TÜRKİYE CUMHURİYETİ
ADANA ALPARSLAN TÜRKEŞ SCIENCE AND TECHNOLOGY UNIVERSITY

GRADUATE SCHOOL
ELECTRICAL AND ELECTRONIC ENGINEERING DEPARTMENT

SMART DRIVING: DEVELOPING IMAGE PROCESSING AND CONTROL APPROACHES FOR SAFER ROADS AND ENHANCED MOBILITY

MUSTAFA EREN TOPAL

M.Sc.

ADANA 2025

TÜRKİYE CUMHURİYETİ
ADANA ALPARSLAN TÜRKEŞ SCIENCE AND TECHNOLOGY UNIVERSITY

GRADUATE SCHOOL
ELECTRICAL AND ELECTRONIC ENGINEERING DEPARTMENT

SMART DRIVING: DEVELOPING IMAGE PROCESSING AND CONTROL APPROACHES FOR SAFER ROADS AND ENHANCED MOBILITY

MUSTAFA EREN TOPAL

M.Sc.

THESIS ADVISOR
PROF. DR. ÖNDER TUTSOY

ADANA, 2025

DECLARATION OF CONFORMITY

In this thesis, which was prepared following the thesis writing rules of Adana Alparslan Türkeş Science and Technology University Institute of Graduate School, I declare that I provide all the information, documents, evaluations and results in accordance with scientific ethics and moral codes without resorting to any means or assistance that would be contrary to scientific ethics and traditions. I also declare that I refer to all the articles I used in this study with appropriate references and accept all moral and legal consequences if a situation is found contrary to my statement regarding my work.

24/01/2025

[Signature]

Mustafa Eren TOPAL

ÖZET

AKILLI SÜRÜŞ: DAHA GÜVENLİ YOLLAR VE GELİŞTİRİLMİŞ HAREKETLİLİK İÇİN GÖRÜNTÜ İŞLEME VE KONTROL YAKLAŞIMLARININ GELİŞTİRİLMESİ

Mustafa Eren TOPAL

Yüksek Lisans, Elektrik-Elektronik Anabilim Dalı

Danışman: PROF. DR. ÖNDER TUTSOY

Ocak 2025, 75 sayfa

Günümüzde modern kentlerde araç sayısının artışı, otopark yeri ve kaza sorunları da beraberinde getirmiştir. Topluma açık alanlarda sürücülerin otopark yeri arayışları ve dikkat dağınıklığı, trafik sıkışıklığına, çevresel kirliliğe ve zaman kaybına yol açmaktadır. Bu sorunların çözümüne yönelik yapılan araştırmada, görüntü işleme ve kısa yol bulma teknikleri kullanarak otopark yönetimine ve sürüş güvenliği sistemlerine katkı sağlanması hedeflenmiştir. Akıllı Ulaşım Sistemlerine (AUS) katkı sağlayan bu çalışmada Toplanmış Kanal Özellikleri (TKÖ) algoritması kullanılarak, araçların tespiti ve konumlandırılması yapılmıştır. Araç konumlandırmaları kullanılarak, otopark alanlarının doluluk haritası oluşturulurken, otoyol için geliştirilen algoritmada ise araçların konumları sürücü niyeti analizlerinde kullanılmıştır. Otopark yeri için geliştirilen algoritmada, Genişlik Öncelikli Arama (GÖA) algoritması kullanılarak yeni gelen sürücüler, boş otopark alanlarından seçilen hedef otopark alanına en kısa yol hesaplanarak yönlendirilmiştir. Otoyol için geliştirilen sürücü niyeti algoritmasında ise araçların konumları ve sürüş hareketleri analiz edilerek, sürücüye geri bildirimler verilmiştir. Yapılan çeşitli testler sonucunda, otopark yönetimi algoritması, otopark alanında bulunan araçları doğru tespit etmiş ve yeni gelen sürücüleri seçilen hedef otopark alanlarına doğru bir şekilde yönlendirilmiştir. Sürücü niyeti algoritması ise başarılı bir şekilde sürücüleri uyarmıştır. Otopark yönetimi algoritması kullanıldığında %61,8 oranında arama süresinde verimlilik kazandırmıştır. Yapılan çalışma sonucunda, etkili otopark yönetimi ve sürüş güvenliğine katkı sağlanmış; ayrıca emisyon ve kaynak israfı azaltılarak sürdürülebilirliğe katkıda bulunulmuştur.

Anahtar Kelimeler: Akıllı Ulaşım Sistemleri (AUS), görüntü işleme, kısa yol bulma algoritması, otopark yönetimi, sürücü niyeti analizi.

ABSTRACT

SMART DRIVING: DEVELOPING IMAGE PROCESSING AND CONTROL APPROACHES FOR SAFER ROADS AND ENHANCED MOBILITY

Mustafa Eren TOPAL

M.Sc., Department of Electrical and Electronic Engineering
Supervisor: PROF. DR. ÖNDER TUTSOY

January 2025, 75 pages

Currently, with the increase in the number of vehicles in modern cities, the parking problem and accidents have emerged as significant issues. The search for parking spaces and driver distractions in public areas contribute to traffic congestion, environmental pollution and time loss. The aim of this research is to contribute to parking management and safely driving solutions by using image processing and shortest path planning machine learning algorithms. Additionally, it focuses on advancing the Smart Transportation Systems (STS) through implementing the Aggregate Channel Features (ACF) algorithm to detect vehicles and determine their positions where the occupancy map of parking areas was determined by using determined locations. Moreover, these vehicle positions were used in the analysis of driver behavior for recognizing driver intentions. In the parking management algorithm, the Breadth-First Search (BFS) algorithm was modified to determine the shortest path to the target parking slot. This algorithm guides the newly entered vehicle from the entrance gate to the target parking area. In the driver intention algorithm, which is developed to prevent accidents, neighboring vehicle positions and their driving movements were analyzed and labeled as “calm” and “beware”. It has performed effectively in various testing scenarios, where the vehicles were correctly detected and labeled correctly according to their movements. In addition, the parking management algorithm was highly accurate in detecting the vehicles within the parking areas. It effectively guided the drivers to their allotted parking slots where it resulted in a 61.8% reduction in searching time. The results confirm that this research made a significant contribution to sustainability by improving parking management and driving safety while also reducing emissions and resource waste.

Keywords: Driver intention analysis, image processing, parking management, shortest path algorithm, Smart Transportation Systems (STS).

Dedication

ACKNOWLEDGEMENTS

I am proud to share the excitement and happiness of finishing my master thesis, which is the biggest step I took to complete my master's degree.

I would like to express my gratitude to Prof. Dr. Önder TUTSOY, who has always supported me on this path. He guided me and answered my questions with great understanding in this challenging process.

I am happy to dedicate my graduation thesis to my closest friends and family.

This work was supported by Adana Alparslan Türkeş Science and Technology University Scientific Research Coordination Unit. Project Number: 23303003

Mustafa Eren TOPAL

Adana, 2025

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

STS : Smart Transportation Systems

AVs : Autonomous Vehicles

V2V : Vehicle-to-Vehicle

V2I : Vehicle-to-Infrastructure

AI : Artificial Intelligence

PROMETHEUS : PROgraMme for a European Traffic of Highest Efficiency and Unprecedented Safety

ACF : Aggregate Channel Features

BFS : Breadth-First Search

IoT : Internet of Things

CNNs : Convolutional Neural Networks

GANs : Generative Adversarial Networks

ADAS : Advanced Driver-Assistance Systems

LiDAR : Laser Imaging Detection and Ranging

DARPA : The Defense Advanced Research Projects Agency

OEM : Original Equipment Manufacturers

NAHSC : National Automated Highway System Consortium

HOG + SVM : Histogram of Oriented Gradients with Support Vector Machine

DPM : Deformable Part Models

R-CNN : Region Based Convolutional Neural Networks

YOLO : You Only Look Once

SSD : Single Shot MultiBox Detector

M2Det : Multi-level Feature Pyramid Network

DETR : Detection Transformer

YOLOS : You Only Look at One Sequence

FPN : Feature Pyramid Networks

SNIPER : Scale Normalization for Image Pyramids with Efficient Resampling

LDCF : Locally Decorrelated Channel Features

MLP : Multi-Layer Perceptron

CONV	: Convolution Layer
ReLU	: Rectified Linear Unit
POOL	: Pooling Layer
NMS	: Non-Maximum Suppression
FAIR	: Facebook AI Research
NAS	: Neural Architecture Search
BiFPN	: Bidirectional Feature Pyramid Network
NLP	: Natural Language Processing
ViTs	: Vision Transformers
IoU	: Intersection over Union
GTA	: Graph Traversal Algorithms
DFS	: Depth First Search
DFST	: Depth-first Spanning Tree
APSP	: All-Pairs Shortest Path
ROI	: Region of Interest
VGG	: Visual Geometry Group
VBAPAS	: Vision-Based Autonomous Parking Assistance System
GM	: General Motors
HOG	: Histogram of Oriented Gradients
SVM	: Support Vector Machine

LIST OF SYMBOLS

$f(v)$	: Total Cost Function
$g(v)$	: Cost to Reach Node
$h(v)$	: Heuristic Estimate Cost
v	: Node
k	: Intermediate Vertex
(i, j)	: Source and Destination
V	: Number of Vertices
$O(V^3)$	: Time Complexity
$s(x)$	: Confidence Score
x	: Detection Window
T	: Classification Accuracy Threshold
I	: Input Image
C_k	: Channels
$f(x)$	: Feature Vector
A	: Aggregation Function
C	: Classifier
$b_{\sigma(1)}, b_{\sigma(2)}, \ldots, b_{\sigma(i)}$	: Bounding Boxes
D	: Set of Bounding Boxes
$Area(b_{\sigma(i)} \cap d)$	: Intersection Between Two Boxes
$R(b_{\sigma(i)}, d)$	: Overlap Ratio
d	: Individual Bounding Box
B	: Bounding Box of Detected Vehicles
$c_k = (c_x, c_y)$	: Centroids Coordinates
(x_{min}, x_{max})	: Bounding Box x Axis
(y_{min}, y_{max})	: Bounding Box y Axis

1. INTRODUCTION

In recent years, global population growth has reached approximately 8 billion people [1], while the global vehicle fleet has increased to over 1.47 billion [2]. The increasing growth of both the population and the vehicle fleet require the implementation of advanced traffic management and safety technologies. Smart Transportation Systems (STS) offer a technological advance in these fields due to their real-time data processing, vehicle-to-vehicle (V2V) communication and traffic flow optimization capabilities. STS aims to reduce traffic congestion, prevent accidents and improve overall transport utilization by exploiting innovations such as autonomous vehicles (AVs), connected infrastructure and sustainable transportation solutions [3][4]. For instance, V2V and vehicle-to-infrastructure (V2I) communications provide vehicles to communicate real-time road conditions, thus they significantly improve decision-making in complex traffic scenarios [5].

Furthermore, the AVs are considered the next evolutionary step in smart transportation, with countries such as Singapore showing a significant level of preparedness for level 5 autonomy at 25.5% [6]. Recent research has highlighted the economic benefits of the STS, predicting large reductions in fuel consumption and emissions, as well as high-cost savings from accident avoidance [7]. The application of Artificial Intelligence (AI) to these systems provides more efficient traffic flow and improved vehicle navigation, providing the framework for future smart cities [8]. However, the implementation of AVs requires a strong infrastructure, including 5G networks that enable low-latency communication between vehicles and road systems [9].

Countries are significantly increasing their investment in STS to address urbanization and mobility issues. For instance, Europe PROgraMme for a European Traffic of Highest Efficiency and Unprecedented Safety (PROMETHEUS) project demonstrated early improvements in autonomous driving technology, focusing on vehicle safety and traffic management through collaborative sensing and decision-making [10]. STS technology continues to evolve, and its deployment not only enhances road safety but also brings significant economic and environmental benefits [11]. The following section provides the purpose and contributions of the thesis, together with a detailed summary of its structure and main objectives.

1.1. Purpose and Contributions of the Thesis

The thesis aims to enhance the field of STS by combining image processing techniques and artificial intelligence technologies. The study uses the Aggregate Channel Features (ACF) algorithm to detect vehicles on highways and in public parking areas. The proposed system consists of two main components: a driver intention algorithm for highway scenarios and a Vision-Based Autonomous Parking Assistance System (VBAPAS) for public parking areas. VBAPAS monitors parking area occupancy from a specific camera angle, identifies empty parking spaces and provides output information to guide vehicles to the most suitable parking slots. Unlike existing vision-based parking management algorithms, which only monitor occupancy states and detect vacant slots, our system fills the gap by directing drivers to the fastest and most efficient paths to available spaces, resulting in increased overall efficiency.

The driver intention algorithm analyses the spatial positions of neighboring vehicles relative to lane boundaries while driving on a highway and provides input to the driver, categorizing circumstances as "safe" or "beware." This approach is different from existing vision-based driver intention systems, which often rely on computationally expensive behavior modeling and trajectory prediction, requiring large labeled datasets. Based on these existing gaps in the literature, the main contributions of this thesis are:

a) VBAPAS contributions:

1) It guides vehicles to the shortest path to available parking space and saves time.
2) It increases efficiency by monitoring parking occupancy in real time.
3) It is flexible to allow drivers to identify and select target parking slots.
4) It has a modular algorithm framework for integration with different object detection algorithms.

b) Driver intention algorithm contributions,

1) It has a simple design to minimize computational load.
2) The driver comfort solution improves user understanding and confidence.
3) Emphasis is on real-time spatial reasoning to provide immediate feedback.
4) It has comprehensive performance in a variety of traffic scenarios.

1.2. Structure of the Thesis

The rest of this thesis are listed below:

- In Chapter 2, general knowledge about smart transportation systems is detailed.
- In Chapter 3, technical information and literature review about image processing techniques, AI models and path planning algorithms are detailed.
- In Chapter 4, methods, advantages and technical details of the proposed algorithms are provided.
- In Chapter 5, evaluations about the test results and analyzes of the proposed algorithms in different test scenarios are detailed.
- In Chapter 6, the conclusion part is given.

The following section provides a comprehensive literature review of image processing and path-planning-based AI approaches for smart transportation systems.

2. REVIEW OF IMAGE PROCESSING AND PATH PLANNING BASED AI APPROACHES FOR SMART TRANSPORTATION SYSTEMS

STS is an innovative approach to modern transportation, which combines new technologies such as the Internet of Things (IoT), AI and data analytics to build more efficient, safer and sustainable transport networks. These systems use advanced communication and sensing technologies to manage traffic, reduce congestion and improve urban mobility. Real-time data collection and analysis are critical in STS which allows for dynamic traffic management, optimal routing and predictive infrastructure maintenance [12]. STS has the ability to connect vehicles, infrastructure and central control systems enabling the implementation of STS and automated transport services such as ridesharing, EV charging networks and autonomous public transport [13].

The development of smart transport can be traced back to the 1990s, when STS was introduced with the aim of modernizing transportation using technologies such as GPS and traffic monitoring systems. These systems have evolved over time in response to breakthroughs in AI and big data analytics, increasing their ability to predict traffic patterns, manage real-time

crises and improve overall road safety [14]. Furthermore, the growth of smart cities has increased the use of STS, which connect transport with other urban systems to enhance sustainable living and reduce environmental impact. As these technologies advance, they have the potential to significantly improve transportation efficiency while addressing the growing concerns of urbanization and climate change.

Image processing algorithms and AI models have become essential components of modern technology, transforming industries such as healthcare, security, transportation and entertainment. Image processing is the manipulation and analysis of visual data to extract meaningful information, a technique that dates back to the mid-twentieth century when early computers were developed. Over the years, advances in algorithms and technology have improved the ability to analyze and understand complex images, allowing for breakthroughs in fields such as medical imaging, facial recognition and autonomous systems [15]. AI models, particularly deep learning algorithms, have accelerated this progress by enabling machines to recognize patterns and make judgements from vast amounts of data without the need for human intervention [16].

The combination of image processing algorithms with AI has resulted in powerful solutions to challenges such as object identification, image categorization and pattern recognition, adding new capabilities to vision systems and computer-based technologies. AI models have evolved over time, starting with the concept of artificial neurons in the 1940s and moving to advanced deep learning frameworks such as Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs) [17]. These models have demonstrated human-level performance in a variety of visual tasks, paving the way for the use of vision technologies in automobiles, including autonomous driving and advanced driver-assistance systems (ADAS), which rely heavily on real-time image processing and AI. The continued advancement of these technologies is transforming businesses by enhancing safety, efficiency and decision-making capabilities in complex situations. The detailed literature review of the vision-based object detection algorithms for vehicles is presented in the following chapter.

2.1. Object Detection Algorithms in Vision Technologies for Vehicles

Vehicle vision technologies are critical to the development of self-driving and driver assistance systems. These technologies integrate cameras, sensors and AI algorithms, which

enable vehicles to detect and respond to their surroundings. Vision-based systems promote safe and efficient driving by using techniques such as image recognition, object identification, lane tracking and pedestrian detection. CNNs, which is a type of deep learning model, are used in vehicles for visual perception tasks such as obstacle detection, traffic sign recognition and lane departure alerts [18]. Furthermore, the addition of Laser Imaging Detection and Ranging (LiDAR), radar and stereo vision systems improves camera-based technologies by providing a more complete image of the vehicle surroundings, ensuring redundancy and better accuracy [19]. These technologies are critical not just for autonomous driving, but also for ADAS, which aims to reduce driver errors and increase road safety.

The development of vehicle vision technology has a long history, dating back to early computer vision and robotics research. Key innovations include the use of vision systems in efforts such as the Defense Advanced Research Projects Agency (DARPA) Urban Challenge, and improvements in deep learning that have significantly improved the performance of vehicle perception systems. These developments continue to improve, making vision-based technologies essential for the next generation of intelligent transport systems [20]. As autonomous driving research advances, the integration of AI and vision technologies has become a focal point in efforts to produce safer, more efficient vehicles capable of operating in dynamic, real-world environments.

Figure 1. Project vehicle of Prometheus [21].

The PROMETHEUS project, which stands for Program for European Traffic with Highest Efficiency and Unprecedented Safety, began in 1986. As shown in Figure 1, the prototype vehicle was developed in collaboration with more than 13 Original Equipment Manufacturers (OEM) from 19 different European countries. In 1987, a computer operated the test vehicle at speeds of up to 96 km/h on the Vamor highway in a 20-kilometer traffic-free zone. In this experiment, the vehicle was fully autonomous, with the camera system providing the necessary input information for both lateral and longitudinal control. The primary objective of one the project was to provide sufficient input information to drive the vehicle. Furthermore, a long-distance completely autonomous driving test was conducted in 1998, Odense, Denmark, which is 1,600 kilometers from Munich. It was found that about 95 percent of driving was done without driver intervention [22].

The PROMETHEUS project is conducting another autonomous driving experiment in a mobile laboratory environment. The vehicles were equipped with cameras, processors and displays to provide input to the driver. The core algorithms can recognize lanes, objects and simulate the external environment. On the other hand, obstacle detection requires stereo images rather than a three-dimensional model to identify empty space around the vehicle. The system which is called GOLD has been used on the ARGO, a Lancia Thema with automated steering. While research into smart vehicle development began in Japan in the 1970s, it became a significant investment in Europe throughout the 1980s and early 1990s, when prototypes of autonomous vehicles were built. With this application, associated research was initiated. In Japan, the project is known as the Personal Vehicle System and numerous companies have contributed and collaborated to pioneer work in this sector [22].

The Advanced Navigation Assisted Highway System Research Association was established in Japan in 1996. The Japanese Smart Way concept vehicle will incorporate driver assistance functions such as lane keeping, intersection collision avoidance and pedestrian detection. A model car with these features is scheduled to enter service in 2003 and to be introduced nationwide by 2015. In the United States, the National Automated Highway System Consortium (NAHSC) was established in 1995, and the smart vehicle effort began in 1997. Carnegie Mellon University Navlab group has a track record of developing advanced driver assistance systems and smart vehicle technology. The Navlab group has completed production of eleven serial vehicles. These vehicles include systems with significant features such as off-road vehicles, automated highways, anti-roll-off and anti-collision. In 1995, the Navlab5 series demonstrated that its vehicle could be driven on a highway from east to west coast using an

automated lateral control system. Over 5000 kilometers, 98% of the journey was completed without human intervention. The Navlab series' most recent serial production model is the Jeep Wrangler, which can identify obstacles at short and medium ranges. Ford and General Motors, two prominent automobile manufacturers, presented concept automobiles in this region. The United States of America, like the European Union, encourages research and investment in this area. The US Department of Transportation has initiated a $35 million effort with General Motors (GM) to develop rear collision avoidance systems [22].

