26 May 2019].

Erica, E. and Sujata, S. (2014). Pioneering Project planning Strategy: An Integration of Agile and Six Sigma. International Journal of Computer Applications, 97(1), pp.41-44.

Fahad, A., Alshatri, N., Tari, Z., Alamri, A., Khalil, I., Zomaya, A., Foufou, S. and Bouras, A. (2014). A Survey of Clustering Algorithms for Big Data: Taxonomy and Empirical Analysis. IEEE Transactions on Emerging Topics in Computing, 2(3), pp.267-279.

Fung, R.H.Y., Chiu, D.K., Ko, E.H., Ho, K.K., & Lo, P. 2016. Heuristic usability evaluation of the University of Hong Kong Libraries' mobile website. The Journal of Academic Librarianship, 42(5), pp. 581-594.

Hui, J. (2019). *Object detection: speed and accuracy comparison (Faster R-CNN, R-FCN, SSD, FPN, RetinaNet and…*. [online] Medium. Available at: <https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359/> [Accessed 27 May 2019].

Ionita, A., Pomp, A., Cochez, M., Meisen, T. and Decker, S. (2018). Where to Park?. Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics - WIMS '18.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., & Darrell, T. (2014, November). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia* (pp. 675-678). ACM.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S. and Darrell, T., 2014, November. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia* (pp. 675-678). ACM.

Khajouei, R., Gohari, S.H., & Mirzaee, M. 2018. Comparison of two heuristic evaluation methods for evaluating the usability of health information systems. Journal of biomedical informatics, 80, pp.37-42.

Krumke, S., Quilliot, A., Wagler, A. and Wegener, J. (2013). Models and Algorithms for Carsharing Systems and Related Problems. Electronic Notes in Discrete Mathematics, 44, pp.201-206.

Kushniruk, A.W., Monkman, H., Tuden, D., Bellwood, P., & Borycki, E. M. 2015. Integrating heuristic evaluation with the cognitive walkthrough: development of a hybrid usability inspection method. ITCH, pp. 221-225.

LeRouge, C., Hasselquist, M.B., Kellogg, L., Austin, E., Fey, B., Hartzler, A., Flum, D.R., & Lavallee, D.C. 2017. Using Heuristic Evaluation to Enhance the Visual Display of a Provider Dashboard for Patient-Reported Outcomes. eGEMs, 5(2).

Lin, T., Rivano, H. and Le Mouel, F. (2017). A Survey of Smart Parking Solutions. IEEE Transactions on Intelligent Transportation Systems, 18(12), pp.3229-3253.

Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. and Zitnick, C.L., 2014, September. Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740-755). Springer, Cham.

lv, D., Yang, Y., Zhang, Y. and Fu, X. (2011). To analyze the parking problems and find solutions in urban central areas. 2011 International Conference on Remote Sensing, Environment and Transportation Engineering.

Ma, S., Jiang, H., Han, M., Xie, J. and Li, C. (2017). Research on Automatic Parking Systems Based on Parking Scene Recognition. IEEE Access, 5, pp.21901-21917.

Mainetti, L., Marasovic, I., Patrono, L., Solic, P., Stefanizzi, M. and Vergallo, R. (2016). A Novel IoT-aware Smart Parking System based on the integration of RFID and WSN technologies. International Journal of RF Technologies, 7(4), pp.175-199.

McCoy, K. (2017). Drivers spend an average of 17 hours a year searching for parking spots. [online] Usatoday.com. Available at: <https://www.usatoday.com/story/money/2017/07/12/parking-pain-causes-financial-and-personal-strain/467637001/> [Accessed 13 Jan. 2019].

Nurminen, J., Rainio, K., Numminen, J., Syrjänen, T., Paganus, N. and Honkoila, K. (2019). Object Detection in Design Diagrams with Machine Learning. Advances in Intelligent Systems and Computing, 977, pp.27-36.

Paidi, V., Fleyeh, H., Håkansson, J. and Nyberg, R. (2018). Smart parking sensors, technologies and applications for open parking lots: a review. IET Intelligent Transport Systems, 12(8), pp.735-741.

Paz, F., Paz, F.A., Villanueva, D., & Pow-Sang, J.A., 2015, April. Heuristic evaluation as a complement to usability testing: a case study in the web domain. In 2015 12th International Conference on Information Technology-New Generations (pp. 546-551). IEEE.

Piovesan, N., Turi, L., Toigo, E., Martinez, B. and Rossi, M. (2016). Data Analytics for Smart Parking Applications. Sensors, 16(10), p.1575.

Qu, Z., Jin, Y. and Feng, J. (2013). Fast Human Detection Using Dynamic Contour and Histograms of Oriented Gradients. *Applied Mechanics and Materials*, 347-350, pp.3600-3603.

Redmon, J. (2013). Darknet: Open source neural networks in c.

Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

Scardapane, S. and Di Lorenzo, P. (2017). A framework for parallel and distributed training of neural networks. Neural Networks, 91, pp.42-54.

Taniguchi, E. (2014). Concepts of City Logistics for Sustainable and Liveable Cities. Procedia - Social and Behavioral Sciences, 151, pp.310-317.

Thonduri, V. (2018). System for Smart Parking using Internet of Things (IoT). International Journal for Research in Applied Science and Engineering Technology, 6(3), pp.187-191.

Tian, Q., Guo, T., Qiao, S., Wei, Y. and Fei, W. (2014). Design of Intelligent Parking Management System Based on License Plate Recognition. Journal of Multimedia, 9(6).

True, N. (2015). AN IMPLEMENTATION OF VACANT PARKING SPACE DETECTION SYSTEM. International Journal of Advance Engineering and Research Development, 2(04).

Vlahogianni, E., Kepaptsoglou, K., Tsetsos, V. and Karlaftis, M. (2015). A Real-Time Parking Prediction System for Smart Cities. Journal of Intelligent Transportation Systems, 20(2), pp.192-204.

Wang, H. and He, W. (2011). A Reservation-based Smart Parking System. 2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS).

Wang, W., Song, Y., Zhang, J. and Deng, H. (2014). Automatic parking of vehicles: A review of literatures. International Journal of Automotive Technology, 15(6), pp.967-978.

Waters, K. (2009). Prioritization using moscow. *Agile Planning*, *12*, 31.

Wilson, C. (2010). Mathematical modeling, clustering algorithms and applications. New York: Nova Science Publishers.