The development of object detection is considered to have taken place in two historical periods over the last two decades: the classic object detection period before 2014 and the deep learning-based detection period after 2014 [23]. Algorithms are classified into four types: classical, deep learning-based, transformer-based and hybrid/specialized algorithms [24]. Classic methods include hand-crafted features and classic machine learning algorithms, which are efficient but less accurate and include techniques such as HOG + SVM. Deep Learning-Based Methods use convolutional neural networks to automatically learn feature representations, resulting in high accuracy and adaptability across several object categories. Examples include the R-CNN and YOLO families. Transformer-Based Methods leverage self-attention processes to characterize interactions between object instances, allowing for a more comprehensive and end-to-end detection technique that does not rely on traditional region suggestions, with models such as DETR leading the way. Hybrid and specialized algorithms integrate the capabilities of many architectures by iteratively enhancing object recommendations or integrating multi-level features, resulting in improved detection performance in challenging environments [23]. The following section provides details of four categories based on the primary object detection algorithms.

Table 1. Classification of the object detection algorithms.

Traditional Methods	Deep Learning Based Methods	Transformer-Based Methods	Hybrid and Specialized Algorithms
HOG + SVM Viola-Jones Detector DPM ACF	R-CNN Family YOLO SSD RetinaNet CenterNet EfficientDet M2Det	DETR YOLOS	Cascade R-CNN RefineDet CornerNet FPN SNIPER TridentNet

The following chapter provides the review of the traditional methods of object detection algorithms, outlining their underlying principles within the scope of the thesis.

2.1.1. Traditional Methods

Traditional methods typically involve extracting certain elements, such as edges, textures, or forms, which are thought to be important for object recognition. These features are then used to train a classifier to recognize or detect different objects. While these methods are more computationally efficient and interpretable than deep learning approaches, they are less adaptive and accurate, particularly when dealing with complex and diverse object appearances [25].

- **Histogram of Oriented Gradients with Support Vector Machine**

  Histogram of Oriented Gradients (HOG) is a feature descriptor that recognizes edge and gradient patterns in images, which are necessary for object detection. HOG features are extracted from an image and sent into a Support Vector Machine (SVM), a machine learning classifier that finds objects. This method is known for its effectiveness in pedestrian detection [26].

- **Viola-Jones Detector**

The Viola-Jones Detector was one of the first object detection systems and it is well-known for in real time face detection. It finds objects rapidly and effectively by integrating Haar-like features with an AdaBoost classifier. The system detects objects of various sizes using an image pyramid and a sliding window approach [27].

- **Deformable Part Models**

Deformable Part Models (DPMs) represent an object as a collection of components arranged in a flexible array. HOG feature describes each component, while a deformable structure defines their spatial interactions. This technique allows for differences in object appearance and deformation, making it ideal for recognizing objects of various forms and placements [28].

- **Aggregate Channel Features**

ACF is a remarkably effective object recognition algorithm that is widely used in computer vision tasks, particularly pedestrian and vehicle detection. ACF extracts low-level image features from a set of pre-processed image channels, such as gradient magnitude, gradient orientation histograms and color channels. These channels capture key visual information from the image, which ACF then integrates to produce a concise feature description. The basic concept of ACF is to extract features from an over-complete set of image channels and then feed these aggregate channel features into a boosted decision tree classifier, which is often done using AdaBoost [29]. This approach strikes a balance between accuracy and processing efficiency, making ACF excellent for real-time applications such as self-driving and surveillance systems.

When compared to previous algorithms such as HOG combined with SVM, ACF outperforms in terms of speed while maintaining similar detection accuracy. Furthermore, the ability of ACF to scale effectively across a wide range of object identification tasks has made it a popular choice in practical applications. Furthermore, ACF can be enhanced with modifications such as Locally Decorrelated Channel Features (LDCF) and it is often

combined with multiscale detection frameworks to improve robustness and precision across many scales and perspectives [29].

The following chapter provides the review of the deep learning-based methods of object detection algorithms, outlining their underlying principles within the scope of the thesis.

2.1.2. Deep Learning Based Methods

Deep learning object detection methods use neural networks to recognize and locate objects in images or video frames. These algorithms extract hierarchical features from images and use them to perform various recognition tasks. The key concept is to train a model to recognize patterns and structures at multiple levels of abstraction, allowing it to distinguish between different objects and their locations [30, 31]. These techniques typically include two main components: feature extraction and object classification/regression. Feature extraction involves passing the image through multiple layers of convolutional and pooling algorithms to get a set of high-level features. Object classification and regression involves interpreting these features to predict the existence, class and bounding box of objects [30, 31].

- **Region Based Convolutional Neural Networks**

  CNN is a type of deep neural network that can be rapidly applied to real-time videos or images to extract specific data. Because of its ability to handle many raw features, the CNN enables the processing of large amounts of data. Because of its deep learning properties related with feature reduction and selection, CNNs have outperformed humans in image recognition and identification tests. It can be classed as an extension of Multi-Layer Perceptron (MLP), a supervised neural network technique for data classification. The MLP has proven to be an excellent learning approach for object categorization and detection. Thus, the MLP is the last component of the CNN algorithm. The CNN contains several hidden layers that are constructed in blocks, such as CONV, ReLu and POOL, in order to minimize the dimension of the input data prior to MLP-based classification [32].

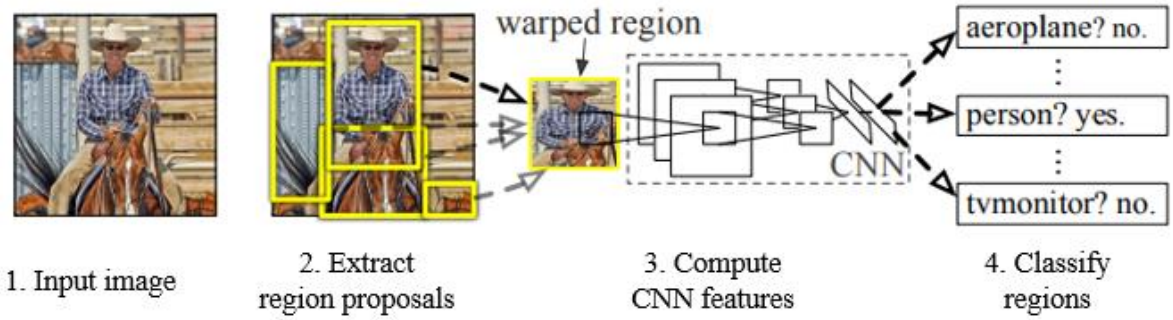


Figure 2. Steps of R-CNN architecture [88].

- **Region Based CNN**

OverFeat, a model developed by the researchers, combines a sliding window method with CNN. However, this method was inefficient due to the windows having to be placed in various locations of the network. Ross Girshick presented a Region-Based CNN (R-CNN) structure. The intended R-CNN operated in three phases. To begin, the range proposal approach was used to differentiate typical objects from the entire image. Then, features were extracted from each object that could be detected in the scene. Finally, the image was divided into regions and categorized using an SVM. Although the results were high accurate, training the network with the available datasets proved problematic [33].

The R-CNN methods face two problems. One of the problems is that the algorithm is not informed about the number of classes in the input images. Also, object identification may be inconsistent, leading to misclassification. The user can select a precise location to identify the object in the image, but other important things may be ignored. This problem can be solved by scanning the entire image, although processing performance would suffer significantly in this scenario. This speed problem has been solved by using selective search, which is a sliding window approach that can emphasize locations where objects can be found in images. The sliding window consists of many windows with varied speeds to detect objects of different sizes and capturing them from different angles. R-CNN can create many region suggestions using the sliding approach and consolidate similar areas into a single region. R-CNN can be directly applied to the remaining regions because the identified regions are ready for classification. Then, CNN models such as AlexNet, Visual Geometry Group (VGG) and MobileNet can be applied to each region. Finally, regression algorithms are used to determine the bounding box coordinates for each element [32].

- **FR-CNN**

The FR-CNN is an improved version of the R-CNN based on constraints. The FR-CNN network receives an entire image and a set of object recommendations as input. The network produces a map of convolutional features by analyzing the entire image using a sequence of convolutional and maximum pooling layers. A region of interest pooling layer takes a fixed-length feature vector from the feature map for each object suggestion [34].

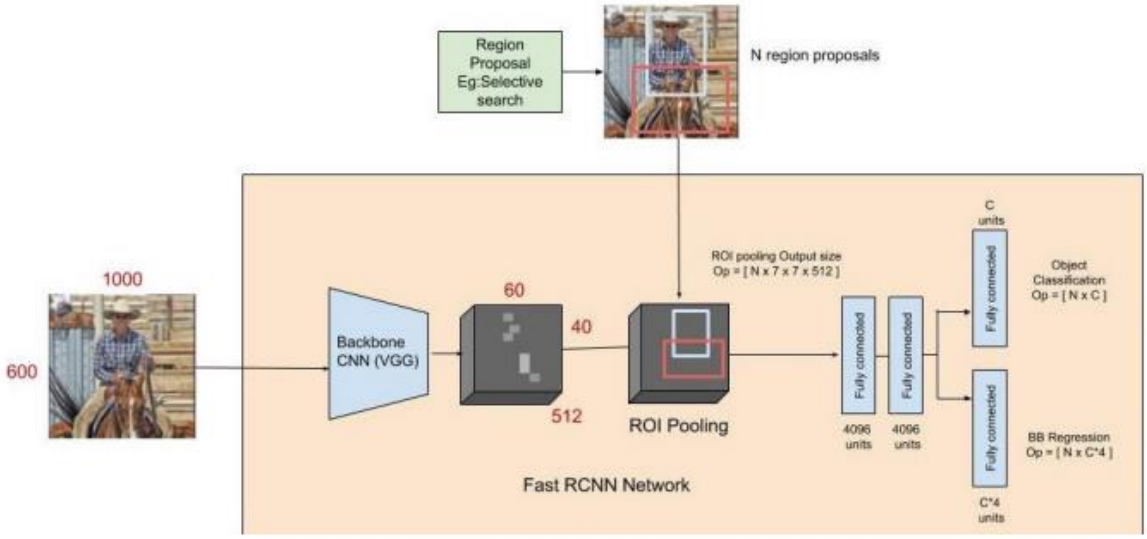


Figure 3. Visualization of the fast R-CNN [89].

Fast R-CNN improves R-CNN by processing all image and object proposals simultaneously. A convolutional network creates a feature map, and a ROI pooling layer extracts a fixed-length feature vector for each suggestion, increasing detection efficiency and accuracy.

- **YOLO**

You Only Look Once (YOLO), developed by Joseph Redmon in 2016, transformed real-time object identification by detecting everything in a single pass. Each subsequent version, including YOLOv2, YOLOv3 and YOLOv4, improved the speed, accuracy and handling of a wide range of object sizes. Later modifications by other developers improved YOLO for use on a wide range of devices, making it indispensable in fields such as autonomous driving and robotics [35].

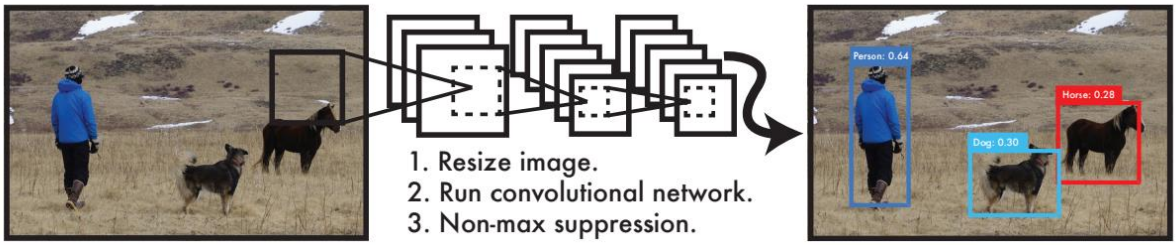


Figure 4. Visualization of the YOLO System [35].

YOLO processes an image in a single network pass, resulting in fast performance. It divides the image into grid cells and estimates their bounding boxes and class probabilities as shown in Figure 4. The main advantage of YOLO is speed, as it detects objects in a single forward pass of the network, as opposed to region-based techniques, which assess many regions separately [37]. The other versions of YOLO can be summarized as follows,

1. YOLOv1, provided a unified model for object identification and classification, but its coarse 7x7 grid structure limited its ability to identify small and overlapping objects [35].
2. YOLOv2, improved accuracy and variety by incorporating Darknet-19, anchor boxes, multi-scale detection and cooperative training, allowing for the recognition of over 9000 types of objects [36].
3. YOLOv3, improved object detection with multi-scale prediction, residual blocks and logistic regression, increasing both accuracy and robustness [37].
4. YOLOv4, improved performance by combining CSPDarknet53 for feature extraction, PANet for feature aggregation and data augmentation, resulting in excellent accuracy while maintaining real-time efficiency [38].
5. YOLOv5, developed by Redmon J., a one-stage object detection method that predicts bounding boxes and classes in a single regression step. Images are divided into S×S cells and results are refined using Non-Maximum Suppression (NMS) [39].
6. YOLOv6, developed by the Meituan team to improve real-time object recognition by strengthening the backbone of network and neck, improving speed and accuracy with advanced feature extraction and integration algorithms and providing a more

efficient and accurate solution than previous versions such as YOLOv4 and YOLOv5 [40].

7. YOLOv7, developed in collaboration with Chien-Yao Wang, Alexey Bochkovskiy and Hong-Yuan Mark Liao, includes new methods and enhancements to improve object detection speed [41].
8. YOLOv8, developed by the Ultralytics team, represents the latest advances in the YOLO series, with significant improvements in accuracy, speed and usability. This version includes new architectural elements and optimizations aimed at increasing performance in object identification tasks [42].

- **Single Shot MultiBox Detector**

Wei Liu et al. developed the Single Shot MultiBox Detector (SSD) in 2016 as a faster alternative to region-based detectors while maintaining competitive accuracy. It detects objects in a single forward pass by estimating bounding boxes and class scores based on numerous feature maps of different widths, as shown in Figure 5.

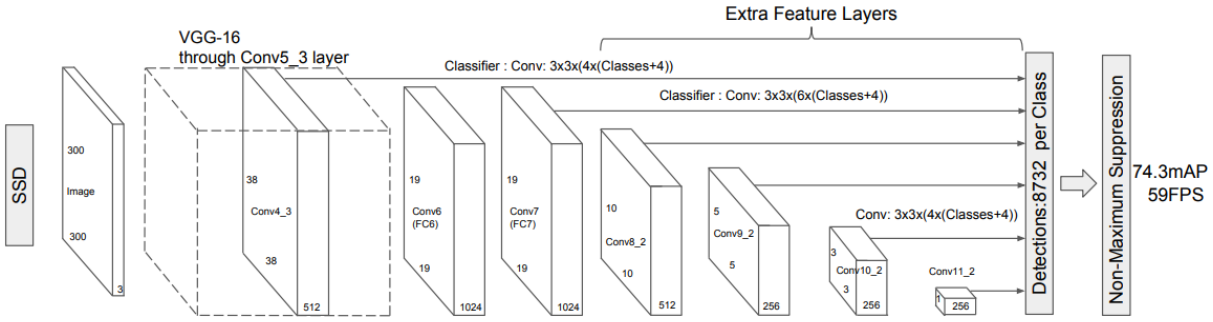


Figure 5. Visualization of the SSD [43].

SSD can detect objects of specific sizes using bounding boxes at several aspect ratios and scales by extracting feature maps at different scales using a sequence of convolutional layers. Non-maximum suppression is used to remove redundant bounding boxes, leaving only the highest scoring detections. The main benefit of SSD is the fast detection speed, which makes it ideal for real-time applications and it strikes a good balance between speed

and accuracy. However, SSD can have difficulty detecting small objects and has lower accuracy when compared to two-stage detectors such as Faster R-CNN [43].

- **RetinaNet**

RetinaNet, developed by Facebook AI Research (FAIR) in 2017, solves class imbalance in dense object detectors by introducing the focus loss function. This function reduces the influence of easily categorized background instances during training, allowing the model to focus on more challenging foreground objects.

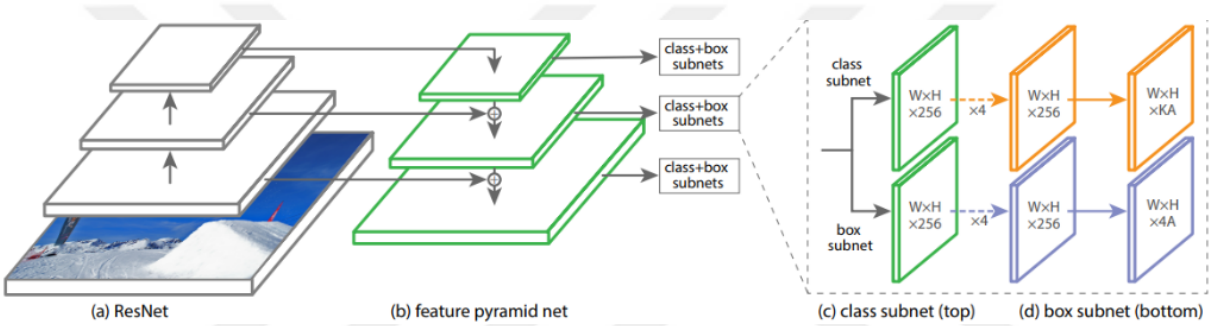


Figure 6. Architecture of the RetinaNet model [44].

RetinaNet uses a one-stage detection architecture with a Feature Pyramid Network (FPN) for effective multi-scale feature extraction, resulting in improved object recognition at different sizes. As shown in Figure 6, while RetinaNet outperforms previous single-stage detectors and approaches the performance of two-stage models such as the faster Faster R-CNN, it is typically slower than SSD due to its more complicated network architecture and the additional computational cost associated with the focal loss function [44].

- **CenterNet**

CenterNet, developed in 2019 by Xingyi Zhou et al., extends the concept of key point-based object identification by providing an anchor-free approach. Instead of traditional bounding boxes, CenterNet predicts the center points of objects using heat maps and then regresses the width and height of the bounding boxes based on these points. The network architecture includes a backbone for feature extraction, a heat map prediction layer to detect

object centers and an offset layer to fine-tune center localization. This anchor-free approach simplifies object detection by eliminating the complexity of anchor boxes, resulting in a more efficient detection process. CenterNet achieves competitive accuracy with anchor-based techniques while reducing computational complexity and hyper parameters. However, its application may be limited for highly small objects or those with unusual geometries, where traditional anchor-based techniques may perform better [45].

- **EfficientDet**

Mingxing Tan, Ruoming Pang and Quoc V. Le of Google Research developed EfficientDet in 2020 with the goal of improving both the architecture and feature fusion in object detection. It uses compound scaling and Neural Architecture Search (NAS) to find a tradeoff between accuracy and efficiency. EfficientDet uses EfficientNet as a backbone for feature extraction and incorporates a Bidirectional Feature Pyramid Network (BiFPN) to successfully fuse multiscale feature maps, as shown in Figure 7.

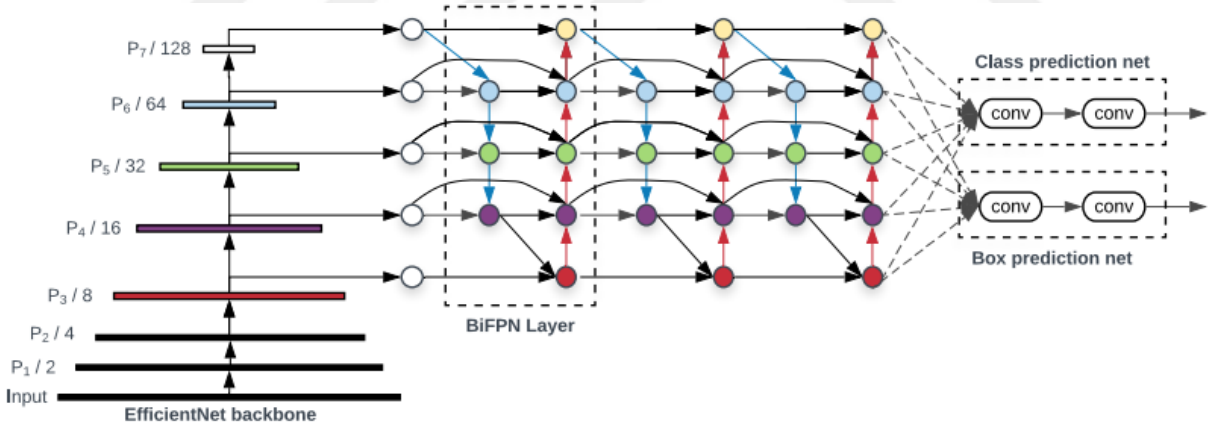


Figure 7. Architecture of the RetinaNet model [46].

The model scales up by simultaneously adjusting the resolution, depth and width of the backbone network and BiFPN layers, improving performance across a wide range of computational budgets. EfficientDet generates state of the art results with fewer parameters and lower computational cost than other detectors, making it ideal for resource-constrained environments such as mobile devices and high-performance servers. However, its reliance

on NAS complicates the design process and drives up computational costs throughout development [46].

- **Multi-level Feature Pyramid Network**

Qijie Zhao et al. developed M2Det in 2018 to improve feature representation through multi-level pyramids. Their architecture uses a U-shaped module to form a multi-level feature pyramid, which extracts features of varying sizes. These features are then used with a collection of detection heads to predict bounding boxes and class scores. This architecture allows M2Det to capture detailed information at multiple scales, increasing its ability to between small and large objects in complex environments. By emphasizing multi-level feature extraction, the network is better adapted to handle objects of different sizes.

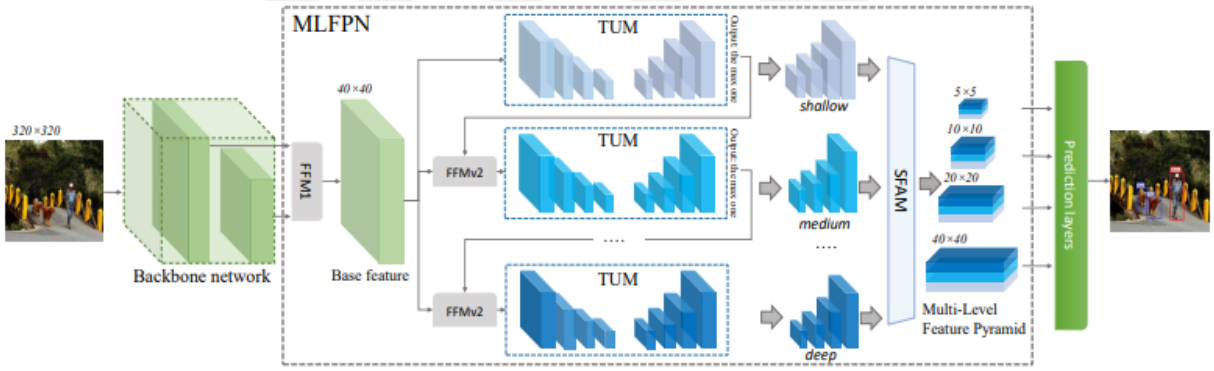


Figure 8. Overview of Multi-level Feature Pyramid Network [47].

M2Det achieves exceptional accuracy, especially when identifying small objects and strikes an ideal balance between speed and performance. However, it is more computationally expensive than simpler technologies such as SSD and the complexity of its multi-level feature pyramid requires precise tuning for optimal performance as shown in Figure 8 [47].

The following chapter provides the review of the transformer-based methods of object detection algorithms, outlining their underlying principles within the scope of the thesis.

2.1.3. Transformer-Based Methods

Transformer-based techniques have had a significant impact on object recognition by providing a novel framework that differs from traditional CNN models. These methods use the self-attention mechanism of transformers, which has proven highly effective in Natural Language Processing (NLP), to capture long-range relationships and global context within images. Transformer-based algorithms approach object recognition as a direct set prediction problem, unlike CNN-based methods that rely on prior features such as anchor boxes or region suggestions. This self-attention allows the model to focus on multiple areas of an image simultaneously, increasing its ability to distinguish objects in complex and chaotic situations [48, 49].

Transformer-based models are efficient at capturing global visual context and managing many objects, but they require large datasets and significant computational resources to train. However, their flexibility and scalability have led to cutting-edge performance in a variety of vision tasks, indicating the potential to revolutionize the future of object recognition. These models represent a significant step forward in applying deep learning to increasingly difficult, real-world vision applications [48, 49].

- **Detection Transformer**

  Detection Transformer (DETR) is a pioneering transformer-based object detection system developed by FAIR in 2020. Unlike standard object detection algorithms, which use CNNs and sophisticated post-processing techniques such as anchor creation and non-maximum suppression, DETR recasts object detection as a straight set prediction issue. It uses an encoder-decoder transformer architecture, in which the encoder evaluates image information while the decoder predicts bounding boxes and class labels as shown in Figure 9.

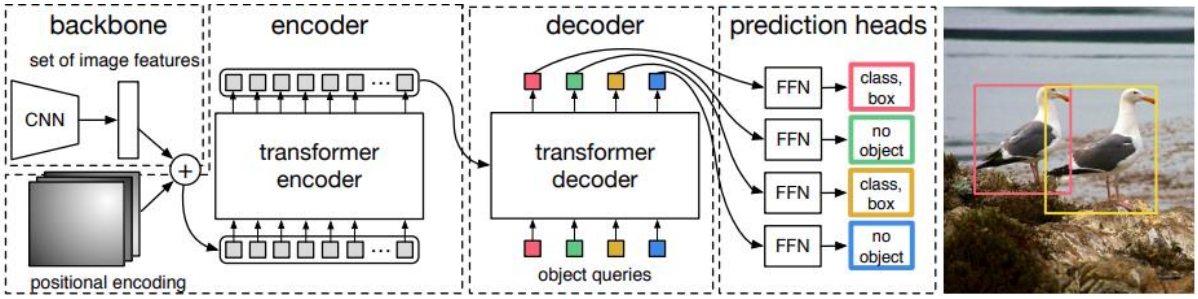


Figure 9. Architecture of the Detection Transformer [49].

This technique uses self-attention processes to capture global context, allowing DETR to recognize objects even in crowded environments. DETR eliminates the need for anchor boxes and region suggestions, which simplifies the detection process. However, it requires more data and takes longer to train than CNN-based detectors. The transformer-based design of DETR improves the handling of complicated spatial relations, however training inefficiency and delayed convergence are often cited as problems. It was created by Nicolas Carion and his colleagues, and it represented a significant step towards the transformer models for object identification [49].

- **You Only Look at One Sequence**

The Hugging Face team created YOLOS, a transformer-based object recognition model, in 2021. Inspired by the success of Vision Transformers (ViTs) in image classification, YOLOS applies the transformer design to object recognition as a sequence modeling problem.

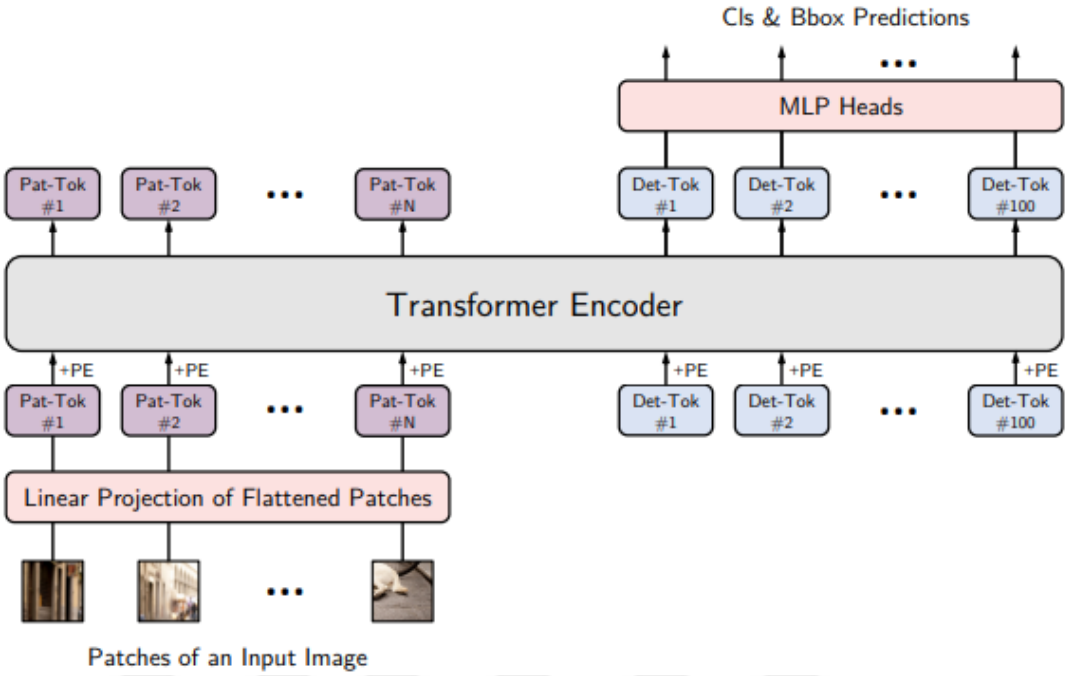


Figure 10. Visualization of the YOLOS [50].

The input image is divided into fixed size patches, which are embedded and sent to the transformer as a series of tokens, similar to how transformers process text as shown in Figure 10. YOLOS does not use typical CNN-based architectures and instead applies the transformer directly to the series of image patches. While YOLOS has the advantage of simplicity and benefits from the scalability of transformers, it shares some of the constraints of early transformer-based approaches such as DETR, including the need for large datasets and substantial computational resources to train. YOLOS proved that transformers can be used well for object detection without relying on the classic grid-based technique, however their performance on smaller datasets and real-time applications may be less competitive [50].

The following chapter provides the review of the hybrid and specialized methods of object detection algorithms, outlining their underlying principles within the scope of the thesis.

2.1.4. Hybrid and Specialized Methods

Hybrid and specialized algorithms make important contributions to object recognition by combining the strengths of classical machine learning and modern deep learning approaches.

They can enhance detection processes by combining feature-based and data-driven techniques, improving accuracy and robustness in a variety of situations. Specialized algorithms focus on specific tasks or types of data, tailoring their designs to increase effectiveness in specific deployment scenarios, such as identifying tiny objects or in limited operational environments. This continued advancement represents a trend toward more adaptable and efficient object detection systems for use in autonomous driving, surveillance and medical imaging. Recent research has demonstrated the effectiveness of these strategies, emphasizing their ability to improve upon current detection methods [51, 52].

- **Cascade R-CNN**

Cascade R-CNN, developed by Zhaowei Cai and Nuno Vasconcelos in 2018, improves object identification by addressing the problem of overfitting in multistage object detectors.

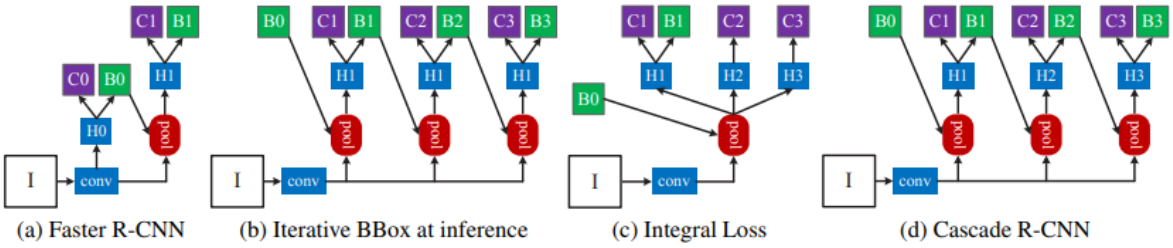


Figure 11. Comparison of Faster R-CNN, Iterative BBox, Integral Loss and Cascade R-CNN [53].

It improves the quality of bounding boxes through a series of detection steps, each with a higher Intersection over Union (IoU) threshold. By cascading many steps, the model gradually improves object localization and classification accuracy. While this technique greatly improves detection quality at high IoU thresholds, its multistage structure results in greater computational complexity and longer inference times when compared to a single-stage detector [53, 54].

- **RefineDet**

Shifeng Zhang et al. developed RefineDet in 2018, a hybrid object detection method that combines the advantages of one-stage and two-stage detectors. It works in two steps: first, it refines anchor boxes by removing obvious negative instances and then it accurately localizes and classifies objects.

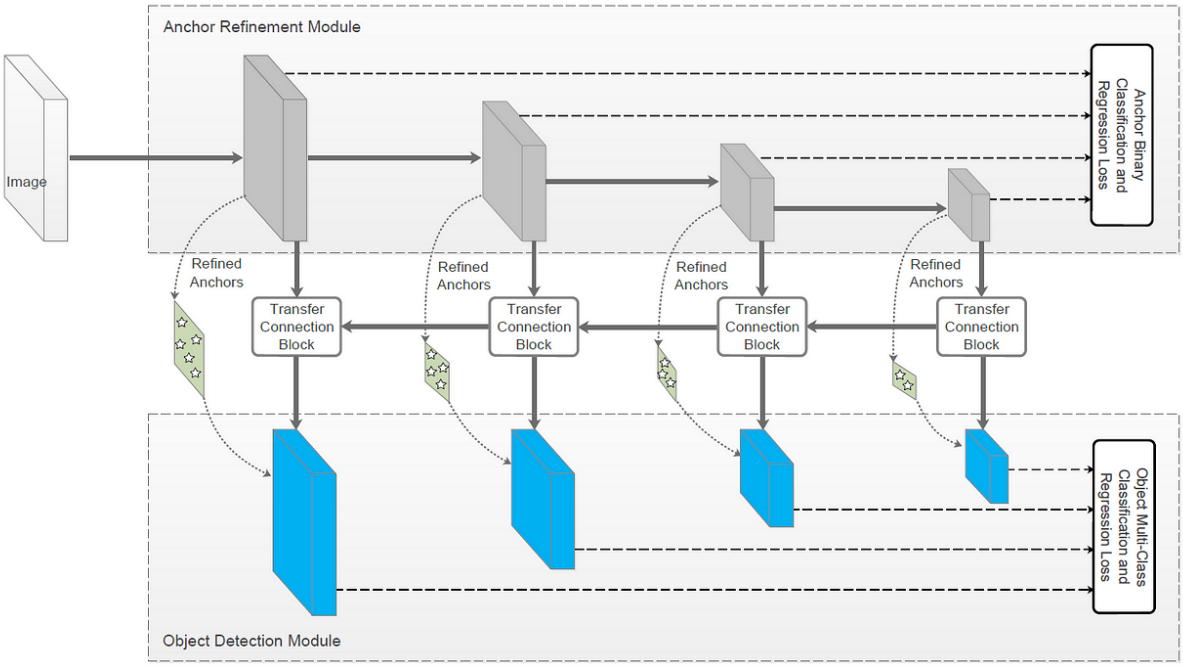


Figure 12. Visualization of the RefineDet [55].

This approach balances speed and accuracy, outperforming standard single-stage detectors while being efficient for real-time applications as shown in Figure 12. However, due to the limitations of anchor-based detection, RefineDet may have difficulty detecting small objects [55].

- **CornerNet**

CornerNet, developed by Hei Law and Jia Deng in 2018, is an anchor-free object identification approach that uses the corners of bounding boxes, specifically the top-left and bottom-right corners, to find objects.

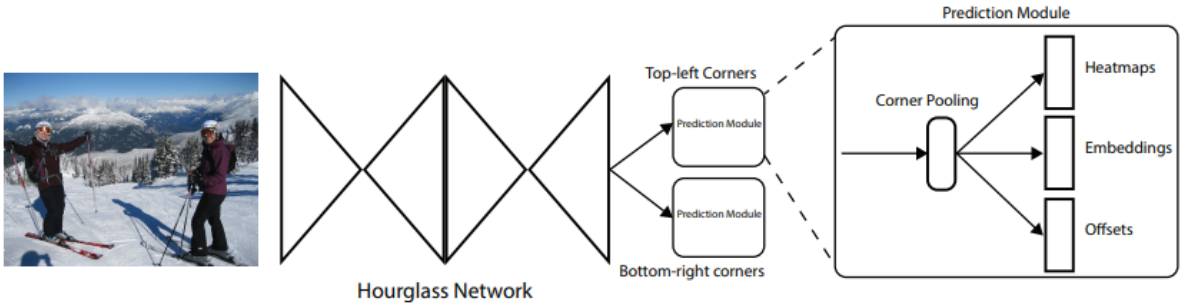


Figure 13. Visualization of the CornerNet [56].

CornerNet simplifies and increases detection accuracy by eliminating the requirement for specified anchor boxes, especially for small objects. However, corner pairing can be computationally expensive in complicated or crowded environments, slowing down detection performance. Its anchor-free design also makes it more intuitive than previous techniques [56, 57].

- **Feature Pyramid Networks**

Tsung-Yi Lin et al. developed FPN in 2017 to improve object detection by building a multiscale feature pyramid that combines low-level spatial properties with high-level semantic data. FPN uses a top-down design that incorporates features of different sizes, allowing for better detection of both small and large objects as shown in Figure 14.

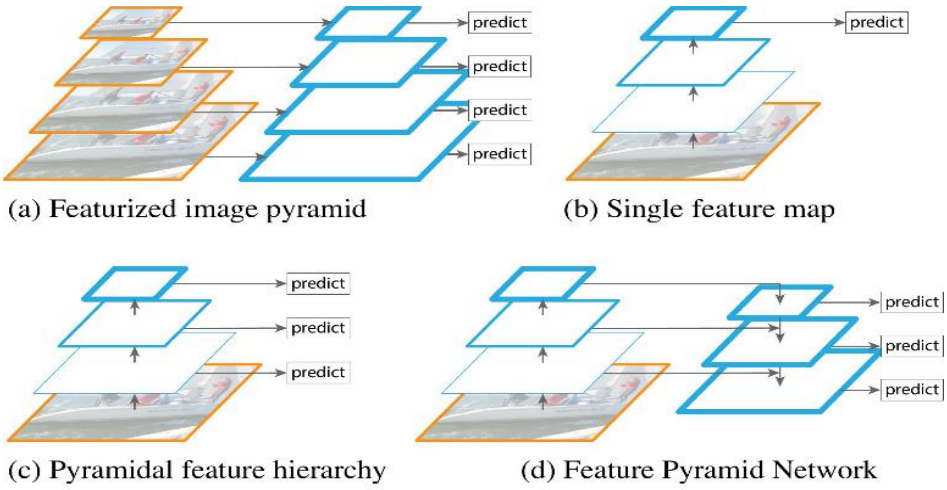


Figure 14. Architecture of Feature Pyramid Networks [58].

This technique improves accuracy over a wide range of object sizes, but it also increases computational cost due to its multi-scale feature fusion procedure, making it less efficient than simpler detection algorithms [58].

- **Scale Normalization for Image Pyramids with Efficient Resampling**

SNIPER, developed in 2018 by Bharat Singh and Larry S. Davis, addresses the challenge of detecting objects of different sizes by selectively processing regions of interest within images. Instead of resampling the entire image at different sizes, SNIPER examines only the region most likely to contain objects, resulting in significant computational savings, as shown in Figure 15.

Figure 15. Region selection example of SNIPER [59].

This method is ideal for large-scale object identification tasks because it provides high accuracy while using minimum computational resources. However, selective resampling can still be resource intensive for larger datasets [59].

- **TridentNet**

TridentNet, developed by Yanghao Li et al. in 2019, improves multiscale object detection by merging parallel convolutional layers with different dilation rates. These layers allow the network to detect objects of different sizes in a single forward pass, eliminating the need for traditional feature pyramids as shown in Figure 16.