Yi, Z., Yongliang, S. and Jun, Z. (2019). An improved tiny-yolov3 pedestrian detection algorithm. Optik, 183, pp.17-23.

# Bibliography

Abad, F., Bendahan, R., Wybo, S., Bougnoux, S., Vestri, C. and Kakinami, T., 2007. Parking space detection. Proc. 14th World Congr. Intell. Transp. Syst, pp.1-8.

Al-Kharusi, H. and Al-Bahadly, I., 2014. Intelligent parking management system based on image processing. World Journal of Engineering and Technology, 2(02), p.55.

Al-Kharusi, H., 2014. Intelligent car parking management system: a thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering in Electronic and Computer System Engineering at Massey University, Palmerston North, New Zealand (Doctoral dissertation, Massey University).

Bibi, N., Majid, M.N., Dawood, H. and Guo, P., 2017, March. Automatic Parking Space Detection System. In 2017 2nd International Conference on Multimedia and Image Processing (ICMIP) (pp. 11-15). IEEE.

Burduk, R., Kurzynski, M., Wozniak, M. and Zolnierek, A. eds., (2011). *Computer Recognition Systems 4* (Vol. 95). Springer Science & Business Media.

Ji, Z., Ganchev, I., O'Droma, M. and Zhang, X., 2014, August. A cloud-based intelligent car parking services for smart cities. In General Assembly and Scientific Symposium (URSI GASS), 2014 XXXIth URSI (pp. 1-4). IEEE.

Kiruthika, S. and Surendran, D., 2016. Smart Car Parking Using Ardunio And Android Application. International Journal of Computer Science and Mobile Computing, 5(2).

Liu, J., Chen, R., Chen, Y., Pei, L. and Chen, L., 2012. iParking: An intelligent indoor location-based smartphone parking service. Sensors, 12(11), pp.14612-14629.

Nawaz, S., Efstratiou, C. and Mascolo, C., 2013, September. Parksense: A smartphone-based sensing system for on-street parking. In Proceedings of the 19th annual international conference on Mobile computing & networking (pp. 75-86). ACM.

Nurminen, J. K., Rainio, K., Numminen, J. P., Syrjänen, T., Paganus, N., & Honkoila, K. (2019, May). Object Detection in Design Diagrams with Machine Learning. In *International Conference on Computer Recognition Systems* (pp. 27-36). Springer, Cham.

Revathi, G. and Dhulipala, V.S., 2012, February. Smart parking systems and sensors: A survey. In Computing, Communication and Applications (ICCCA), 2012 International Conference on (pp. 1-5). IEEE.

Saxena, S., Sharma, S. and Sharma, N., 2016. Parallel image processing techniques, benefits and limitations. Research Journal of Applied Sciences, Engineering and Technology, 12(2), pp.223-238.

Stolfi, D.H., Alba, E. and Yao, X., 2017, June. Predicting car park occupancy rates in smart cities. In International Conference on Smart Cities (pp. 107-117). Springer, Cham.

Vlahogianni, E.I., Kepaptsoglou, K., Tsetsos, V. and Karlaftis, M.G., 2016. A real-time parking prediction system for smart cities. Journal of Intelligent Transportation Systems, 20(2), pp.192-204.

Yusnita, R., Norbaya, F. and Basharuddin, N., 2012. Intelligent Parking Space Detection System Based onImage Processing. International Journal of Innovation, Management and Technology, 3(3), p.232.

# Appendices

## Terms of Reference

Project Title

‘Vision-Based Detection of Parking Spaces Occupation and Violation’

Background

Nowadays, technology has become advanced and accessible enough that many drivers use it to be informed about their journeys’ congestion levels, closures, route recommendations, etc… all in the aim of having the best possible experience with least delay. However, many journeys end with the driver having to look for a parking space, which causes tremendous stress on parking sites in highly populated areas, especially in peak times. This is where this project comes in, with the aim to improve the how parking spaces are managed by keeping the site managers in direct contact with live information about the current occupation and violations in their parking site.

**Problem**

Looking for a parking spaces in the city-centre is often stressful due to their popularity early in the day. Additionally, small car parks are often that fill quickly are highly popular, which often leads to other drivers queuing to wait for a space or driving to a different car park. Due to the frequency of this issue, it carries an economic cost for wasted time and an environment cost for vehicles’ Carbon Dioxide emissions. Although, other car parks can have empty spaces with very low demand for the whole day.

Primarily, there are two key factors that drive the need of utilising intelligent and efficient parking systems.

* The continuous increase of vehicular traffic. According to the Department of Transport’s vehicle licensing statistic, the total number of registered vehicles in 2017 exceeded 37.7 million vehicles; 3.1 million of which were registered for the first time. The figure shows an increase of 1.3% compared to 2016 and 2.5% to 2015. Logically, the increase in number of vehicles puts tremendous stress on traffic flow, parking spaces and CO2 emission rate.
* The growing condensation of people in cities. According to the UN’s data booklet (2016), 54.5% of the world’s population are living in a city with at least 500 thousand people and estimated to reach 60% by 2030. The UN’s agriculture database also shows, that 83.4% of the UK’s population are residing in urban areas with a forecasted continuous increase of at least 0.2% every following year.

Consequently, if the two key factors are present it drastically increases the probability of spending more time to find a parking space in the city-centre; this is particularly true for highly condensed cities. A statistic published by the Department of Transport, shows the number of licensed cars in London (2.67m cars) to be higher than Scotland as a whole (2.46m cars), which clearly demonstrate the difference in size-of-area to number-of-cars ratio to be taken into consideration. Correspondingly, the average driver in London spends 67 hrs/annum, according to study performed by INRIX (2017) - a global Software as a Service and Data as a Service company that specialises in connected car services and transportation analytics.

**Solution**

The ability to improve the efficiency of parking sites by calculating the occupation rate and vehicles violations in the designated parking spaces and informing the parking administrators about them will help them find more solutions to utilise their spaces better or stop violations which disrupts the usage of appropriate drivers. This can improve the overall experience of looking for a parking space, while reducing the time spent looking for one. As a result, improving the efficiency of the parking system and traffic flow in the area.