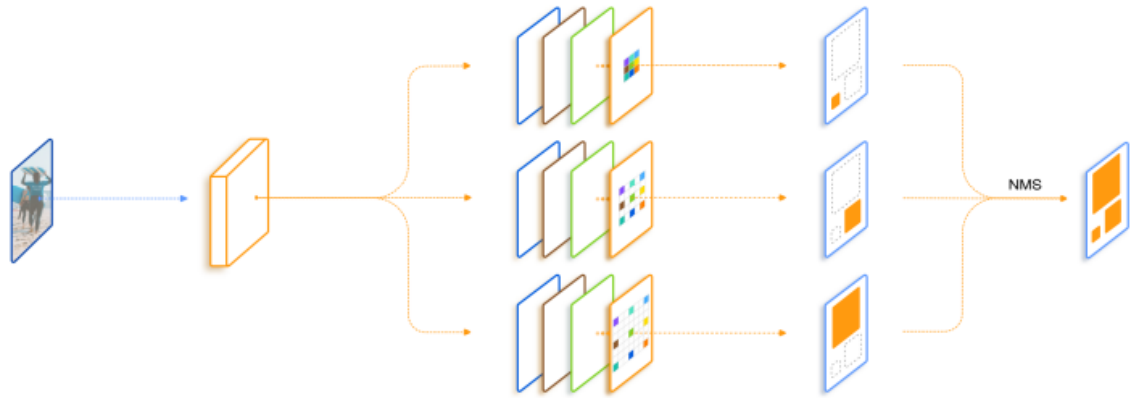


Figure 16. Visualization of the TridentNet [60].

TridentNet increases accuracy for both small and large objects while being computationally efficient. However, the use of dilated convolutions can slow down inference in some scenarios, particularly when processing complex scenes [60].

Table 2. Comparison table of the object detection algorithms.

Category	Algorithm	Key Features	Strengths	Weaknesses
Traditional Methods	**HOG + SVM**	HOG features, SVM classifier	Robust in controlled setups	Limited generalization, slow
	Viola-Jones	Haar features, cascade classifier	Fast in face detection	High false positives
	DPM	Deformable parts model	High accuracy	Slow, computationally expensive
	ACF	Aggregated channel features	Efficient	Lower accuracy

Deep Learning	**R-CNN Family**	Region proposals, CNN-based	High accuracy	Slow, computationally expensive
	YOLO Family	One-stage detection	Fast, real-time detection	Lower accuracy for small objects
	SSD	Single-shot detection	Fast	Lower accuracy
	RetinaNet	Focal loss for class imbalance	Good for small objects	Slower than YOLO
	EfficientDet	Efficient multi-scale detection	Fast, efficient	Complex training
Transformer-Based	**DETR**	Transformer-based detection	No post-processing needed	Slow training
	YOLOS	Transformer sequence prediction	High accuracy	High computational cost
Hybrid & Specialized	**Cascade R-CNN**	Multi-stage detection	High accuracy	Slow, computationally expensive
	FPN	Multi-scale feature pyramid	Efficient multi-scale	Limited in dense objects
	TridentNet	Multi-branch scale detection	Accurate for varying scales	Heavy, slow

The following chapter provides the review of the path planning algorithms, including their working principles and strengths.

2.2. Path Planning Algorithms

Path planning algorithms have been extensively investigated and applied to autonomous driving problems such as minimum time and minimum distance in dense maps. The research focuses on transforming road network images in pixel coordinates to a global coordinate system, which allows autonomous vehicles to navigate in real time. To find the shortest path, a hybrid technique is used that combines the Dijkstra's algorithm and a greedy BFS algorithm [61]. Path planning algorithms can be divided into 4 main categories as detailed in the following table.

Table 3. Classification of the path planning algorithms.

Graph Traversal Algorithms	Shortest Path Algorithms	All-Pairs Shortest Path Algorithms	Advanced Pathfinding Techniques
DFS (Depth-First Search) BFS (Breadth-First Search)	Dijkstra Algorithm A* Algorithm Bellman-Ford Algorithm ALT (Landmark-Based Algorithm)	Floyd-Warshall Algorithm	Contraction Hierarchies

The following chapter provides the review of the graph traversal algorithms of path planning algorithms, outlining their underlying principles within the scope of the thesis.

2.2.1. Graph Traversal Algorithms

Graph Traversal Algorithms (GTA) locate a given vertex in a graph by traversing all available parking spaces during the search phase [62]. There are two common path planning algorithms which are Breadth First Search (BFS) and Depth First Search (DFS) [63].

- **Depth First Search**

DFS is a graph traversal technique that explores all vertices in a graph. The procedure starts at a given vertex and proceeds down each branch as far as possible before returning. Recursion or an explicit stack data structure are also possible implementation approaches. In the recursive method, the call stack acts as a stack data structure, allowing systematic exploration of the graph vertices. The DFS method can also be used to solve a variety of problems, such as finding related components, detecting cycles, and traversing a graph. It is important to note that the DFS does not always find the shortest path between two nodes in a weighted graph, as it may get stuck examining a long path before discovering a shorter one as shown in Figure 17 [64].

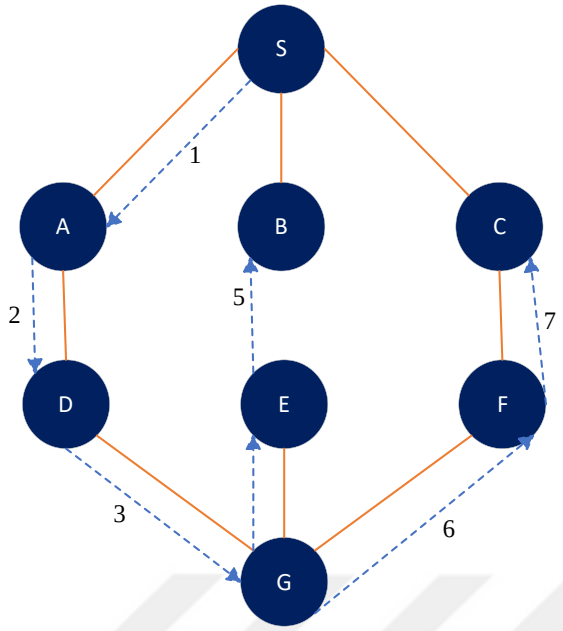


Figure 17. The DFS working principle.

The DFS works in several steps. First, a vertex is marked as visited and added to the result set. Then, the DFS is applied recursively to all unvisited neighbors of the current vertex. If there are no unvisited neighbors, the algorithm backtracks to the previous vertex along the path and continues the process until all vertices are explored.

- **Breadth First Search**

The BFS method searches a tree for nodes that meet certain requirements, such as the shortest path. It starts at the root of the tree; the algorithm evaluates all nodes at the current depth level before going on to the next nodes in the depth level. This method can be used for a variety of graph theory issues [65]. Counting the number of edges within the path, as shown in Figure 18, can help determine the shortest path between two vertices A and B.

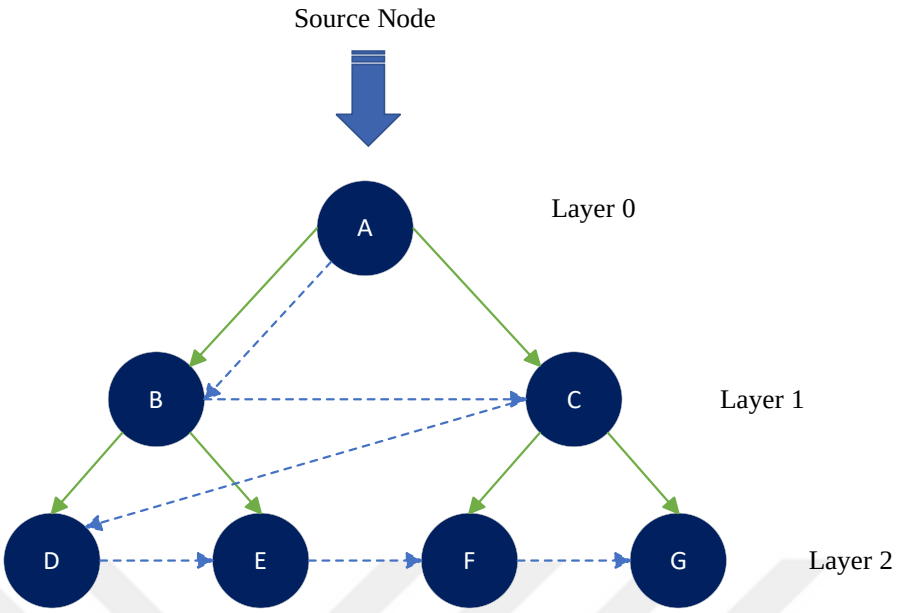


Figure 18. BFS algorithm schematic view.

The working principle of BFS starts with the source node, which is the starting point of the algorithm. The source node is added to the visited list and moved to the front of the queue. The program then examines the nodes adjacent to the current vertex and adds them to the visited list. After processing, the current node is dequeued. This method is performed iteratively until the queue is empty, ensuring that each node is explored layer by layer.

Key features of the BFS are:

- It is a simple and reliable architecture.
- It evaluates the network nodes and finds the shortest path between them.
- It is capable of traversing a graph with as few iterations as feasible.
- Iterations are smooth.
- It is impossible to get stuck in an infinite loop.
- Compared to other algorithms, it produces highly accurate results.

The following chapter provides the review of the shortest path algorithms of path planning algorithms, outlining their underlying principles within the scope of the thesis.

2.2.2. Shortest Path Algorithms

Shortest Path Algorithms are essential algorithms in graph theory and computer science for finding the most efficient path between two points in a network. These algorithms have many applications, including navigation systems, computer networking and transportation planning. Several well-known algorithms, including the Dijkstra's Algorithm, Bellman-Ford and A*, are designed to solve the shortest path problem by minimizing the total distance or cost associated with linking nodes in a graph. Each of these techniques has unique tradeoffs in terms of time complexity, accuracy and application to certain types of graphs, whether weighted or unweighted, directed or undirected [66].

- **Dijkstra's Algorithm**

The Dijkstra's algorithm, which was developed in 1956 by Dutch computer scientist Edsger Dijkstra [67], is widely used to find the shortest paths. Dijkstra's algorithm addresses the single-source shortest path problem in networks with non-negative edge path costs [68]. If the graph has negative edges, the path cannot be followed and may yield inaccurate results [69]. This technique selects the nearest point with the lowest weight [70], where the weights represent the value of each edge [71]. The Dijkstra's algorithm, a variation of the greedy algorithm, processes each node in the graph once, with the number of nodes affecting the rate of finding a solution. The method finds the shortest path with the best results, but it takes longer [72].

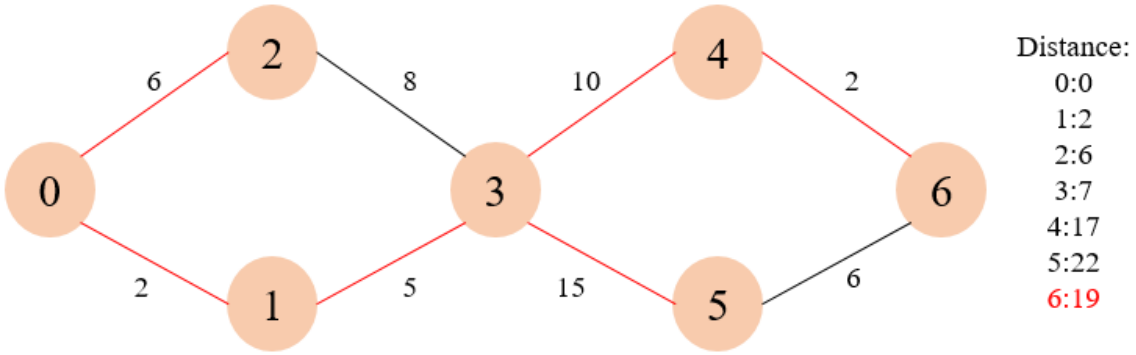


Figure 19. Working principle of Dijkstra's Algorithm.

The Dijkstra's algorithm begins by setting the distance to the source node to zero and the distances to all other nodes to infinity as shown in Figure 19. The algorithm uses a priority queue, which is commonly implemented as a min-heap, to manage node processing based on the current shortest distances. At each step, the unvisited node with the shortest distance is selected and designated as visited. The algorithm then changes the distances between its nearby nodes, moving any shorter paths into the priority queue for further consideration. This process is repeated until the priority queue is empty, ensuring that all nodes have been processed. Upon completion, the final distance values for each node are used to determine the shortest path.

Key features of the Dijkstra's Algorithm are:

- It computes the shortest path from a source node in a weighted network with non-negative weights.
- For efficient node selection, a greedy technique is used in conjunction with a priority queue (min-heap).
- It updates the shortest known paths using a relaxation process.
- It has a time complexity of $O(VlogV + E)$ and ensures optimal paths from a single source.

where V represents the number of vertices/nodes in the graph, E represents the number of edges in the graph and $logV$ which is a term that arises from the use of a priority queue (typically a binary heap) to manage the vertices.

- **A* Algorithm**

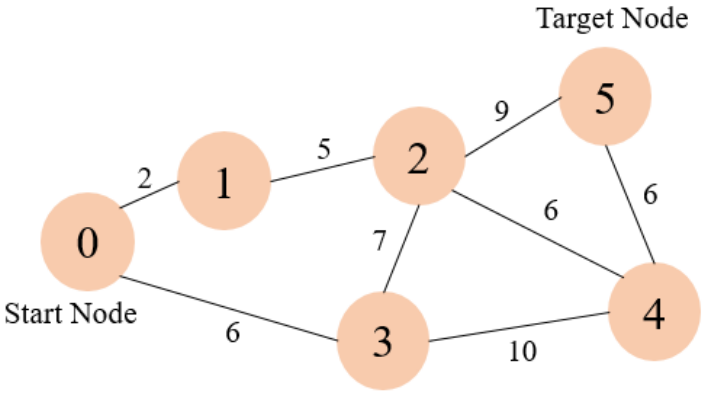
The A* method is a well-known path planning technique that can be used in both metric and topological configuration spaces [73]. This algorithm combines heuristic and shortest path searches. The A* algorithm is characterized as a best-first algorithm, since each cell in the configuration space is evaluated by the value:

$$f(v) = h(v) + g(v) \#(1)$$

where $h(v)$ is the heuristic distance of cell (Manhattan, Euclidean, or Chebyshev) to the target state and $g(v)$ is the length of the path from the beginning state to the goal state using the

chosen cell sequence. Obviously, this sequence terminates with the assessed cell. The value $f(v)$ is used to compare each neighboring cell to the target cell. The cell with the lowest value of $f(v)$ is selected as the next in the series. This approach has the benefit of being able to use, alter, or introduce new distances as a criterion. This allows for a wide range of modifications to the core assumption. The function $f(v)$ can include variables such as time, energy consumption and safety.

The A* algorithm, a path planning approach for mobile robots, has been successfully constructed and tested. The results can be found here. However, the results were not entirely satisfactory or positive. The computation time was quite long. Calculating the map with 60,000 cells takes more than an hour on an average computer. It is quite disappointing to apply this strategy to the robot. As a result, more research was done one possible modification to the A* algorithm [74].



Node	h(n)
0	20
1	16
2	6
3	10
4	4
5	0

Figure 20. Working principle of A* Algorithm

The A* technique estimates the most efficient path to a destination node using a combination of cost tracking and heuristic evaluation. It determines the cost of reaching the target from the current node by adding the cost from the starting node to the current node and the predicted cost from the current node to the destination as shown in Figure 20. Nodes are prioritized for research depending on their total expected cost, guaranteeing a thorough search. The method maintains two lists: an open list, which contains nodes that have yet to be investigated and a closed list, which contains nodes that have been handled. The process

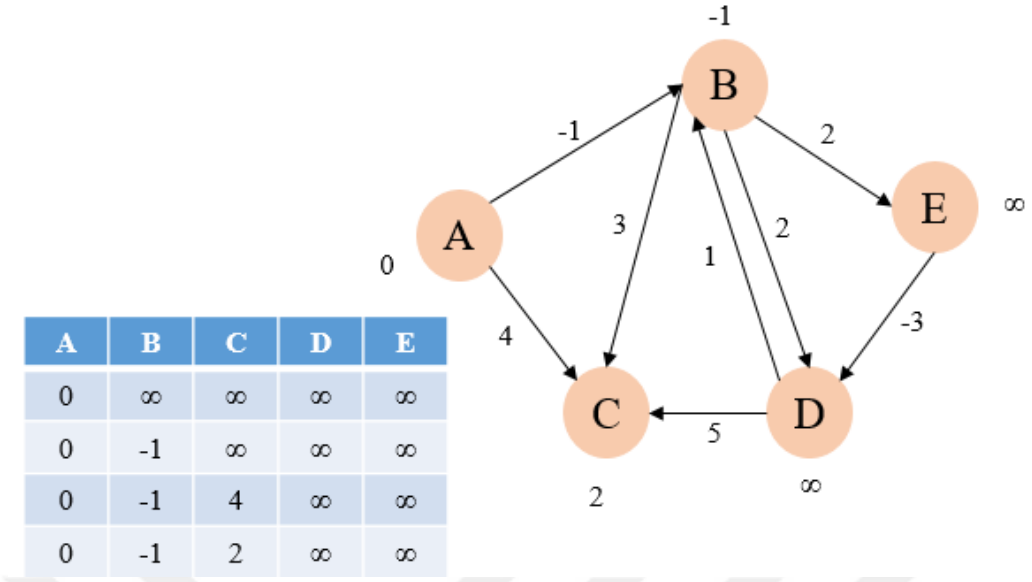
continues until the target node is reached, after which the algorithm reconstructs the shortest path.

Key features of the A* Algorithm are:

- It estimates the cost of reaching the goal.
- It tracks the actual cost from the starting node.
- It prioritizes nodes using both cost and heuristics.
- It manages nodes according to their total expected cost.
- Nodes have not yet been analyzed.
- Nodes have been processed.
- It recreates the path from start to finish after reaching the target.
- It uses an acceptable heuristic to find the shortest path.
- It searches for a path if one exists.

- **Bellman-Ford Algorithm**

Richard Bellman and Lester R. Ford Jr. developed the Bellman-Ford algorithm, a graph traversal tool, in the 1950s. It is designed to find the shortest paths between a single source vertex and all other vertices in a graph, even if negative edge weights are present. Unlike Dijkstra's technique, which fails with negative weights, Bellman-Ford can handle such situations and identify negative weight cycles with a total weight less than zero. [75, 76]. Fortunately, there is a more forgiving way to deal with negative edges: the Bellman-Ford algorithm. That is why a graph can include cycles with negative weights, resulting in a huge number of paths from the starting point to the goal, with each cycle reducing the length of the shortest path. Algorithm is a Dynamic Programming algorithm that determines the shortest path. It evaluates the structure of the graph and repeatedly improves each solution depending on the previous one until it finds the best response. Bellman-Ford can easily handle negative weights since it uses the entire graph to improve a solution [77].



A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞

Figure 21. Working principle of Bellman-Ford Algorithm.

The Bellman-Ford Algorithm begins by setting the distance to the source vertex to zero and the distances to all other vertices to infinite as shown in Figure 21. Edge relaxation is performed repeatedly $V - 1$ times, where V is representing the number of vertices in the graph. During each cycle, each edge is examined to see if there is a shorter path to a vertex and if so, the distance is changed. To find negative weight cycles, the system goes through another relaxation phase; if any distance can still be updated, a negative weight cycle is identified. If no negative cycles are found, the final distances are the shortest path between the source vertex and all other vertices.

Key features of the Bellman-Ford Algorithm are:

- It calculates shortest paths with negative edge weights.
- It identifies cycles with negative total edge weights, suggesting an infinite decrease in route cost.
- It finds the shortest paths between one source vertex and all other vertices in the graph.
- Distances are updated by relaxing edges for V-1 times.
- It is applicable to both types of graphs, assuming they are weighted.
- It is less efficient than other methods, such as Dijkstra's, especially for graphs with many edges.
- It uses a step-by-step approach to refine shortest path estimations.

- It propagates shortest path information throughout the graph, accommodating complex network configurations.

- **Landmark-Based Algorithms**

Landmark-Based Algorithms, proposed in the early 2000s, are a significant advance in improving shortest path searches for large networks. These algorithms dramatically improve search efficiency by computing distances from strategically selected nodes known as landmarks. Researchers like David Peleg contributed to this subject by using landmark-based heuristics, which improved path-finding methods by decreasing the search space. During preprocessing, techniques such as Dijkstra's or Bellman-Ford are employed to compute distances between each landmark and all other nodes. During querying, these pre-computed distances are used to determine the shortest path between any two nodes, which speeds up the search process [78, 79].