After researching similar projects, sensors were the first option to be considered. However, it quickly became apparent that using sensors would be to detect the occupation of parking spaces will be undesirable, from a client’s perspective. This was concluded based on number of reasons:

* Some methods require a sensor for each parking space. This introduces maintenance overhead, which will become problematic when the recorded data is analysed.
* They are susceptible to external factors, such as, weather (e.g. water damage).
* Some require to be implemented in the infrastructure of the parking space.
* Inability to detect cars in the process of finding a parking space
* Offers less expansion opportunities.

Therefore, I decided to the next best option, machine learning to perform object detection and recognition. This is because, the needed equipment (a camera with a clear view of the parking spaces) is already installed in most car parks; as safety is a big concern when it comes to car parks. This option will also be cost effective and easier to implement. This also allows for future expansions, e.g. car theft detection, overdue stay of cars, rescaling parking spaces size to vehicles size, and many more.

The method implemented to inform drivers can vary from a head unit application, mobile application or/and website. While the first two options (head unit application, mobile application) are platform dependant, a website can work on any mobile device. Which allows for much better accessibility, as it can be checked without being in the car (Head unit) or using any device regardless of operating system or form factor. A statistical analysis of parking spaces usage can also be considered and displayed to users, allowing them to plan their journeys with a higher chance of finding a parking space.

Proposed Work

The project will start by researching, reviewing and evaluating existing work done on the subject to avoid known issues or mistakes, all the while, giving an overview of their current progress, functionality and limitations. The analysis does not discriminate nor require previous work to have the exact same functionality, if, they are compatible with the proposed system. For example, highways vehicle detection systems are taken into consideration for this project’s vehicle detection and tracking system. Additionally, it will not discriminate against existing work based on country, language of implementation or choice of equipment, as taking them into consideration should not affect the compatibility of the project deliverables. Following the initial research, a literature review will be initiated, where findings including different detection and recognition algorithms, machine learning tools and equipment will be compiled, analysed and synthesised. The outcome of the analysis will be used for the implementation of the proposed system. Additionally, it will

After concluding the literature review, a set of documents detailing the system’s requirements, design and structure will have to be produced before any development is begun. The system requirement document will specify the functionality and actions performed by the system and prioritise them based on their importance to insure a highly functional product. Design documentations will include, interface wireframes and Unified Modelling Language (UML) diagrams. The system will be designed to using the Object-Oriented Programming (OOP) paradigm and follow SOLID – a mnemonic acronym referring to five design principles; **S**ingle responsibility, **O**pen/closed, **L**iskov substitution, **I**nterface segregation and **D**ependency inversion, Robert and Micah (2006) – and Keep It Simple Stupid (KISS) design principles. Lastly, the structure documentations will detail the structure of stored data locally or otherwise.

System development will be started once all necessary documentations is completed. Development will strictly follow design and requirement specifications. The system most basic architecture will be developed first, followed by blocks of functionality based on the System requirements, to ensure the highest possible compatibility and modularity. There will be an emphasis on clean coding and commenting practices by using Integrated Development Environment (IDE) Style Cop plug-in to stop the project building process, if Best Practices Rules are violated. In the case of a necessary amendment in the design specification, it should be amended and revised before continuing the system development. Insuring correct and up-to-date documentation throughout the project is considered vital.

Test-Driven Development (TDD) was avoided due its time consumption and learning overhead. However, ensuring a modular system allows for the use of Unit Testing without strictly following the TDD practices. This will allow for the test of separate functionality and integration testing in the development phase. Using unit tests will reduce the time taken to test the system at the end of the project, as they should cover all system functionalities, as well as, their compatibility with each other using the integration unit tests.

The evaluation of the system will be included in the final phase of the project. It will include a reflection of the project process altogether and point out challenges and observations faced. It will also recommend possible expansions along with current limitations of the system.

Aims

* “The development of a vehicle recognition and detection within parking spaces using a deep neural network to allow for parking occupation and violation detection”.

Objectives:

1. Review and evaluate existing work on smart parking systems.
2. Produce prioritised system specification requirements document.
3. Identifying and learning/improving knowledge required to complete the project (technical, academic and professionalism) - using online tutorials, labs from previous years.
4. Produce system structure and behaviour documentation, detailing system’s classes (Class diagram), sub system interactions (Communication and Use Cases diagrams) and complex processes (Activity diagrams) and interface wireframes
5. Produce system skeleton structure design specification document.
6. Setup development environment.
7. Development, testing and implementation of vehicle detection within parking spaces.
8. Reflection on project, product and personal management and performance.
9. Evaluate and analyse system to verify the implementation of all requirements and establish conclusion and recommendations.
10. Production of project report chapters.

Skills

Knowledge required in the completion of the project will depend on existing knowledge from previous years and newly acquired knowledge during the project process through final year modules and online resources.

|  |  |
| --- | --- |
| Skill | Acquisition method |
| Academic writing | Existing & new knowledge. The skill has been developed throughout the years in the course, however, much improvement is required to ensure the project report’s quality. This will be enhanced using online resources (University’s Skills Plus and PluralSight) and observations of existing academic literature during the literature review. |
| C# and .NET Framework | Existing knowledge. Worked on multiple projects that use the same technology during the author’s placement year. |
| OOP Design Principles | Existing & new knowledge. Acquired during the author’s placement year and second year’s Program Design & Development (CM0570) and Small Embedded Systems (CM0506) modules. However, additional knowledge is to be acquired using online resources (PluralSight). |
| Machine learning modelling and training | New knowledge. To be acquired through final year’s AI and Affective Computing (CM0671) and Machine Learning and Computer Vision(CM0669) modules, as well as, private study using online resources (PluralSight) |
| Design Specification Documents (interface, behaviour and structure wise) | Existing & new knowledge. Acquired during the author’s placement year, second year’s Program Design & Development (CM0570) module and first year’s Systems Analysis (CM0432) module. However, additional knowledge is to be acquired using online resources (PluralSight). |
| Unit Testing and Mocking. | Existing knowledge. Acquired during the author’s placement year. If needed, additional information can be acquired using online resources (PluralSight). |
| Website Development and Deployment, using ASP.NET MVC Architectural Pattern. | Existing knowledge. Acquired during the author’s placement year. If needed, additional information can be acquired using online resources (PluralSight). |

References

Dawei, L., Yonghong, Y., Yongming, Z., and Xinsha, F. (2011). 'To analyze the parking problems and find solutions in urban central areas', *2011 International Conference on Remote Sensing,* pp.5606-5609. doi: 10.1109/RSETE.2011.5965623

Department of Transport (2016) *Vehicle Licensing Statistics: Quarter 4 2015*. Available at: <https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/516429/vehicle-licensing-statistics-2015.pdf> (Accessed at: 10/11/2018).

Department of Transport (2017) *Vehicle Licensing Statistics: Annual 2016*. Available at: <https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/608374/vehicle-licensing-statistics-2016.pdf> (Accessed at: 10/11/2 018).

Department of Transport (2018) *Vehicle Licensing Statistics: Annual 2017*. Available at: <https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/716075/vehicle-licensing-statistics-2017-revised.pdf> (Accessed at: 10/11/2018).

INRIX (2017) *Searching for Parking Costs the UK £23.3 Billion a Year* [Press release]. 12 July. Available at: <http://inrix.com/press-releases/parking-pain-uk/> (Accessed at: 11/11/208).

Lin, T., Rivano, H., Le Mouel, F. (2017) 'A Survey of Smart Parking Solutions', *IEEE Transactions on Intelligent Transportation Systems*, Vol.18 (12), pp.3229-3253. doi: 10.1109/TITS.2017.2685143 <https://ieeexplore.ieee.org/document/7895130>

Martin, R. and Martin, M. (2006) *Agile Principles, Patterns, and Practices in C#*. Upper Saddle River, N.J.: Pearson Prentice Hall. ISBN-13: 978-0-13-185725-4

Paidi, V. G., Fleyeh, H., Håkansson, J., and Nyberg, R. (2018) 'Smart parking sensors, technologies and applications for open parking lots: A review.' *IET Intelligent Transport Systems,* *12*(8), 735-741. doi: 10.1049/iet-its.2017.0406

Polycarpou, E., Lambrinos, L., and Protopapadakis, E. (2013). 'Smart parking solutions for urban areas.' *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a,* 1-6. doi: 10.1109/WoWMoM.2013.6583499

United Nations (2016) *The World’s Cities in 2016, Data Booklet (ST/ESA/ SER.A/392)*. New York: United Nations Department of Economic and Social Affairs. Available at: <http://www.un.org/en/development/desa/population/publications/pdf/urbanization/the_worlds_cities_in_2016_data_booklet.pdf> (Accessed: 10/11/2018)

United Nations (2018) *Interactive Data Queries: World Urbanization Prospects*. New York: United Nations Department of Economic and Social Affairs. Available at: <https://population.un.org/wup/DataQuery/> (Accessed: 10/11/2018)

Resources

All resources required to complete the project are available on the labs in the CIS building or will be legally acquired during the project process.

* Microsoft Visio or StarUML: To generate the system designs and Proof of concept.
* Development PC or laptop: Will be used to develop and demonstrate the product.
* Microsoft Visual Studio 2017 (IDE): To develop the product. Alternatively, any text editor can be used, but, MS Visual Studio offers integrated debugger and console compilers which will increase the developer’s productivity.
* GIT repository: to ensure source code is backed and accessible at any time.

Structure & contents of project report

Abstract

Chapter 1) Introduction

1.1 Aims and Objectives

1.2 Overview – Project purpose, scope and main features

1.3 Context of the Problem

1.4 Implemented Tools and Techniques

Chapter 2) Literature Review

Chapter 3) Requirement Specification

3.1 Introduction

3.2 Commentary

3.2.1 Overall characteristics

3.2.2 YOLO Prediction framework

3.2.3 Performance and Redeployment

3.2.4 Storage and Privacy

3.2.5 User Interface (11.1.2 & 11.1.6)

3.3 Summary

Chapter 4) System modelling & design

4.1 Introduction

4.2 Class Diagram

4.3 Activity Diagram

4.4 Dictionary and Data Transfer Objects (DTOs)

4.5 YOLOv3 network Structure

4.6 User Interface

4.7 Summary

Chapter 5) System development

5.1 Introduction

5.2 Main program

5.3 Detection Module

5.3.1 Video input and output

5.3.2 Detector training and Viewer integration

5.3.3 Introducing ROIs

5.3.4 Integration

5.4 User Interface

5.5 Summary

Chapter 6) System testing

6.1 Introduction

6.2 Test plan and Results

6.3 Performance Testing

6.4 Summary

Chapter 7) Evaluation

7.1 Product Evaluation

7.1.1 Requirements evaluation

7.1.2 Fitness for Purpose

7.1.3 Build Quality

7.1.4 Usability Evaluation

7.2 Personal and Process Evaluation

Chapter 8) Conclusion and Further Work

8.1 Introduction

8.2 Project Summary

8.3 Conclusion and Recommendation

Chapter 9) References

Chapter 10) Bibliography

Chapter 11) Appendices

11.1 Terms of Reference

11.2 Requirements Specifications

11.3 Design Specifications

Marking scheme

Project type

“Software Engineering Project”

Project report (40%)

The project follows the marks distribution table below.

|  |  |  |
| --- | --- | --- |
| Report Component | Marks | Report Chapters |
| Abstract & Introduction | 5 | Abstract  Introduction |
| Analysis | 30 | Literature Review  Requirement Specification |
| Synthesis | 30 | System modelling & design  System development  Factory testing |
| Evaluation, Conclusion and Recommendations | 10 | Evaluation  Conclusion & recommendations |

Product (50%)

The default product marking scheme is desired.