The technique consists of two steps: preparation and query. Calculating the distances between nodes and landmarks during the preparation phase takes a significant amount of time and storage. However, during the query phase, precomputed distances are used to provide faster results by reducing the search space. The proper selection of landmarks is critical to the performance of these algorithms, as it determines the balance between preprocessing overhead and query time [79].

The method starts with landmarks chosen from the network and then algorithms such as Dijkstra's or Bellman-Ford are used to calculate the shortest path distances between each landmark and every other node. During a query, these precomputed distances are used to calculate the shortest path by adding the distances from the source node to a landmark and from the landmark to the target node. Heuristic approaches are used to improve the distance estimates and reduce the search space. The number of landmarks is important because it affects the balance between preprocessing time, storage needs and query efficiency. A well-balanced number of landmarks improves query accuracy while minimizing computational and storage requirements.

Key features of the Landmark-Based Algorithms are:

- It uses precomputed distances from specified landmarks to generate heuristic estimates for shortest path requests.

- It calculates shortest path lengths from landmarks to all other nodes and demands a substantial amount of computer resources and storage.
- It reduces query times by employing heuristic estimations to narrow the search space, typically resulting in near-instantaneous query replies.
- It uses a small, strategically selected collection of landmark nodes to balance preprocessing cost and query accuracy.
- It provides a trade-off between preprocessing time and query performance; adding landmarks can improve heuristic accuracy but at the cost of longer preparation and storage times.
- It is effective for large-scale graphs, such as transportation networks, where rapid shortest path queries are required.
- It is adaptable for numerous graph types and can be fine-tuned by varying the number and location of landmarks.

The following chapter provides the review of all pairs shortest path algorithms of the path planning algorithms, outlining their underlying principles within the scope of the thesis.

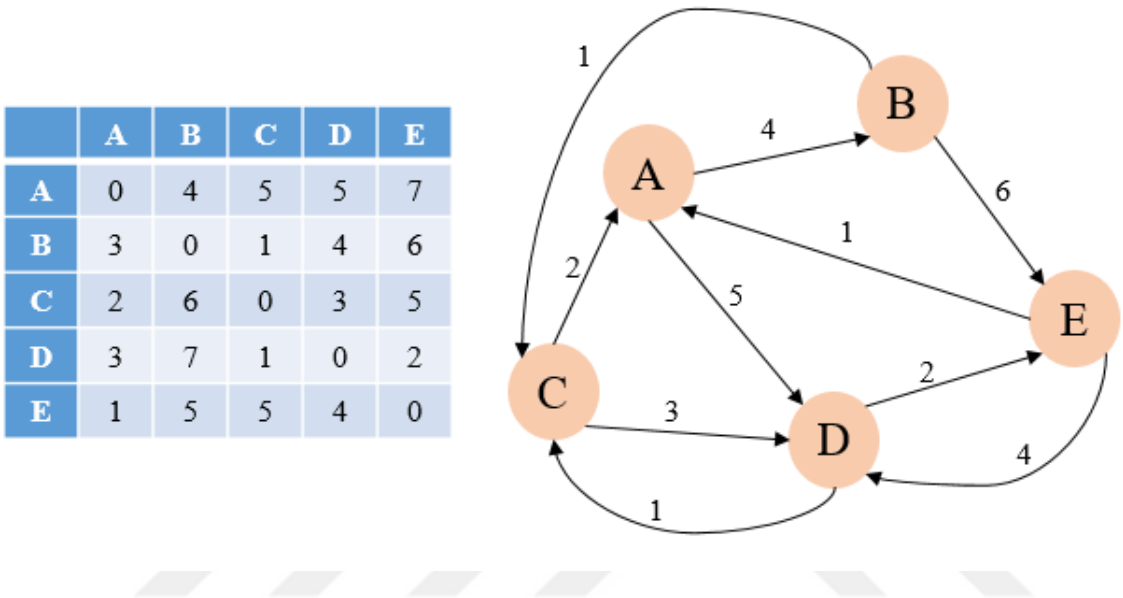
2.2.3. All-Pairs Shortest Path Algorithms

All-Pairs Shortest Path (APSP) algorithms are useful in computational graph theory because they address the problem of identifying the shortest paths between all pairs of vertices in a graph. These algorithms are widely used in many fields, including transportation planning, communication networks and bioinformatics. They efficiently explore connectivity and network structure for optimization and analysis. APSP algorithms improve route planning and data flow in complex systems by analyzing how nodes communicate [80].

- **Floyd-Warshall Algorithm**

The Floyd-Warshall algorithm, developed by Robert Floyd and Stephen Warshall, is a basic method in computer science and graph theory. They found the method in 1967 [81]. It computes the shortest path between each pair of nodes in a weighted network. This method is extremely efficient and can manage networks with both positive and negative edge weights, making it a useful tool for resolving a variety of network and connectivity problems [82]. Floyd Warshall uses an all-pair shortest path approach, while Dijkstra's and Bellman Ford use a single-source

shortest path method. This approach applies to both directed and undirected weighted graphs. However, it does not apply to graphs with negative cycles. It uses Dynamic Programming to examine every potential path via every conceivable node to determine the shortest distance between each pair of nodes [83].



	A	B	C	D	E
A	0	4	5	5	7
B	3	0	1	4	6
C	2	6	0	3	5
D	3	7	1	0	2
E	1	5	5	4	0

Figure 22. Working principle of the Floyd-Warshall Algorithm.

The algorithm initializes distances using edge weights, with each distance vertex to itself set to zero as shown in Figure 22. It uses three stacked loops to iterate over all pairs of vertices i and j, with an intermediate vertex k updating and refining the shortest path estimations. For each pair, the program determines whether the path through k provides a shorter route and changes the distance matrix accordingly. The resulting matrix comprises the shortest paths between all vertex pairs. Furthermore, the system detects negative weight cycles by observing whether any diagonal entry (the distance between a vertex and itself) becomes negative. It is best suited for small to medium-sized graphs, with a time complexity of $O(V^3)$, where V represents the number of vertices in the graph.

Key features of the Floyd-Warshall Algorithm are:

- It finds the shortest paths between all pairs of vertices in a graph.
- It can handle negative edge weights but does not anticipate any negative weight cycles.

- Negative weight cycles occur when the distance between a vertex and itself is negative.
- The shortest path estimates are iteratively improved using a dynamic programming approach.
- It creates a three-loop structure to update the distance matrix for all vertex pairings.
- It creates a distance matrix displaying the shortest pathways between all pairs of vertices.
- It has a temporal complexity of $O(V^3)$, which may be inefficient for big graphs.
- The clear iterative technique makes it simple to learn and apply.

The following chapter provides the review of the advanced pathfinding techniques of the path planning algorithms, outlining their underlying principles within the scope of the thesis.

2.2.4. Advanced Pathfinding Techniques

This section covers the Contraction Hierarchies technique, a popular advanced pathfinding technique. It focuses on the core ideas of the algorithm, such as its structure and how it improves pathfinding performance in large networks. It also discusses the operating principles of the method, such as graph contraction and node prioritization and highlights its unique features, such as better query times and scalability. These features make Contraction Hierarchies suitable for use in large networks which are transportation and logistics systems.

- **Contraction Hierarchies**

Sanders, Schultes and Geisberger introduced Contraction Hierarchies in 2008. It is a clever algorithmic technique for speeding up shortest route queries in large networks. It works notably well in roads and other large transportation networks. It is designed to preprocess the graph in order to minimize query times [84]. During preprocessing, hierarchical shortest-path algorithms build a multi-layered vertex hierarchy. For example, road networks have hierarchical features such as the arrangement of major streets, highways and city streets. In general, techniques that use contraction hierarchies have low spatial complexity. Contraction hierarchies come in various forms, including reach-based techniques, highway hierarchies and vertex routing. Transit-vertex Routing and Hub Labels, on the other hand, provide rapid query response times [85,86].

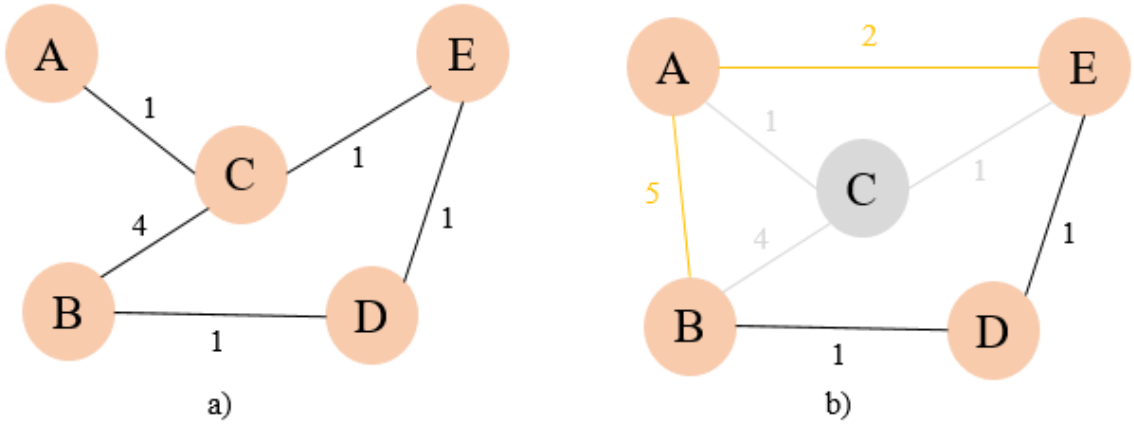


Figure 23. Working principle of the Contraction Hierarchies.

Contraction Hierarchies works by incrementally eliminating the nodes and adding shortcuts to maintain the shortest path information, resulting in a node hierarchy as shown in Figure 23. The graph is simplified by removing less significant nodes while retaining the fastest paths via new shortcuts. The search is performed by first examining higher levels of the hierarchy and then refining the findings at lower levels as needed. Although the preprocessing procedure is time consuming and complicated, it dramatically reduces query times. This method is especially well-suited for large graphs such as road networks, where quick query times are critical.

Key features of the Contraction Hierarchies are:

- It creates simplified graphs by building a multi-level hierarchy using node contraction and shortcut addition.
- Despite a time-consuming preparation procedure, the network is optimized to perform faster shortest path queries.
- It reduces the number of nodes and edges inspected during queries, resulting in faster execution.
- It Executes queries in a hierarchical order, prioritizing higher levels before refining lower levels.
- Adding shortcuts after eliminating less relevant nodes preserves the shortest path information.
- Especially useful for large-scale graphs, like transportation networks, where query speed is crucial.

- Adaptable to a variety of graph types, with a particular emphasis on road networks and other large graphs.

Table 4. Comparison table of the shortest path algorithms.

Algorithm	Time Complexity	Space Complexity	Graph Type	Preprocessing	Key Characteristics
Dijkstra's Algorithm	$O((V+E)\log V)$ (with min-heap)	$O(V)$	Weighted, non-negative	No	Greedy approach, efficient for graphs with non-negative weights.
Bellman-Ford Algorithm	$O(V*E)$	$O(V)$	Weighted, can be Negative	No	Handles negative weights and detects negative cycles.
A* Algorithm	Depends on the heuristic	Depends on the heuristic	Weighted	Yes (with heuristic)	Uses heuristics to prioritize paths, effective for pathfinding.
ALT	Depends on preprocessing and landmark count	High for preprocessing, low for query	Weighted	Yes (landmark preprocessing)	Uses landmarks and triangle inequality to optimize shortest path queries.
Contraction Hierarchies	$O(E)$ (Query)	High preprocessing	Weighted	Yes	Preprocessing simplifies the graph, highly efficient for large-scale queries.
Floyd-Warshall Algorithm	$O(V^3)$	$O(V^2)$	Weighted, can be Negative	No	Computes all-pairs shortest paths, handles negative weights but not negative cycles.
BFS	$O(V+E)$	$O(V)$	Unweighted	No	Explores nodes level by level, guarantees the shortest path in unweighted graphs.
DFS	$O(V+E)$	$O(V)$	Unweighted	No	Explores as far as possible along a branch before backtracking, useful for connectivity and cycle detection.

where the V represents vertices of the graph and E represents edges.

The following chapter provides the details of the working principle of the developed algorithm of the thesis.

3. MATERIALS AND METHODS

This section provides an in-depth review of the proposed algorithm, as well as an evaluation of the methods used in its architecture. The research focuses on the functionality, effectiveness and integration of different approaches, providing a thorough knowledge of how they contribute to the overall performance of the algorithm. The ACF technique offers significant advantages over previous methods for vehicle detection, especially in real-time applications. ACF excels at combining many image channels, such as color, gradient magnitude and orientation, to extract the key features needed for accurate vehicle detection. Unlike previous techniques such as HOG+SVM, which can struggle with computational efficiency in dynamic environments, ACF allows for faster detection at a lower computational cost. This is especially useful in real-world traffic situations where rapid response times are essential.

Furthermore, unlike deep learning-based algorithms such as R-CNN or YOLO, which need extensive training data and computational resources, ACF strikes a balance between accuracy and efficiency, making it ideal for onboard vehicle detection systems. In particular, when combined with Inverse Perspective Mapping (IPM), it increases the ability of the method to correctly estimate vehicle distance, which is critical for autonomous driving systems that require real-time decision making [87]. Shortest path techniques such as Dijkstra's, Bellman-Ford, A*, ALT, Contraction Hierarchies and Floyd-Warshall are used to find the shortest path between nodes in graphs. A number of factors influence the approach used, including the presence of negative edge weights, the need for preprocessing and the type of query being answered, such as a single-source or all-pairs shortest path. On the other hand, graph traversal algorithms such as BFS and DFS are critical for investigating graph structure. These algorithms are crucial for solving a wide range of graph-related problems, such as connection checking, cycle identification and topological sorting and they are essential tools in graph theory and applications.

BFS is a fundamental graph theory technique for finding the shortest path between a source and a destination in unweighted networks by carefully analyzing nodes at each level. This

feature qualifies BFS for issues where the optimal solution is the shortest path in an unweighted network. In addition to path finding, BFS is efficient at determining interactions within a graph. It can determine whether the graph is connected, identify all nodes reachable from a certain location and locate associated components. One of BFS notable benefits is simplicity; the approach is conceptually uncomplicated and easy to implement, making it suitable for both small and large networks. Furthermore, the flexibility of BFS is useful for a variety of applications other than simple graph traversal, such as network broadcasting, finding the shortest transformation sequences in word ladder problems and determining whether a graph is bipartite.

The algorithm uses key parts of both ACF and BFS to produce an efficient and effective vehicle detection system. Using the ability of ACF to extract strong features from multiple image channels, the system was able to accurately classify vehicles in complex and dynamic scenarios. The use of ACF allowed the detection system to maintain high accuracy while requiring low processing complexity, making it suitable for real-time applications. In parallel, BFS was used to manage shortest path computations for vehicle navigation, leveraging its ability to systematically examine nodes and analyze connectedness within the graph. This allowed the system to quickly identify the optimum path, resulting in improved real-time decision making. The combination of ACF for vehicle detection and BFS for path finding resulted in an algorithm that balanced spatial accuracy and computational efficiency to meet the requirements of real-time autonomous driving applications.

3.1. Method of VBAPAS Algorithm

The following section provides a thorough and comprehensive review of the proposed VBAPAS algorithm. This includes a thorough examination of its conceptual framework, operational techniques and the technological factors that underlie its architectural design. Each stage of the algorithm is detailed to offer light on the concepts that guided its creation, as well as the implementation methods used to achieve its objectives.

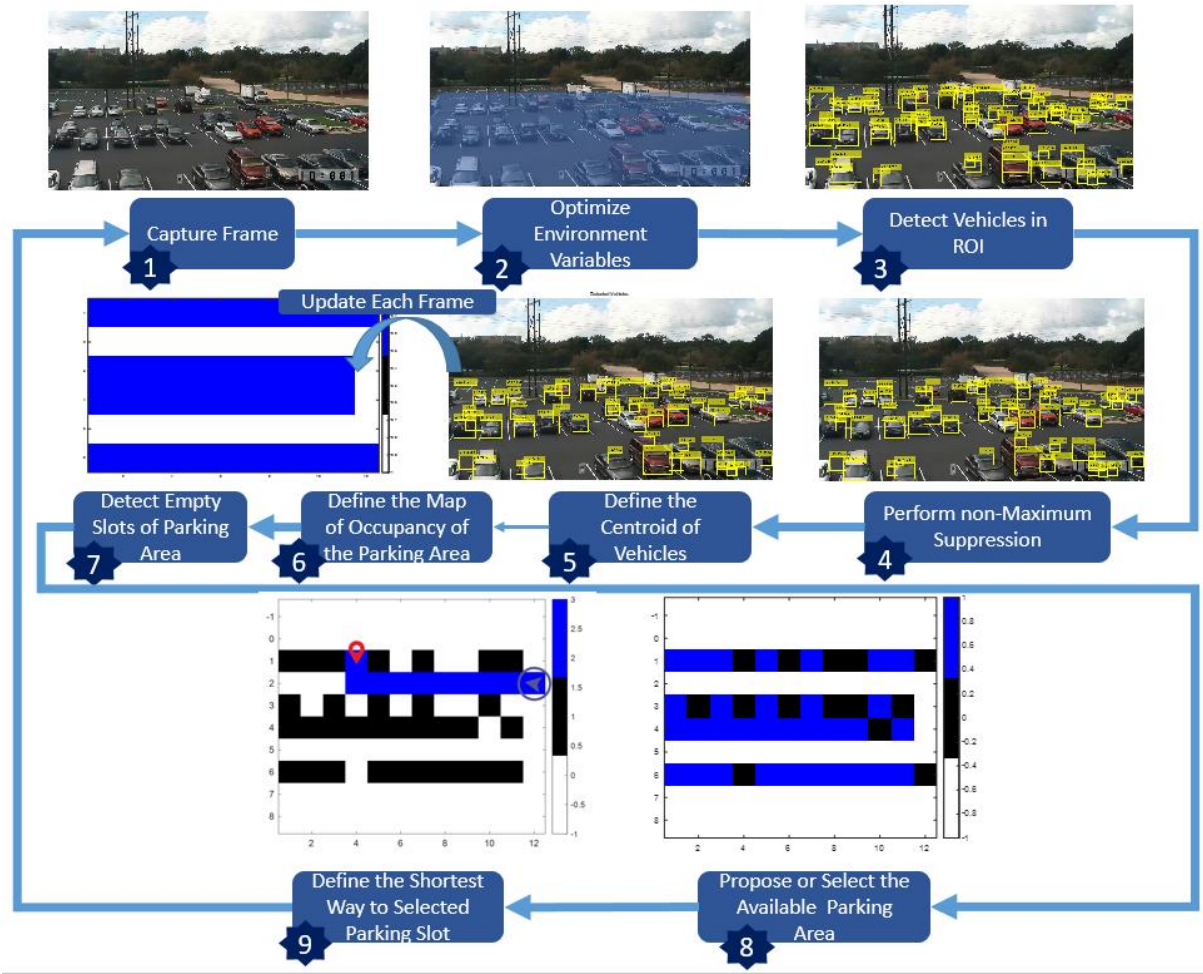


Figure 24. Proposed system flowchart.

The proposed system utilizes the ACF detector and BFS path planning for real-time applications which are detailed as follows:

1. **Capture image:** The input image is taken by the camera.
2. **Optimize variables:** Image processing variables such as detector threshold are optimized based on the environmental conditions.
3. **Detect vehicles in ROI:** Vehicles remaining in the Region of Interest (ROI) are detected.
4. **Perform a non-maximum suppression:** Vehicles in the overlapping ROIs are eliminated and only one of them is kept.
5. **Define the centroid of the vehicles:** Centers of gravity of the parked vehicles are determined.

6. **Define the map of occupancy of the parking area:** An occupancy map of the parking area within the ROI is created.
7. **Detect empty slots in the parking area:** Empty parking slots within the parking area are determined.
8. **Propose or select the available parking area:** The available parking spaces on the occupancy map are identified for the drivers.
9. **Define the shortest path to the selected parking slot:** The algorithm performs the shortest path planning based on the selected empty parking space and the position of the vehicle.

The algorithm can be basically examined under two main stages which are vehicle detection, localization and mapping, proposed shortest path to the target slot. The next subsection reviews the vehicle detection and shortest path stages.