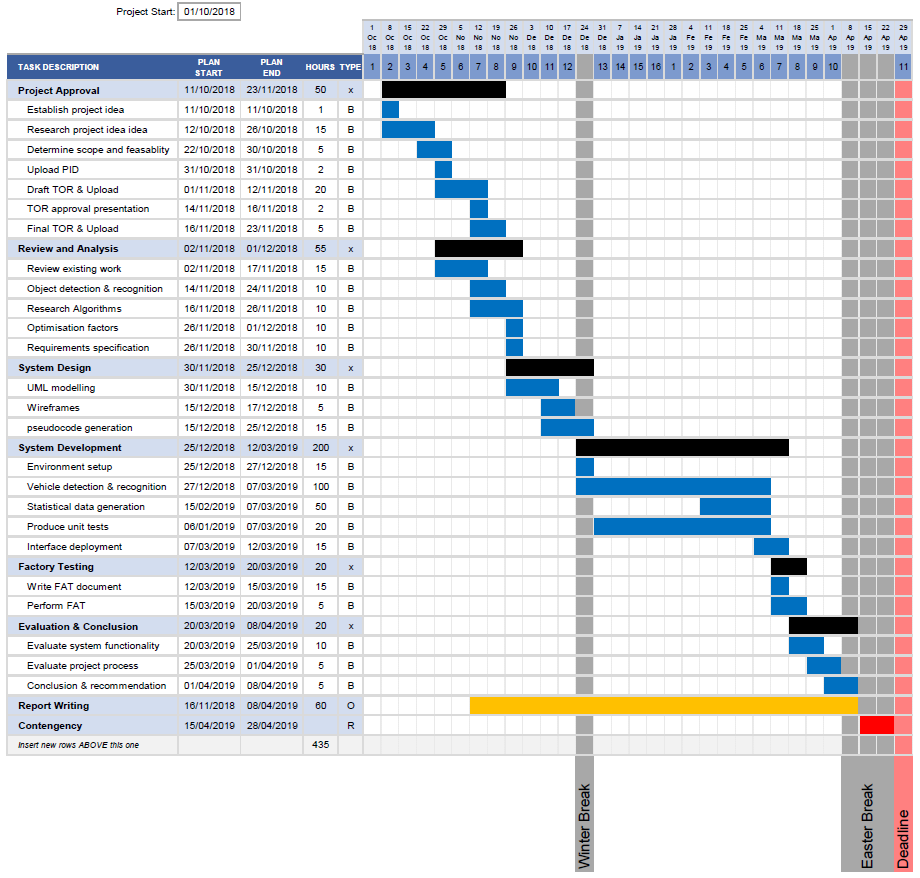
* **Fitness for Purpose (40%)**
* **Build Quality (60%)**

Product deliverables:

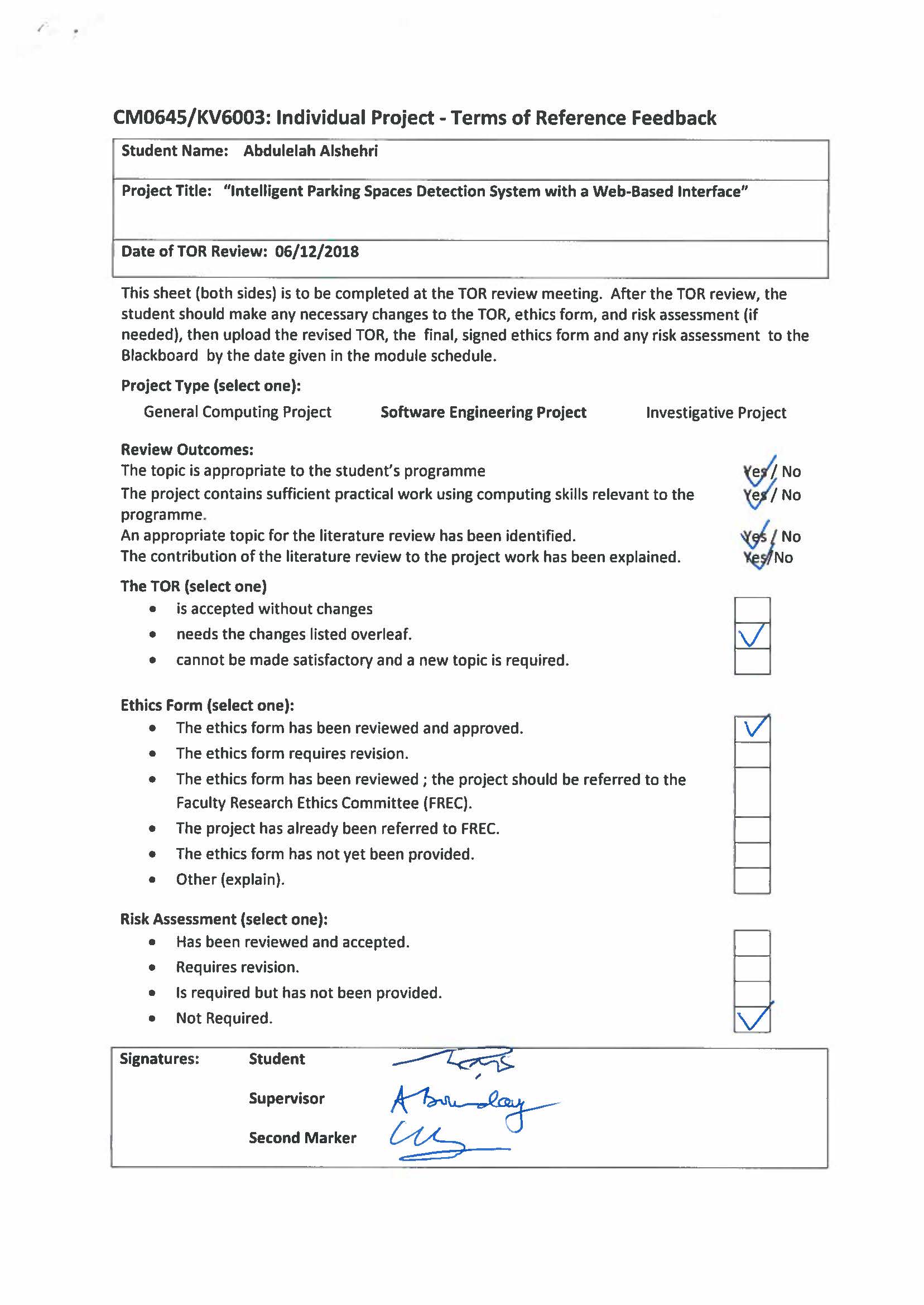
* **Requirement specification**
* **Design Documentations (Class, Use-Cases, Sequences and Activity diagrams and interface wireframe)**
* **Source Code to build application executables**
* **Test plan and results**