3.1.1. Vehicle Detection, Localization

The detection and localization stage can be defined as the first step of the developed algorithm which provides the raw position and presence information of the vehicle.

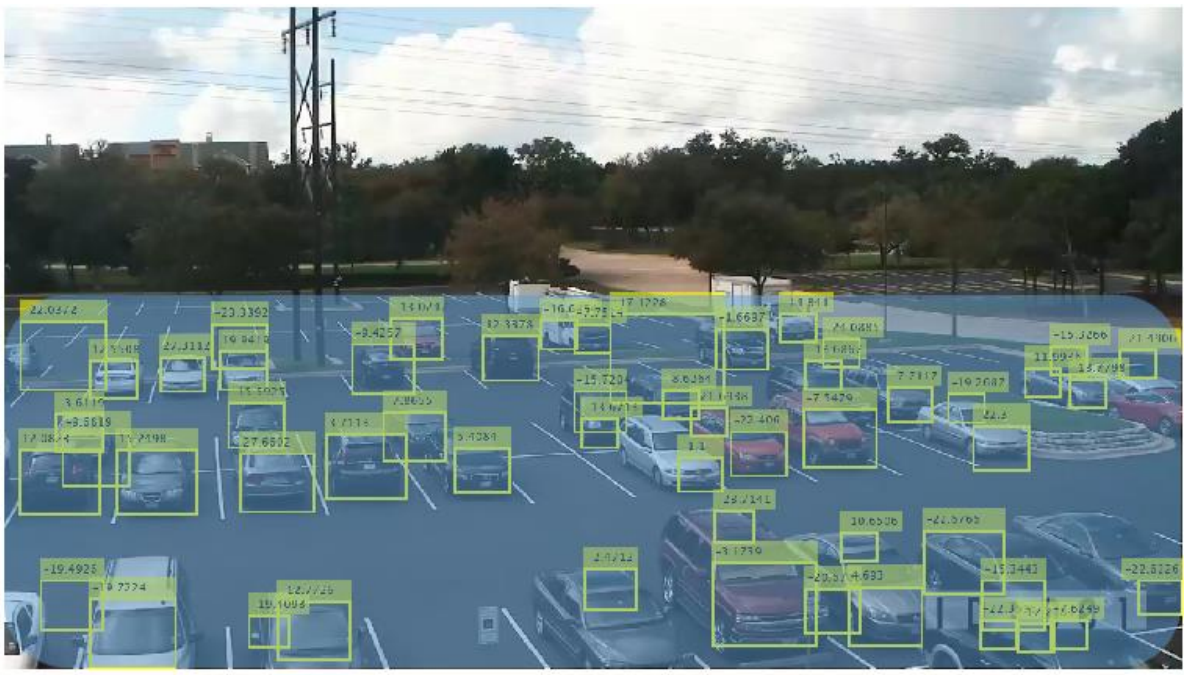


Figure 25. Vehicle detection results in the defined ROI.

Figure 25 describes the boundary boxes of the detected vehicles in the ROI. The details of the detection steps and position localization of the vehicles within the images are reviewed in the next sections.

- **Capture Image**

The developed algorithm initially processes the images taken from a single camera. In this thesis, the input frames are analyzed from a single mono camera monitoring a public parking area.

- **Variables Optimization**

The classification accuracy threshold is a key variable that determines the balance between the positive and negative results which is set during the detector pre-processing stage. The classifier generates a confidence score for each detection window, indicating the likelihood of a window containing a vehicle. The threshold is then applied to these confidence scores to determine whether the detection result is the existence of a vehicle or not. If the confidence score exceeds the threshold, the detection is considered as vehicle-exist; otherwise, it is categorized as a non-vehicle-exist. The threshold is determined depending on the desired balance between precision (percentage of true positives among all positives) and recall (percentage of true positives among all actual vehicles). A higher threshold increases the precision of detection, but it may decrease the recall (fewer false positives but more false negatives). Conversely, a lower threshold boosts the recall while decreasing precision of detection, resulting in more true positives but also more false positives.

The classification can be represented as:

$$Class(x) = \begin{cases} vehicle & if\ s(x) \geq T \\ non-vehicle & if\ s(x) < T \end{cases} \quad (1)$$

where $s(x)$ is confidence score, x is the detection window and T is the classification accuracy threshold.

When the confidence score $s(x)$ is greater than or equal to the threshold T, the detection window is classified as vehicle-exist. Otherwise, the detection window is classified as non-vehicle-exist. In the developed algorithm, the threshold is experimentally determined as $T = -25$ to determine the positive and negative results in the detection stage.

The optimum threshold is set by taking ambient circumstances and ROI into account to improve the data quality of the images captured by the camera and the algorithm effectiveness in vehicle detection. The optimization procedure involves tailoring the detection threshold values to a specific environment and conditions in the parking area. In different application fields, the threshold value can be adjusted based on ambient circumstances such as light, weather and camera angle.

As a result, the detector sensitivity is increased significantly, and the algorithm desired outputs are acquired with more accuracy. The algorithm improved sensitivity ensures that the vehicles are accurately recognized while minimizing the false detections. This modification improved the system overall dependability and accuracy. The accurate threshold value accelerates the learning and reduces the processing time. Thus, the system performance and efficiency are enhanced in real-time applications to attain effective results.

- **Detecting the Vehicles in the ROI**

Vehicles in a parking area can be parked at odd angles and partially obscured by other vehicles, or surrounded by signage, pedestrians, or plants where the advanced object detection methods should be used to overcome these challenges. The ACF detector handles these issues in the developed algorithm, because it is able to detect a wide range of visual areas, allowing it to reliably differentiate vehicles from background clutter and other objects. The ACF detector collects information from multiple channels, including color, texture and edge. This detector uses the visual cues to effectively differentiate vehicles from complex and busy backgrounds in the frame.

In this thesis, the algorithm is developed to scan a wide parking area and accurately recognize the vehicles in the parking area. The ACF detector working principle can be represented under five sub-stages as follows:

1. **Channel extraction:** ACF extracts several channels from input image. Color channels, gradient magnitude and HOG are common ones. The extracted channels represent as follows:

$$C_1(I), C_2(I), C_3(I) \ldots . C_k(I) \tag{2}$$

Where the input image is, channels on the frame are C_k .

2. **Aggregation:** Following channel extraction, the characteristics are aggregated over a local neighborhood. This comprises adding or averaging feature values inside a specific frame, which decreases dimensionality and makes features more resistant to slight fluctuations.
3. **Feature vector:** The vector $f(x)$ is created for each detection window by concatenating the aggregated features from all channels, as follows:

$$f(x) = \left[A\left(C_1(x)\right), A\left(C_2(x)\right), \ldots, A\left(C_k(x)\right)\right] \tag{3}$$

where $f(x)$ is a feature vector, $\left(C_k(x)\right)$ are different channels derived from x which is the input frame and A is the aggregation function.

4. **Classifier training:** To distinguish between positive and negative samples, the classifier, which is commonly a boosted decision tree or random forest, is trained using aggregated channel data.
5. **Detection:** The trained classifier is used to test images using a sliding window method. The detection window is dragged over the picture at different scales in order to identify objects of varied sizes. The aggregated channel features are calculated at each point of the detection window. The classifier determines if the window includes a vehicle. To obtain a confidence score $s(x)$, the feature vector $f(x)$ is used from the classifier C , for each window x as:

$$s(x) = C\left(f(x)\right) \tag{4}$$

6. **Decision rule:** The vehicle is classified based on confidence score and threshold values.

- **Perform Non-Maximum Suppression**

The Non-Maximum Suppression (NMS) is an essential post-processing approach used to improve the detection accuracy. It is highly beneficial from solving the overlapping boundary boxes issue because it is able to eliminate the duplicate detections and retains only the most confident predictions. This approach works iteratively by picking the detection with the highest confidence score and suppressing any detections that overlap significantly with it. During this iterative process, redundant detections are progressively pruned, resulting in a final set of non-overlapping bounding boxes that accurately delineate the classified vehicles.

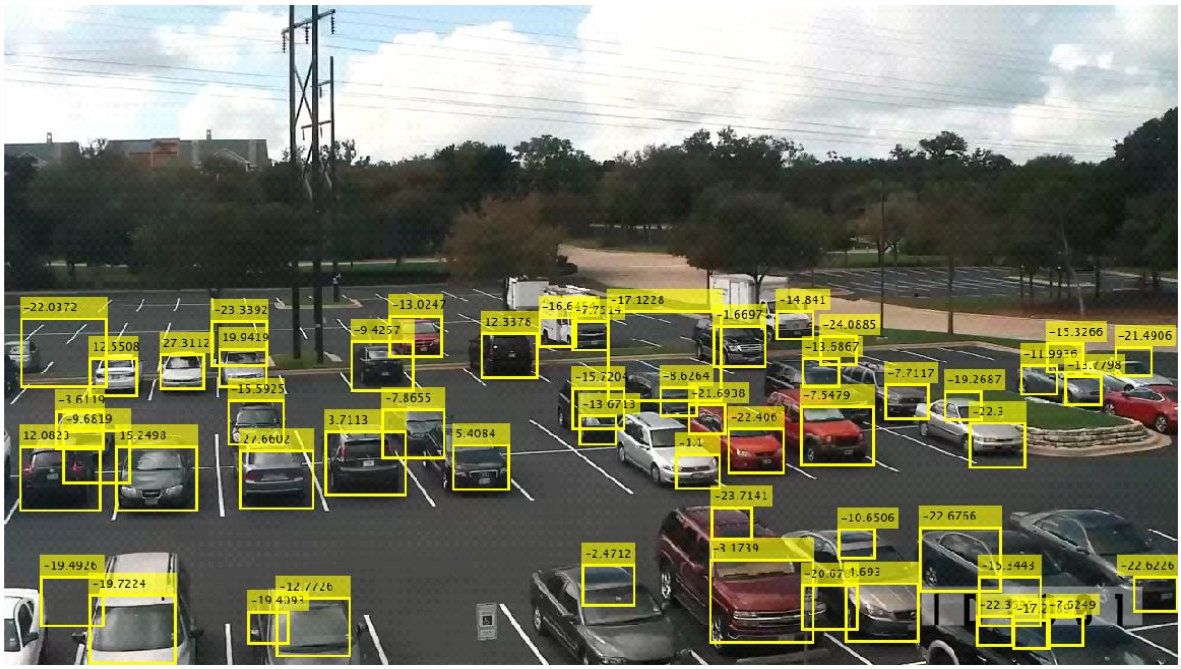


Figure 26. Vehicle detections after the NMS.

The NMS has improved the accuracy and reliability of vehicle detection and contributed to its overall performance by effectively dealing with the issue of overlapping detections as shown in Figure 26. The NMS can be detailed as:

1. **Initialization**: The algorithm starts with sorted bounding boxes $b_{\sigma(1)}, b_{\sigma(2)}, ..., b_{\sigma(i)}$ and an empty set D.
2. **Selection and Suppression:** Each bounding boxes $b_{\sigma(i)}$ are compared with each previous selected box $d \in D$. The computation of ratio $R(b_{\sigma(i)}, d)$ is defined as:

$$R\left(b_{\sigma(i)}, d\right) = \frac{Area\left(b_{\sigma(i)} \cap d\right)}{\min\left(Area\left(b_{\sigma(i)}\right), Area(d)\right)} \tag{5}$$

where $b_{\sigma(i)}$ are the bounding boxes, d is a different boundary box and $Area\left(b_{\sigma(i)} \cap d\right)$ represents the intersection between two boxes. Then, the decision process is completed based on:

$$Out = \begin{cases} add\ \ b_{\sigma(i)} & if\ R\left(b_{\sigma(i)}, d\right) \leq Overlapthreshold \\ \text{sup}press\ \ b_{\sigma(i)} & otherwise \end{cases} \tag{6}$$

3. **Output:** The output can be defined as the final set of bounding boxes.

After implementing the NMS algorithm to simplify the overlapped boundary boxes on the frame, the centroids of the classified vehicles are determined in the next subsection.

- **Defining the Centroid of the Vehicles**

During the mapping stage, the locations of the parking slots and detected vehicles are classified on two-dimensional occupancy maps based on the ROI defined according to the camera location. The method iterates through each parking slot in real time, inspecting the defined boundaries and determining whether the centroids of the detected vehicles intersect with these boundaries. This is done by comparing the coordinates of the centroids with the corresponding boundaries defined for each parking slot. The centroids of vehicles are determined as:

$$B = \{(x_{\min}, y_{\min}), (x_{\max}, y_{\max})\} \tag{7}$$

where B is the bounding box of the detected vehicles which includes the coordinate of top and bottom corners. Then the centroid of the coordinates which c_x, c_y are defined as follows:

$$c_x = \frac{x_{\min} + x_{\max}}{2} \tag{8}$$

$$c_y = \frac{y_{\min} + y_{\max}}{2} \tag{9}$$

$$c_k = (c_x, c_y) \tag{10}$$

where c_k is the center points of bounding box, y_{min} and y_{max} are the y axis of bounding box, x_{min} and x_{max} are the x axis of bounding box shown in Figure 27.

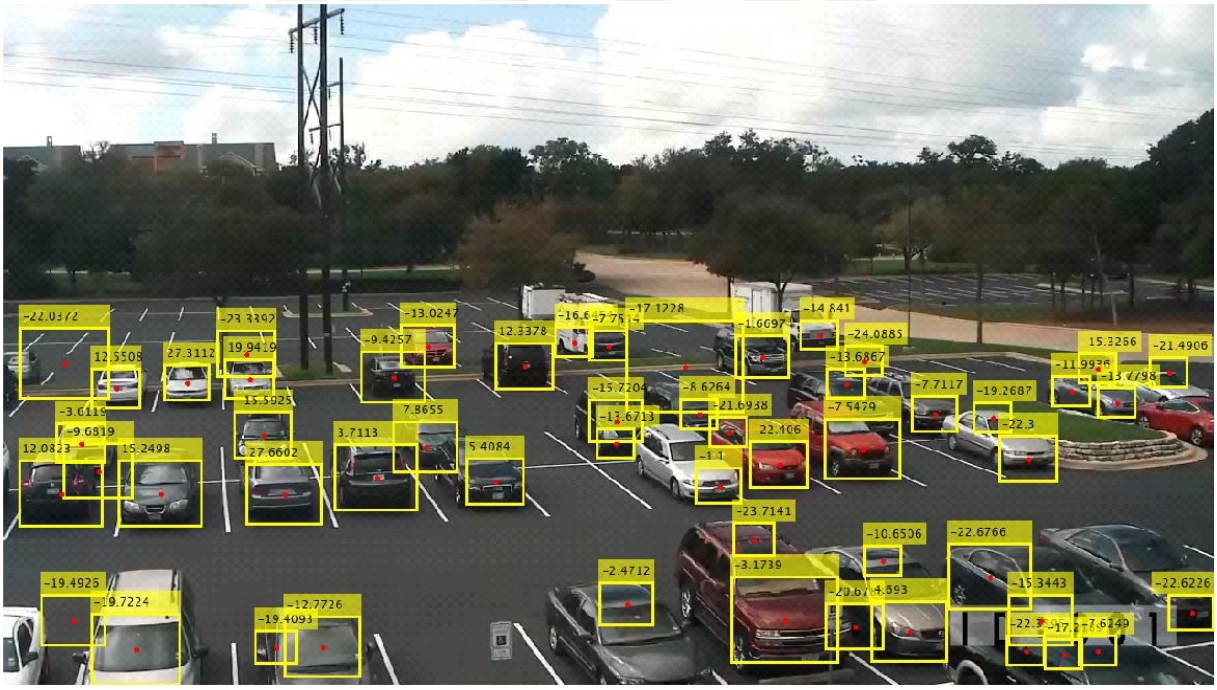


Figure 27. Defined location of the vehicle centroids.

After defining the location of centroids as shown in Figure 27 that fall within the defined bounds, the method updates the parking area matrix illustrating the occupancy status of each parking slot with detected centroids. Slots are marked as occupied, without detected centroids

are left unfilled. This systematic technique makes it easier to identify occupied and vacant slots based on the presence or absence of the observed vehicle centroids. Furthermore, by including a robust handling for cases with no detection, the algorithm ensures that the parking area analysis results are accurate and reliable. After defining the centroids of the classified vehicles on the frame, the mapping and proposing shortest path stages can be initiated. The mapping and planning stages of the VBAPAS algorithm are provided in the following chapter.

3.1.2. Mapping and Planning the Shortest Path to the Target Slot

The mapping and planning of the shortest path to the target slot can be defined as the second step of the developed algorithm. The details of the mapping steps and proposing shortest path to the target slot are reviewed in the next subsections.

- **Map of Parking Area Occupancy**

The vehicles and centroid points are detected on the frame which are used to create the occupancy map of the observed parking area.

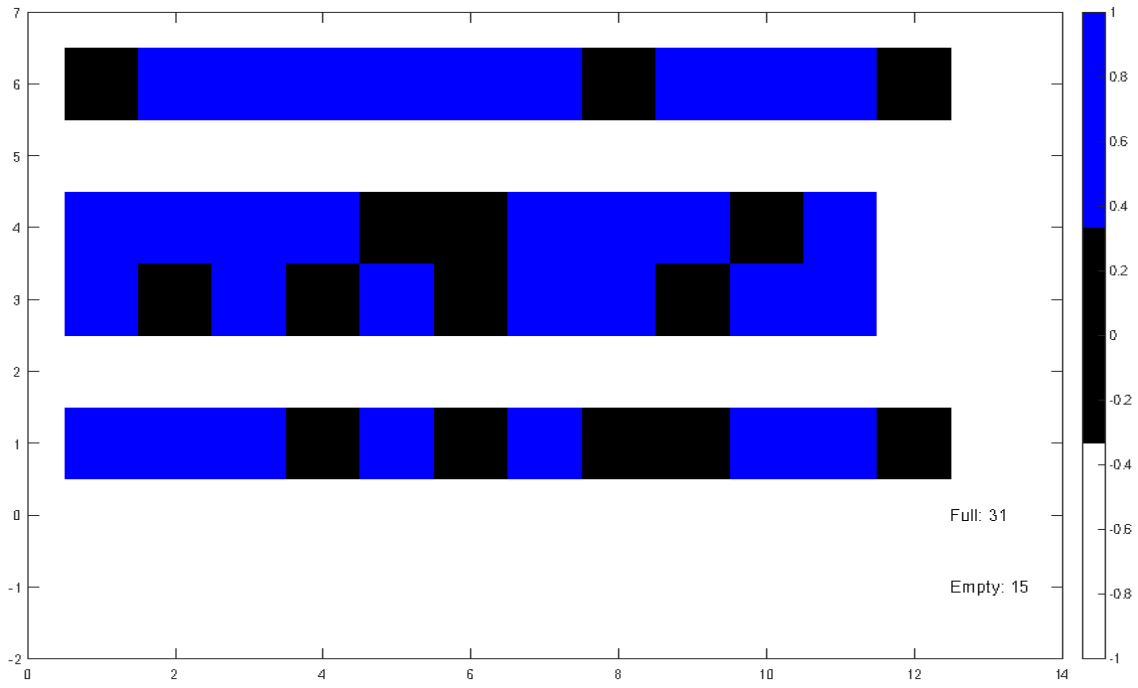


Figure 28. Occupancy map of the parking area in defined ROI.

This map includes the classification of the empty and occupied parking slots in the existing area as shown in Figure 28. When the defined map is examined, it is simply designed from a bird eye view. The occupancy map is included the parking slots, roads and limitations where the vehicles can move within the parking area. In the occupancy map, the parking slots are marked in blue as full and the areas are marked in black as empty parking areas.

	Centroid x,y	1	2	3	4	5	6	7	8	9	10	11	12
Parking Slot	1	x=91-157 y=382-434	x=167-230 y=382-434	x=235-300 y=382-434	x=311-380 y=382-434	x=381-445 y=382-434	x=448-508 y=382-434	x=520-589 y=382-434	x=593-655 y=382-434	x=656-706 y=382-434	x=707-762 y=382-434	x=776-842 y=382-434	
Road	2	R	R	R	R	R	R	R	R	R	R	R	Starting Point
Parking Slot	3	x= 52-141 y=436-472	x=143-233 y=436-472	x= 234-321 y=436-472	x= 322-408 y=436-472	x= 409-487 y=436-472	x= 490-591 y=436-472	x= 593-698 y=436-472	x= 700-767 y=436-472	x= 770-827 y=436-472	x= 830-945 y=436-472	x= 950-1020 y=436-472	R
Parking Slot	4	x= 10-114 y=473-563	x=115-233 y=473-563	x=235-349 y=473-563	x=350-453 y=473-563	x=455-567 y=473-563	x=578-670 y=473-563	x=680-781 y=473-563	x=782-858 y=473-563	x=860-953 y=473-563	x=957-997 y=473-563	x=1000-1123 y=473-563	R
Road	5	R	R	R	R	R	R	R	R	R	R	R	R
Parking Slot	6	x= 3-85 y=580-720	x= 86-241 y=580-720	x= 242-400 y=580-720	x= 401-561 y=580-720	x= 562-717 y=580-720	x= 720-895 y=580-720	x= 898-1015 y=580-720	x= 1026-1086 y=580-720	x= 1090-1279 y=580-720	1(Not in ROI)	1(Not in ROI)	

Figure 29. Decision principle for the occupancy map construction.

The occupancy map is constructed with marking individual parking slots as occupied by comparing the centroid coordinates of the vehicles with the previously determined parking slots as shown in Figure 29. If there is not a center point within the boundaries of the parking slots, then they are indicated as empty parking slots. Subsequently, the occupancy map of the park area from a bird eye view is defined for the next stages of the developed algorithm.

- **Detecting the Empty Slots in the Parking Area**

After defining the occupancy map of the parking area, the next stage involves identifying vacant parking slots where the vehicles can be parked. This stage includes a comprehensive analysis of each parking space on the occupancy map. This analysis is recorded on the map, ensuring that the data accurately depicts the current availability of the parking spaces. As shown in Figure 28, the empty slots of the parking area are marked with black color on the occupancy mapping.

- **Selecting the Most Suitable Parking Slot**

The developed algorithm includes two different options while proposing the suitable empty parking slot for the driver who has entered the parking area. The first option is the interactive selection, which provides the driver to interactively select a parking slot. The second option is automatic proposal which offers completely automatic suggestions for the vehicle drivers. In the interactive selection, the driver can choose the empty slot by using the occupancy map which shows the cluster of the vehicles. Then, the drive choice is used to classify as the target parking slot to determine the shortest path. In the automatic proposing, the algorithm chooses the empty parking slot according to the vehicle and parking slot location. Then the algorithm choice is used to classify the target parking slot to plan the shortest path.

- **Planning the Shortest Path to the Selected Parking Slot**

In determining the shortest path to the selected parking slot stage, the approach is based on a BFS implementation that is developed for path finding in a parking area. The algorithm utilizes the vehicle location and the selected empty slot location as the inputs. After that, the algorithm is initialized with the required data structures for its operation. These include a queue for managing location traversal, a Boolean matrix for tracking visited positions and a cell array to store the parent-child relationships between the positions. These structures allow for the methodical exploration of nearby positions from the starting point and ensure that each position is explored only once. The algorithm iteratively removes the points from the queue and investigates their adjacent positions, preferring unvisited and valid slots for further inquiry.

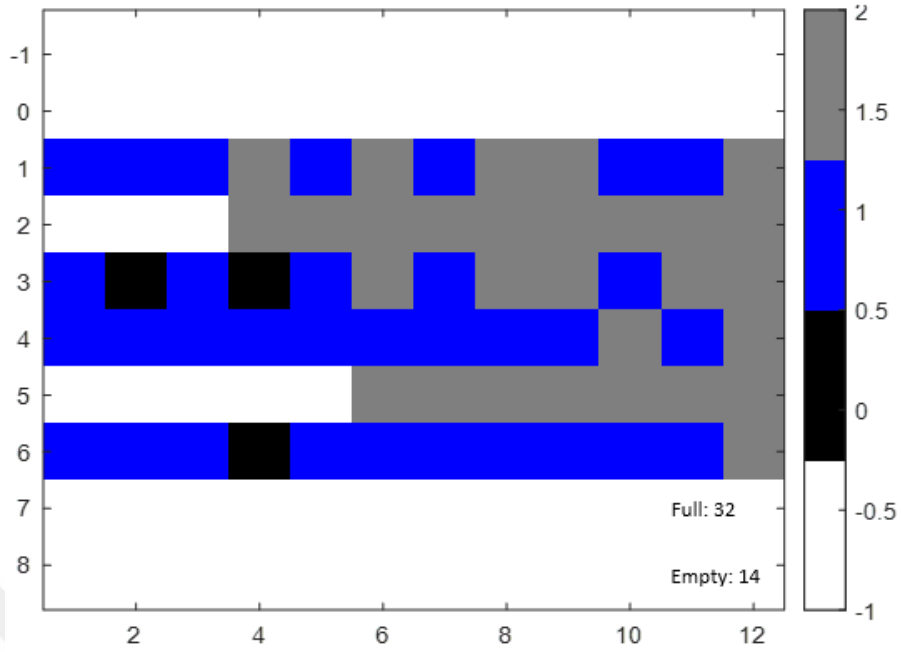


Figure 30. Parking slots that are visited by the shortest path algorithm.

When one of the empty slots is selected as the target location, the BFS algorithm searches the paths through iterations to find the shortest path from the current vehicle location to the target location. Figure 30 represents the visited nodes of the BFS algorithm in the parking area. Valid positions are determined based on their location within the parking area matrix and the absence of obstructions. When the algorithm reaches the target slot, the shortest path is reconstructed by tracing the parent-child links from the target to the present point.

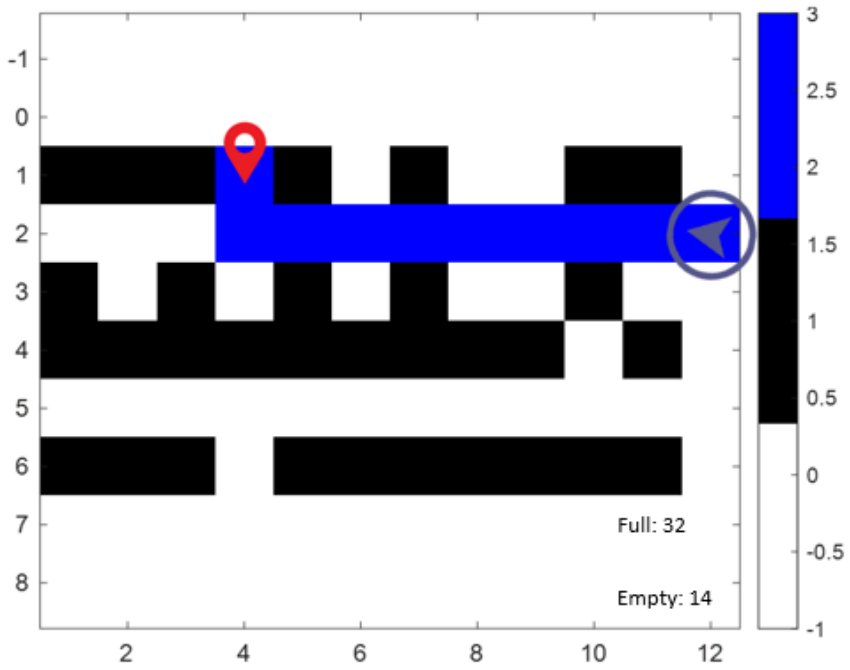


Figure 31. Planned optimal path.

The shortest path is clearly highlighted within the parking space matrix. The developed algorithm does not only provide the shortest path, but also generates the visual representation of the parking map. The visual output provides the evaluation and navigation of the parking space. The BFS algorithm effectively plans the shortest path from the vehicle location to the target slot. The compatibility of the developed algorithm with parking area management systems has been clearly demonstrated, thanks to its visualizable interface, fast real-time data transmission to the user is achieved. The vehicle position is represented in the parking area and the shortest path to the target parking slot is visually indicated for the driver by the blue line, as given in Figure 31. The simulation results of the VBAPAS algorithm are provided in the following chapter.

3.1.3. Simulation of VBAPAS Testing Results

In this section, the VBAPAS algorithm, which was developed for efficient parking area management is simulated to show its effectiveness and performance. MATLAB software was used for the simulation purposes. The video records used in the simulation were taken from publicly available video recordings of a real-world parking area and were carefully chosen to depict a variety of dynamic parking scenarios. These images, taken from various perspectives

and lighting settings, offered a realistic and diverse dataset that allowed for a thorough assessment of the VBAPAS algorithm's capacity to handle various parking issues.

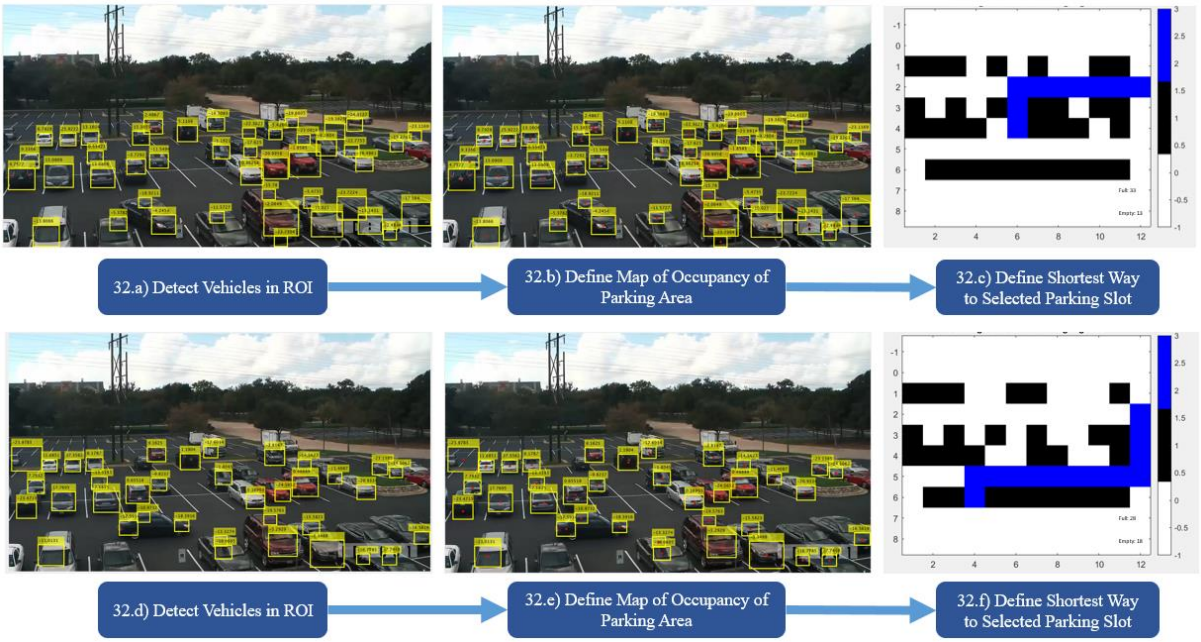


Figure 32. Finding the shortest path to the parking slot where another vehicle is leaving.

Figure 32.a represents the first testing scenario in which the algorithm was assessed by monitoring a parked vehicle that departed its parking space and exited the parking area. Figures 32.b, 32.d and 32.e show how the vehicle location was captured and updated in each frame of the series. This continuous tracking enabled real-time changes to the occupancy map of the parking area, displaying the vacated slot in Figures 32.c and 32.f. The dynamic nature of the ACF approach enabled instant modification of the occupancy data based on the positions of vehicles, ensuring that the system always had up-to-date information.

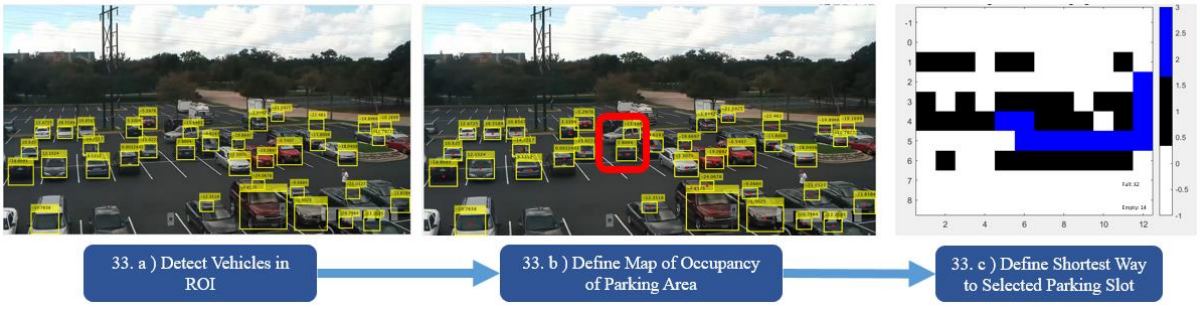


Figure 33. Finding the shortest path results based on vehicles moving on parking slots

In the second test scenario, two separate automobiles were tracked as they went around the parking lot. The red box in Figure 33.b indicates the point at which these vehicles collided. One vehicle was parked in a slot and the other was stopped on the road within the parking lot.

Figure 34. Finding the shortest path results with moving person in the parking area.

Figure 34 illustrates people in the parking lot during the final stage of the test scenario. While the system adequately manages vehicle detection and parking space allocation, it does not consider human mobility in this scenario.

3.2. Driver Intention Algorithm

The following section gives a thorough and comprehensive review of the proposed Driver Intention Algorithm. This involves a thorough examination of its conceptual framework, operational techniques and the technological factors that underlie its architectural design. Each stage of the algorithm is detailed to offer light on the concepts that guided its creation, as well as the methods of implementation used to achieve its objectives. The fundamental working principle of the created algorithm is divided into many steps as shown in Figure 35. First, the algorithm recognizes the vehicle and the lanes. Next, it detects and tracks vehicles on the road while confirming lane change statuses. The program assesses and classifies vehicle behaviors during lane changes. Finally, the driver receives feedback based on the collected and processed data.

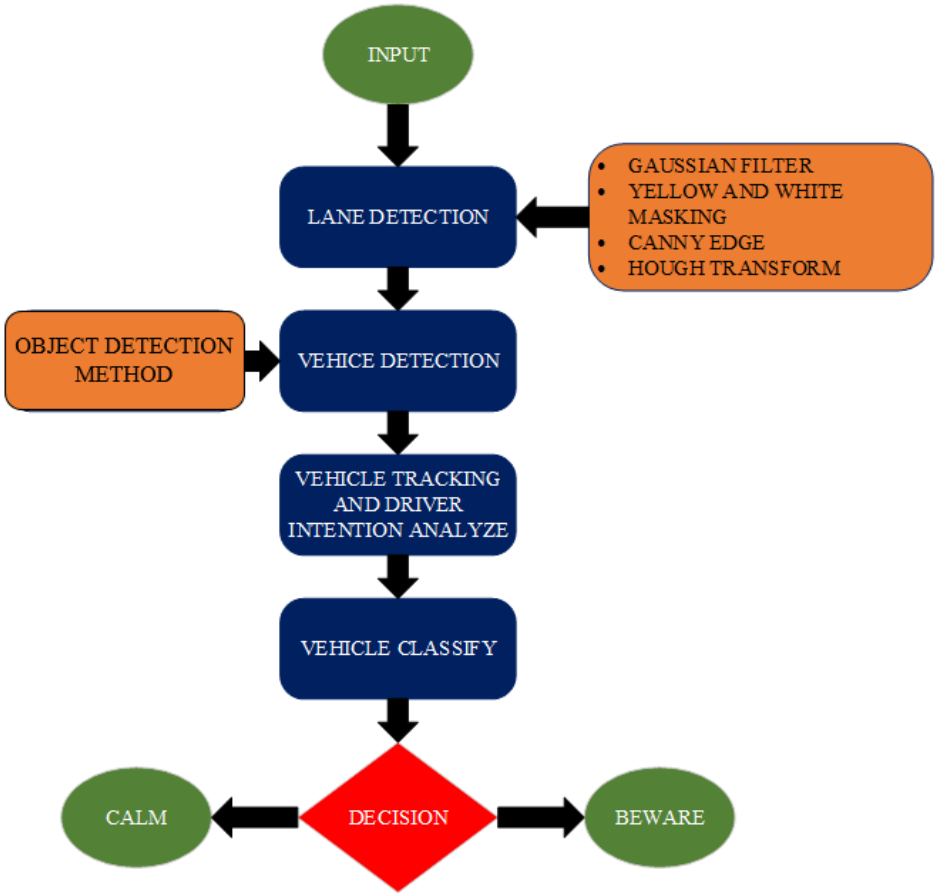


Figure 35. Architecture of the driver intention algorithm.

This algorithm focused on video-based object detection and assessment. As a result, the precision of object detection systems is critical while tracking things. A vision-based object recognition method is being considered, with the highway traffic video as input. The outputs of the algorithm are studied in greater depth in the test and results section. The vehicle classification stage of the Driver Intention Algorithm is detailed in the following chapter.

3.2.1. Vehicle Classification

The algorithm detects lanes and vehicles on the highway. The lanes were identified using the Hough transform and the canny edge detection technique. Vehicles were spotted using the vision-based detector. The output of the algorithm is the classified vehicles, which are marked with red and green boxes based on their position and course. Highway lanes are identified using

red lines and tracked in real time. The driver intention analysis stage of the Driver Intention Algorithm is detailed in the following chapter.

3.2.2. Driver Intention Analysis

The positions of the vehicles and the road lines on the frame were used to determine driver intent. The algorithm-determined road lanes are used to analyze the movement of other vehicles into these lanes based on where the reference vehicle is positioned. Thus, when neighboring vehicles on the highway enter the same lane, the driver is notified to be aware of the vehicle as feedback. The detection of road lanes reveals the movement of other vehicles in those lanes relative to the location reference vehicle. Thus, when other vehicles on the highway enter the same lane, the driver is warned to beware as feedback. The main logic of the decision mechanism controls the position of the vehicles and road lanes in the visual frame. The algorithm tracks the vehicles' positions in relation to the moving lanes. If the vehicle enters the space between the two lanes, the driver is advised to beware. Vehicles that do not enter the self-motion lanes are rated as calm. The simulation results of the Driver Intention Algorithm are provided in the following chapter.

3.2.3. Simulation of Driver Intention Algorithm

In this section, the Driver Intention Algorithm, which was developed to assess and predict driver intentions, is simulated to evaluate its performance and efficacy in real-world driving scenarios. MATLAB software was used for the simulation. Video records of the simulation were taken from publicly available video recordings of real-world highway driving collected from online public highway records. These videos, which record a wide range of traffic circumstances and driver behaviors, provide a complete dataset for testing and validating the algorithm in various types of driving scenarios.

Scenario - I

Scenario - II

Scenario - III

Figure 36. Results from Highway-I images.

As demonstrated in the Figure 36, real-time images taken from a vehicle moving in the left lane were used. Neighboring vehicles moving in the right two lanes were successfully detected. Detected vehicles are labeled "calm".

Scenario - I

Scenario - II

Figure 37. Results from Highway-II images.

Figure 37 illustrates a scenario in which the vehicle recorded by the camera is traveling down the center of the highway. In the accompanying simulation, surrounding vehicles are located, positioned, and labeled correctly. Vehicles in the same lane and those making lane changes are labeled as "beware," offering feedback that warns the driver to be careful. Vehicles

traveling in neighboring lanes without making lane changes are branded "calm," indicating a non-critical situation.

Scenario - I

Scenario - II

Figure 38. Results from Highway-III images.

.

Figure 38 demonstrates a scenario in which the vehicle recorded by the camera is traveling towards the center of the highway. Vehicles making lane changes are labeled as "beware," with feedback alerting the driver to exercise caution. In contrast, vehicles going in neighboring lanes, whether on the right and left, are classified as "calm," signifying a non-critical situation.

Scenario - I

Scenario - II

Scenario - III

Figure 39. Results from Highway-IV images.

Figure 39 illustrates a highway situation in which vehicles are dynamically classified according to their position and behavior. The vehicle in the center lane is consistently labeled "beware," indicating that the driver should exercise caution due to its proximity and possibility for interaction. In contrast, surrounding vehicles in adjacent lanes, such as the vehicle in the left lane, are designated as "calm," indicating that they pose no imminent threat because they maintain a constant positions without changing lanes.

Scenario - I

Scenario - II

Figure 40. Results from Highway-V images.