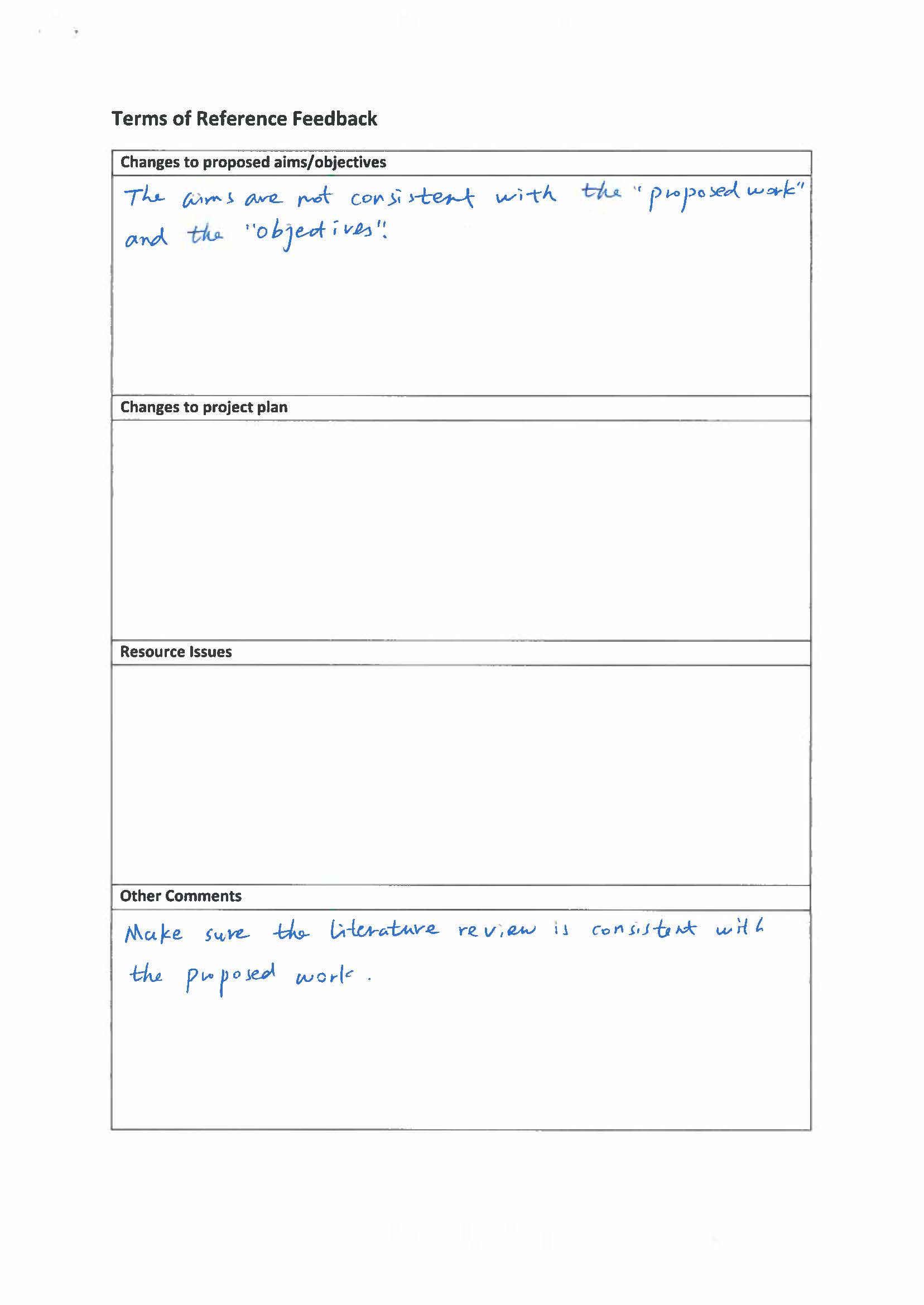
Viva (10%)

Project plan



Approved Ethical Form





## Requirements Specifications

### General

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement | Description | Type | Priority |
| 1. OS | The system is operational in Windows 10 environment and built in Visual studio IDE using C# | Operational | Must |
| 1. Internet connection | The system is able to run in offline mode. | Operational & Security | Must |
| 1. Threads-Safe | The system is able to run its different modules simultaneously and follow the OOP structure to create multiple instances of certain modules | Operational | Must |
| 1. Error Handling | The system is able to tolerate errors and not cause the system to crash | Operational | Should |
| 1. Display Appropriate error msgs | Clear error messages should be displayed letting the user what went wrong | Accessibility | Should |
| 1. Error prevention | Whenever appropriate system should stop users from performing invalid actions | Operational | Should |
| 1. User aware | Always maintain user feedback. I.e. loading screen should be indicated | Accessibility | Should |
| 1. Extensions | The proposed system can easily implement future | extensibility | Should |
| 1. Configurable | Application configuration files are produced to easily manipulate application functionality | Operational | Could |
| 1. Data Logging | Information processed are logged in a suitable format for future use. | Storage | Could |

### Dashboard Interface

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement | Description | Type | Priority |
| 1. Information | Allow users to view current state of added parking locations | Functional | Must |
| 1. Locations | Ability to add/remove locations profiles. | Functional | Must |
| 1. Comprehendible | The UI must be designed in a way to allow users to find the logical order of its application | Accessibility | Should |
| 1. Automatic updates | Information displayed on the dashboard will be automatically updated | Functional | Could |
| 1. Export Information | Allow information displayed on the screen to be exported into an excel sheet. | Functional | Would |
| 1. Print Information | Allow information displayed on the screen to be printed. | Functional | Would |

### Video Input manager

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement | Description | Type | Priority |
| 1. Detection | Detect all devices installed on the workstation. | Functional | Must |
| 1. Information | Retrieve information of all devices, i.e. GUID, name and description. | Functional | Should |

### Training a model

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement | Description | Type | Priority |
| 1. Algorithm | Setup Darknet framework to train and output the prediction model | operational | Must |
| 1. Images | Images are collected to train the network on 3 objects (Cars, Trucks and Motorbikes) | Training | Must |
| 1. Images (Adv.) | Variety of different images (different lighting conditions, distances, orientations and clarity) are collected. | Training | should |
| 1. Annotations | Find an appropriate software to help manually annotating the collected images in YOLO format | Training | Must |
| 1. Segmenting the data | Images are segmented into three batches (training, testing and validation). | Training | should |
| 1. Network Design | Design a suitable network and generate a suitable YOLOv3 network configuration file | Training | Must |
| 1. Trained model | High accuracy and low error-rate of the trained model. | Functional | Should |

### System Detection

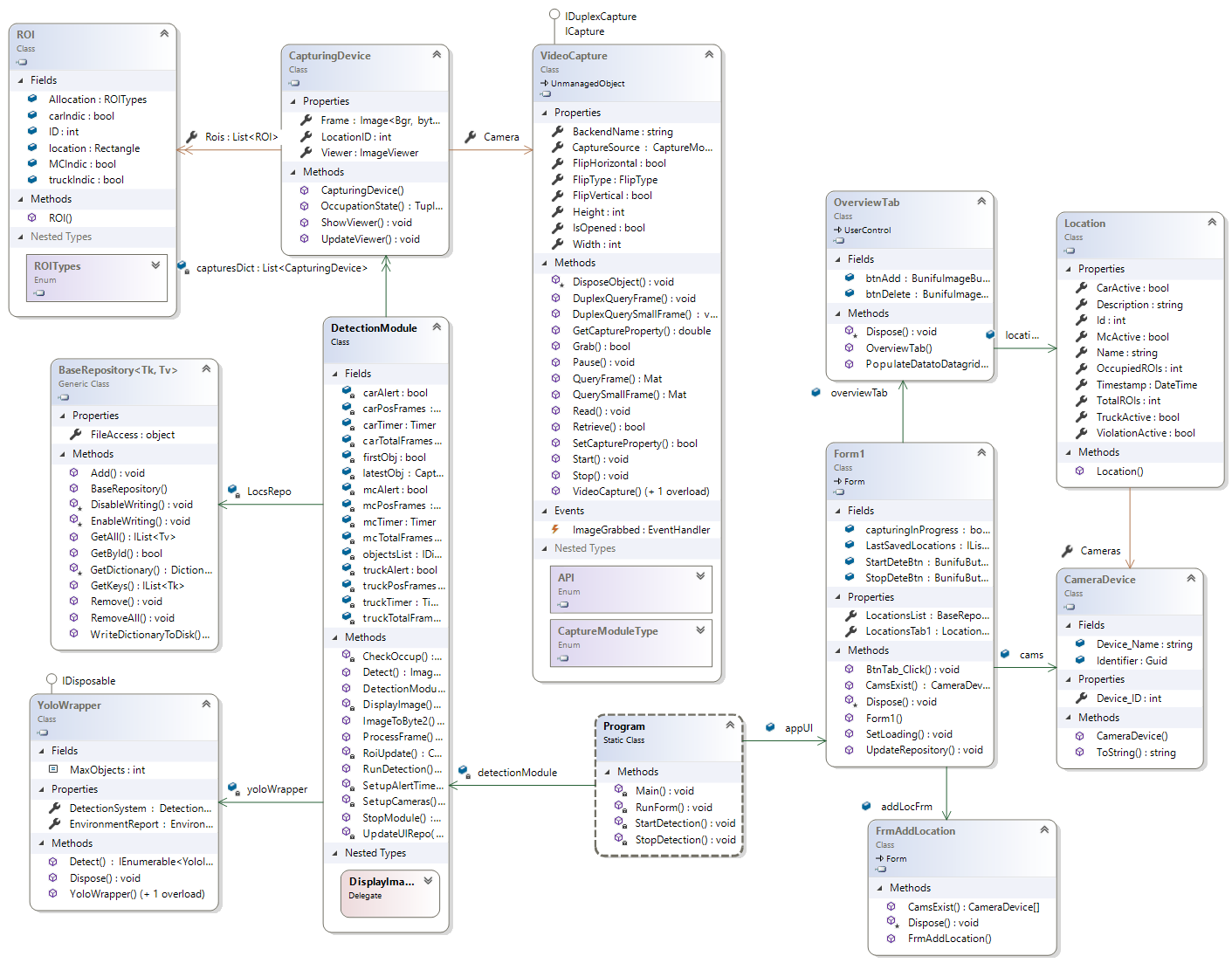
|  |  |  |  |
| --- | --- | --- | --- |
| Requirement | Description | Type | Priority |
| 1. Model | Implement a YOLO wrapper to implement the trained model | Functional | Must |
| 1. Real-time detection | Making detections in real time use the video input | Functional | Must |
| 1. Predictions information | Predictions made have other dependant information, e.g. Confidence, type etc… | Functional | Must |
| 1. No-CUDA assist | Ability for the system to run without having CUDA drivers/device. | Operational | Must |
| 1. CUDA assist | Ability for the system to, if applicable, make use of CUDA to increase the performance of predictions | Operational | Should |
| 1. Multiple instances | Multiple locations can be detected at once in addition to | Functional | Could |

### Detector Interface

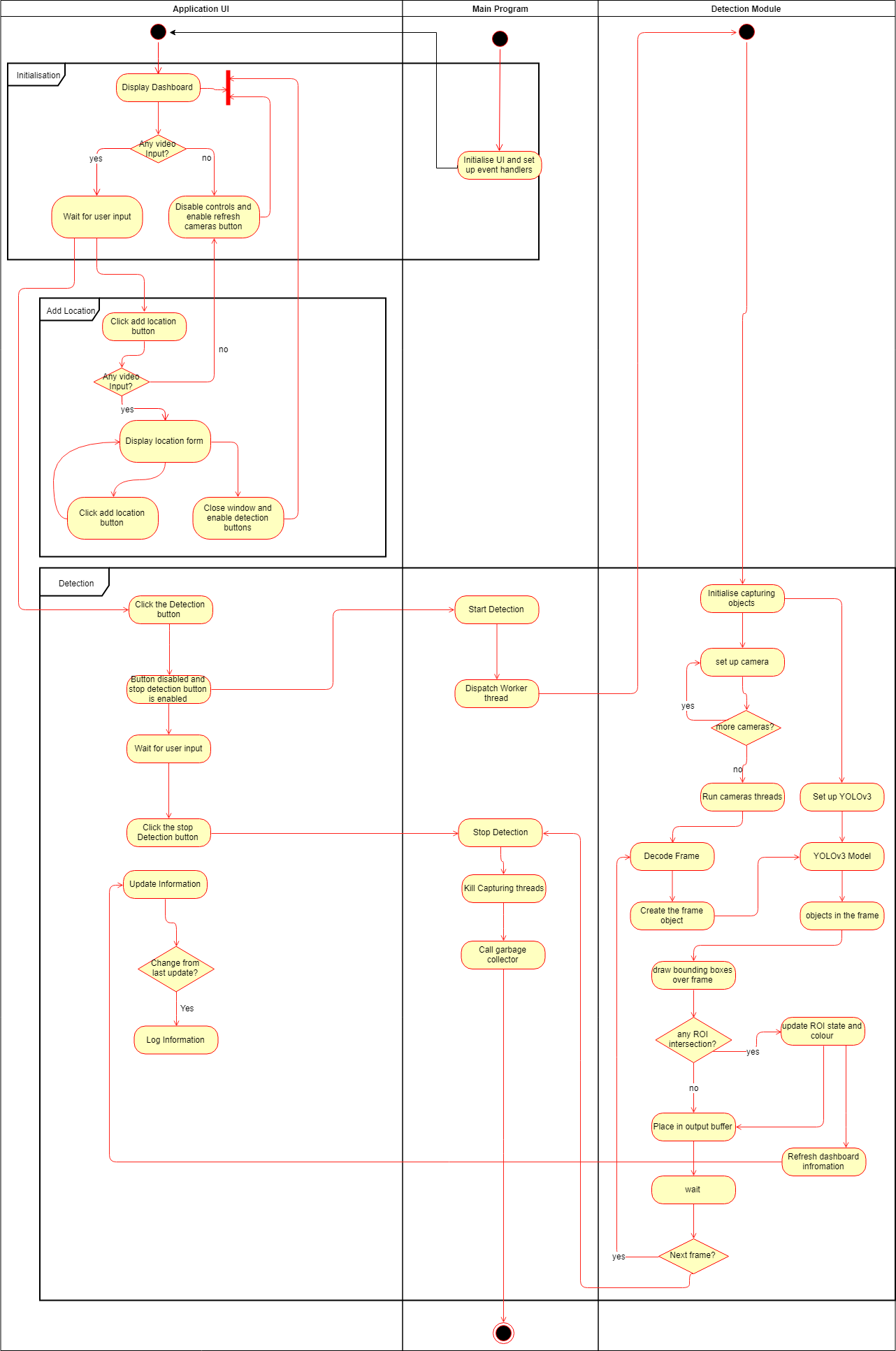
|  |  |  |  |
| --- | --- | --- | --- |
| Requirement | Description | Type | Priority |
| 1. Detection Information | Detected objects are displayed, with confidence levels | Functional | Must |
| 1. Regions of Interest | Dynamically adding/deleting regions of interest | Functional | Must |
| 1. ROI types | Dynamically setting ROI types (cars, truck or motorbikes) | Functional | Must |
| 1. Displaying occupation | Occupations of ROIs are indicated | Functional | Must |
| 1. Allow for debugging | Creating another child window to help debugging in the debugging process. | Debugging | Should |
| 1. Comprehendible | The UI must be designed in a way to allow users to find the logical order of its application | Accessibility | Should |

## Design Documentation

### Class Diagram



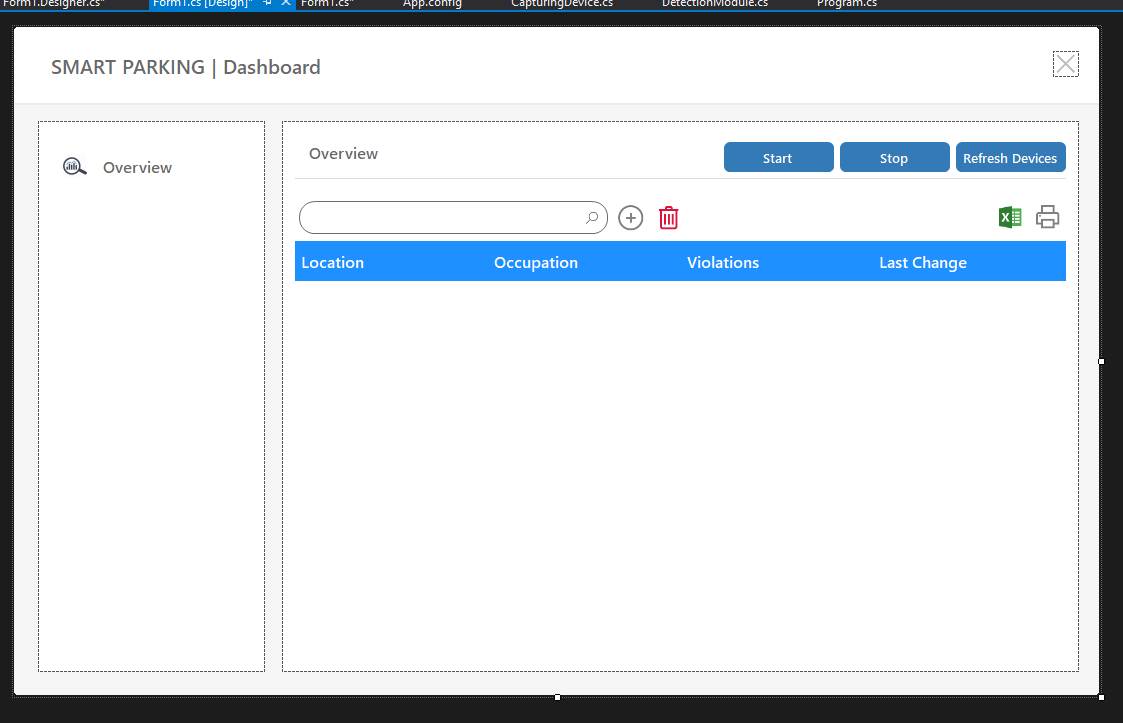
### Activity Diagram



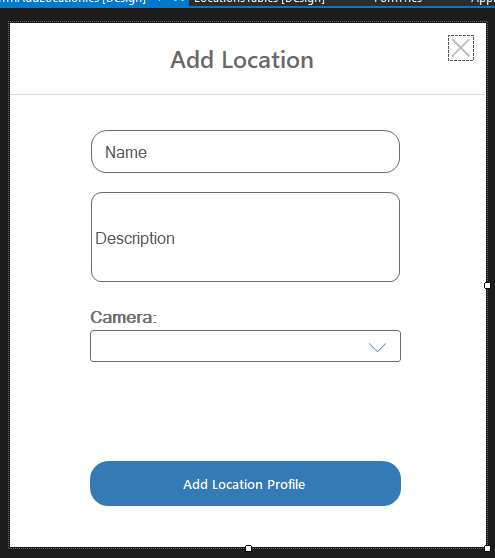
### System Use Case

### Interface wireframes

#### Dashboard Wireframe



#### Add location Wireframe



### Use Case Diagram