Figure 40 demonstrates two highway scenarios in which vehicles are dynamically labeled based on their position and behavior, taking into account lighting conditions. In Scenario I, the subject vehicle in the middle lane has been labeled "beware," indicating that the driver should exercise caution due to its proximity. A neighboring vehicle in the right adjacent lane that maintains a consistent position is categorized as "calm," indicating no immediate threat. In Scenario II, additional vehicles in the left and right adjacent lanes are spotted and labelled as "calm," suggesting that they are in stable positions and not interfering with the subject vehicle. The vehicle in the middle lane retains the "beware" label due to the possibility of interaction. The analysis of the simulation results of both VBAPAS and Driver Intention Algorithm are provided in the following chapter.

4. ANALYSIS AND DISCUSSION

This section provides a complete examination of the simulation results for the developed VBAPAS and Driver Intention Algorithm. The VBAPAS algorithm is tested in a variety of parking circumstances, demonstrating its usefulness in locating parking spaces and designing best routes to the target slot. The Driver Intention algorithm's effectiveness across several driving circumstances is thoroughly examined, with a focus on its ability to predict driver behavior and respond accordingly. The following chapter provides the analysis of the VBAPAS simulation results.

4.1. Analysis of VBAPAS Testing Results

In this scenario of the Figure 32, real-time monitoring capabilities, new drivers entering the parking lot got accurate and up-to-date information on available spaces. The system assigned the new driver to the previously vacated slot, as shown in Figure 32.f. Furthermore, the BFS algorithm was used to calculate the shortest path, which effectively guided the driver to the empty parking place. The integration of BFS ensured that the best route was discovered quickly, eliminating needless diversions, and saving time for the driver. The combination of ACF for real-time vehicle identification and BFS for shortest pathfinding worked well in managing a parking lot dynamic environment. After the short path calculation, the bird eye view map was

created as in Figure 32.c and Figure 32.f in order to guide the new driver. The suggested route was indicated in blue to make it easier for the driver.

In the Figure 33, the system correctly detected both vehicles and precisely identified their positions. As shown in Figure 33.c, the occupancy map reflected this by labeling the vehicle in the parking space as occupied. However, the vehicle on the road was not registered on the map, therefore the system just updated the parking slot availability. The proposed algorithm real-time capacity enabled continuous monitoring of vehicle locations and changes to the occupancy map. The recognized automobiles were accurately represented on the map, allowing for a clear view of the parking situation. In addition, as shown in Figure 33.c, the system successfully routed a new vehicle to an available unoccupied parking place. The BFS method was successful once more to find the shortest path, allowing for quick travel to the available slot. This result demonstrates the ability of the algorithm to handle many moving objects concurrently while properly representing parking availability in real time.

In the Figure 34, this exclusion is justified since pedestrian activity does not conflict with the occupancy map of the parking lot or restrict vehicle mobility within it. As a result, the algorithm effectively sends a new vehicle to its allotted parking space, as shown in Figure 34.c. This technique emphasizes the capacity of algorithms to prioritize vehicular navigation and parking efficiency while remaining operationally focused and unaffected by non-obstructive human activities.

Centroid x,y		1	2	3	4	5	6	7	8	9	10	11	12
Parking Slot	1	x=91-157 y=382-434	x=167-230 y=382-434	x=235-300 y=382-434	x=311-380 y=382-434	x=381-445 y=382-434	x=448-508 y=382-434	x=520-589 y=382-434	x=593-655 y=382-434	x=656-706 y=382-434	x=707-762 y=382-434	x=776-842 y=382-434	0 (Not in ROI)
Road	2	R	R	R	R	R	R	R	R	R	R	R	Starting Point
Parking Slot	3	x= 52-141 y=436-472	x=143-233 y=436-472	x= 234-321 y=436-472	x= 322-408 y=436-472	x= 409-487 y=436-472	x= 490-591 y=436-472	x= 593-698 y=436-472	x= 700-767 y=436-472	x= 770-827 y=436-472	x= 830-945 y=436-472	x= 950-1020 y=436-472	R
Parking Slot	4	x= 10-114 y=473-563	x=115-233 y=473-563	x=235-349 y=473-563	x=350-453 y=473-563	x=455-567 y=473-563	x=578-670 y=473-563	x=680-781 y=473-563	x=782-858 y=473-563	x=860-953 y=473-563	x=957-997 y=473-563	x=1000-1123 y=473-563	R
Road	5	R	R	R	R	R	R	R	R	R	R	R	R
Parking Slot	6	x= 3-85 y=580-720	x= 86-241 y=580-720	x= 242-400 y=580-720	x= 401-561 y=580-720	x= 562-717 y=580-720	x= 720-895 y=580-720	x= 898-1015 y=580-720	x= 1026-1086 y=580-720	x= 1090-1279 y=580-720	1(Not in ROI)	1(Not in ROI)	0 (Not in ROI)

without VBAPAS

with VBAPAS

Figure 41. Comparison of without VBAPAS and VBAPAS.

The implementation of VBAPAS has led to a significant improvement in both parking slot efficiency and overall driver experience. In an ordinary parking scenario without VBAPAS, a driver must navigate both rows of the parking lot to find an empty slot. Assuming the average speed restriction in the parking slot is 10 mph, the vehicle would need to travel an estimated 288 feet to reach an empty parking slot, specifically in the sixth row and first column of the parking area, as shown in Figure 41. This approach requires extra maneuvering and wastes time, which can aggravate driver displeasure and exacerbate congestion.

However, with the VBAPAS algorithm in place, the system efficiently guides the vehicle to the target parking slot, lowering the average trip distance to about 110 feet. This reduction in travel distance demonstrates the algorithm's effectiveness in optimizing parking slot navigation. The enhanced efficiency of slot utilization with VBAPAS not only streamlines the parking process, but it also results in a significant boost in search efficiency. Specifically, the algorithm contributes to a 61.8% increase in search efficiency, indicating a significant improvement in the time and effort necessary to find an available parking slot. This improvement not only benefits the individual drivers by reducing the amount of time they spend looking for parking, but it also reduces congestion within the parking slot, resulting in a smoother and more efficient parking experience overall. The following chapter provides the analysis of the Driver Intention Algorithm simulation results.

4.2. Analysis of Driver Intention Algorithm

Figure 36 illustrates the capacity of the system to reliably recognize and track both vehicles and lane locations in real time. As demonstrated, the system effectively detected the closest vehicles and promptly monitored their motions, allowing for continuous tracking throughout the simulation. The sensitivity of the algorithm to dynamic lane changes is shown, as the system rapidly recognizes when vehicles change lanes. The positions of two adjacent vehicles traveling in the two lanes to the right of the reference vehicle were examined and successfully categorized as "Calm."

Figure 37 represents that while the reference vehicle is moving in the middle lane, nearby vehicles are detected with great accuracy. Vehicles in the same lane as the reference vehicle and those changing lanes are effectively branded as "beware," emphasizing their potential

importance to the driver. The proposed methodology performs well in recognizing vehicles traveling in the same direction. Furthermore, the labels for the surrounding vehicles are properly positioned, ensuring clear interpretation. Notably, the system gives the driver with feedback on vehicles identified at a distance, which, although valuable for situational awareness, emphasizes the importance of exercising caution when interpreting such notifications.

In Figure 38, vehicles changing lanes were correctly labeled as “beware” accompanied by feedback warning the driver to be careful. In contrast, vehicles in adjacent lanes, whether on the right or left, were classified as “calm”, indicating a non-critical situation. This scenario proves that the algorithm can analyze and label multiple vehicles simultaneously. In addition, the developed algorithm successfully performed its task when there were multiple vehicles in the images, both on the right and left.

In Figure 39, the vehicle in the right lane looks to shift lanes from a distance. As a result of the exercise, the vehicle on the left was correctly identified and given the appropriate label. The detection of the vehicle in the rightmost lane was successful. Both the vehicle that remains in the same lane and the vehicle that changes lanes have been appropriately labeled.

In Figure 40, the image given in this part has a less light intensity than the previous options. Even under low-light settings, the system was able to identify and categorize vehicles in their respective lanes. It accomplished this in real time, labelling the vehicles it discovered in the next lane with appropriate warnings. The results of the tests show that the vehicles on the road were correctly identified under varied conditions. The provided markings for vehicles adjacent to the reference vehicle and moving in the same lane are also correct. Drivers of automobiles with red tags have stated that they are aware of the situation. The conclusion of this thesis is provided in the following chapter.

5. CONCLUSION

In this thesis, VBAPAS and Driver Intention Algorithm have been developed within the scope of smart transportation systems. The tests of the proposed algorithms were carried out using camera records placed in a public parking area and camera records of a vehicle moving on the highway. According to the testing results, the algorithms successfully detected the vehicles within the defined ROI. Algorithms tested in different scenarios, light and weather conditions played an important role in detecting vehicles during real-time tracking. Adaptive optimization algorithms were applied to manage such environmental variables. The occupancy

map of the parking area was successfully defined in the VBAPAS algorithm using the positions of the detected vehicles. When the results and images were compared, no difference was seen between the occupancy map of the parking areas and the real frames. In the second stage of the developed algorithm, the planned shortest path to the desired parking area was successfully obtained. The algorithm, whose results were evaluated with different scenarios, planned the shortest path to the target slot properly and presented it to the drivers. The predictive calculation showed that the VBAPAS application provided 61.8% time efficiency during the search for an empty slot.

In the Driver Intention Algorithm, the lanes and the number of vehicles on the road were detected by using the Hough Transform based on the Canny Edge detection and ACF. The movement patterns of the detected vehicles were analyzed using the lanes as reference points. Considering the lane changes of the vehicles in the same lane as the reference vehicle and marked with the "beware" label. As a result of the tests, the lines on the highway and neighboring vehicles moving on the highway were successfully detected. The neighboring vehicles whose movements were analyzed were successfully tagged and contributed to the prediction of potential accidents that may occur during the journey and the taking of precautions.

In future research, vehicle detection techniques can be developed to account for various environmental conditions such as weather, light fluctuations, shadow movements and night vision. By addressing these conditions that hinder vehicle detection accuracy, the overall efficiency of the system can be significantly increased. In addition, improvements can be made to expand the ability of the algorithms to detect other objects and living things both on the highway and in the parking lot. This can reduce the probability of possible accidents by allowing the detection system to account for objects and obstacles on the road for the drivers. In addition, these studies can be integrated into smart vehicle systems and environmental monitoring networks by incorporating middleware based on IoT technologies. This integration will enable the system to overcome its current limitations in certain areas and contribute to wider smart city applications. For example, with the increase in alternative fuel vehicles, public areas such as charging and hydrogen filling stations can contribute greatly to parking area management and these two developed algorithms can help prevent accidents by working in integration with each other.

REFERENCES

1. United Nations, Department of Economic and Social Affairs, Population Division (2020): UN World Population Prospects
2. International Organization of Motor Vehicle Manufacturers (OICA), 2020: OICA Global Vehicle Data
3. Sussman J.M. Perspectives on intelligent transportation systems. New York: Springer; 2005. 276 p. ISBN 0-387-23257-5
4. Papadimitratos P, La Fortelle AD, Evenssen K, Brignolo R, Cosenza S. Vehicular communication systems: enabling technologies, applications, and future outlook on intelligent transportation. IEEE Commun Mag. 2009 Nov;47(11):84-95. doi: 10.1109/MCOM.2009.5307471
5. Talebpour A, Mahmassani H.S. Influence of connected and autonomous vehicles on traffic flow stability and throughput. Transp Res Part C Emerg Technol. 2016;71:143-63. doi: 10.1016/j.trc.2016.07.007
6. KPMG. Autonomous Vehicles Readiness Index: 2020 AVRI report. KPMG; 2020
7. Barth M, Boriboonsomsin K. Real-world carbon dioxide impacts of traffic congestion. Transp Res Rec. 2008;2058(1):163-71. doi: 10.3141/2058-20.
8. Kumar A, Batra N, Mudgal A, Yadav AL. Navigating urban mobility: A review of AI-driven traffic flow management in smart cities. 2024 11th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO); 2024; Noida, India. p. 1-5. doi: 10.1109/ICRITO61523.2024.10522206.
9. Zhou B, Liu Z, Su H. 5G networks enabling cooperative autonomous vehicle localization: A survey. IEEE Trans Intell Transp Syst. 2024;25(11):15291-15313. doi: 10.1109/TITS.2024.3420084.
10. Dickmann J, Appenrodt N, Brenk C. How we gave sight to the Mercedes robotic car: Radar is the key to Mercedes Benz's autonomous car. Published July 24, 2014.
11. Azim MI, Khorasany M, Razzaghi R. Techno-economic-environmental assessment of green transportation systems. In: Interconnected Modern Multi-Energy Networks and Intelligent Transportation Systems: Towards a Green Economy and Sustainable Development. IEEE; 2024. p. 39-58. doi: 10.1002/9781394188789.ch3
12. Oladimeji D, Gupta K, Kose NA, Gundogan K, Ge L, Liang F. Smart transportation: An overview of technologies and applications. *Sensors*. 2023;23:3880. doi: 10.3390/s23083880
13. Zantalis F, Koulouras G, Karabetsos S, Kandris D. A review of machine learning and IoT in smart transportation. Future Internet. 2019;11:94. doi: 10.3390/fi11040094
14. Zhu L, Yu FR, Wang Y, Ning B, Tang T. Big data analytics in intelligent transportation systems: A survey. IEEE Trans Intell Transp Syst. 2019;20(1):383-398. doi: 10.1109/TITS.2018.2815678
15. Szeliski R. Computer Vision: Algorithms and Applications. Springer; 2010.
16. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521(7553):436-444. doi:10.1038/nature14539
17. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial nets. In: Ghahramani Z, Welling M, Cortes C, Lawrence N, Weinberger KQ, editors. Advances in Neural Information Processing Systems; 2014; Curran Associates, Inc.; 2014. p.

18. Horgan J, Hughes C, McDonald J, Yogamani S. Vision-based Driver Assistance Systems: Survey, Taxonomy and Advances. arXiv:2104.12583. 2021 Apr 26.
19. Grigorescu S, Trasnea B, Cocias T, Macesanu G. A survey of deep learning techniques for autonomous driving. arXiv. 2019 Oct 17. Available from: https://arxiv.org/abs/1910.07738
20. Chen L, Zhang H, Cheng B. Vision-based perception for autonomous driving: The current challenges and future perspectives. IEEE Trans Intell Veh. 2021.doi.org/10.1109/TIV.2021.3055618
21. The Prometheus Project: The Story Behind One of AV's Greatest Developments. Autonomous Vehicle International. Available from: www.autonomousvehicleinternational.com/features/the-prometheus-project.html
22. Sun Z, Bebis G, Miller R. On-road vehicle detection: a review. IEEE Trans Pattern Anal Mach Intell. 2006 May;28(5):694-711. doi: 10.1109/TPAMI.2006.104. PMID: 16640257.
23. Zou Z, Chen K, Shi Z, Guo Y, Ye J. Object Detection in 20 Years: A Survey. arXiv:1905.05055. 2019 May 13.
24. Benali Amjoud A, Amrouch M. Object detection using deep learning, CNNs, and vision transformers: A review. IEEE Access. 2023 Jan; PP(99):1-1. doi: 10.1109/ACCESS.2023.3266093.
25. Zhao ZQ, Zheng P, Xu ST, Wu X. Object detection with deep learning: A review. IEEE Trans Neural Netw Learn Syst. 2019;30(11):3212-3232. doi: 10.1109/TNNLS.2018.2876865.
26. Dalal N, Triggs B. Histograms of oriented gradients for human detection. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05); 2005 Jun 20-26; San Diego, CA, USA. IEEE; 2005. p. 886-93 vol. 1. doi: 10.1109/CVPR.2005.177.
27. Viola P, Jones M. Rapid object detection using a boosted cascade of simple features. In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001; 2001 Dec 8-14; Kauai, HI, USA. IEEE; 2001. p. I-I. doi: 10.1109/CVPR.2001.990517.
28. Song X, Wu T, Jia Y, Zhu S-C. Discriminatively trained And-Or tree models for object detection. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition; 2013 Jun 23-28; Portland, OR, USA. IEEE; 2013. p. 3278-85. doi: 10.1109/CVPR.2013.421.
29. Dollár P, Appel R, Belongie S, Perona P. Fast feature pyramids for object detection. IEEE Trans Pattern Anal Mach Intell. 2014 Aug;36(8):1532-45. doi: 10.1109/TPAMI.2014.2300479.
30. Ren S, He K, Girshick R, Sun J. Faster R-CNN: Towards real-time object detection with region proposal networks. arXiv. 2015 Jun 4. Available from: arXiv:1506.01497.
31. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C-Y, Berg AC. SSD: Single Shot MultiBox Detector. arXiv. 2015 Dec 8. Available from: arXiv:1512.02325.
32. Othmani M. A vehicle detection and tracking method for traffic video based on Faster R-CNN. Multimedia Tools and Applications. 2022 Aug;81(2):2205-2221. doi: 10.1007/s11042-022-12715-4.
33. Abbas SM, Singh SN. Region-based object detection and classification using Faster R-CNN. In: 2018 4th International Conference on Computational Intelligence & Communication Technology (CICT); 2018 Feb 15-17; Noida, India. IEEE; 2018. p. 260-5. doi: 10.1109/CIACT.2018.8480413.

34. Girshick R. Fast R-CNN. In: 2015 IEEE International Conference on Computer Vision (ICCV); 2015 Dec 7-13; Santiago, Chile. IEEE; 2015. p. 1440-8. doi: 10.1109/ICCV.2015.169.
35. Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: Unified, real-time object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2016 Jun 27-Jul 2; Las Vegas, NV, USA. IEEE; 2016. p. 779-88. doi: 10.1109/CVPR.2016.91.
36. Redmon J, Farhadi A. YOLO9000: Better, faster, stronger. arXiv. 2016 Dec 25. Available from: arXiv:1612.08242.
37. Redmon J, Farhadi A. YOLOv3: An incremental improvement. arXiv. 2018 Apr 8. Available from: arXiv:1804.02767.
38. Bochkovskiy A, Wang C-Y, Liao H-YM. YOLOv4: Optimal speed and accuracy of object detection. arXiv. 2020 Apr 23. Available from: arXiv:2004.10934.
39. Zhang Y, Guo Z, Wu J, Tian Y, Tang H, Guo X. Real-time vehicle detection based on improved YOLO v5. Sustainability. 2022;14(19):12274. doi:10.3390/su141912274.
40. Li C, Li L, Jiang H, Weng K, Geng Y, Li L, Ke Z, Li Q, Cheng M, Nie W, Li Y, Zhang B, Liang Y, Zhou L, Xu X, Chu X, Wei X, Wei X. YOLOv6: A single-stage object detection framework for industrial applications. arXiv. 2022 Sep 7. Available from: arXiv:2209.02976.
41. Wang C-Y, Bochkovskiy A, Liao H-YM. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv. 2022 Jul 6. Available from: arXiv:2207.02696.
42. Hussain M. YOLO-v1 to YOLO-v8, the rise of YOLO and its complementary nature toward digital manufacturing and industrial defect detection. Machines. 2023;11(7):677. doi:10.3390/machines11070677.
43. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C-Y, Berg AC. SSD: Single shot multibox detector. arXiv. 2015 Dec 8. Available from: arXiv:1512.02325.
44. Lin TY, Goyal P, Girshick R, He K, Dollár P. Focal loss for dense object detection. arXiv. 2017 Aug 7. Available from: arXiv:1708.02002.
45. Zhou X, Wang D, Krähenbühl P. Objects as points. arXiv. 2019 Apr 16. Available from: arXiv:1904.07850.
46. Tan M, Pang R, Le QV. EfficientDet: Scalable and efficient object detection. arXiv. 2019 Nov 20. Available from: arXiv:1911.09070.
47. Zhao Q, Sheng T, Wang Y, Tang Z, Chen Y, Cai L, Ling H. M2Det: A single-shot object detector based on multi-level feature pyramid network. arXiv. 2018 Nov 12. Available from: arXiv:1811.04533.
48. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, Uszkoreit J, Houlsby N. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv. 2020 Oct 22. Available from: arXiv:2010.11929.
DETR (Detection Transformer)
49. Carion N, Massa F, Synnaeve G, Usunier N, Kirillov A, Zagoruyko S. End-to-end object detection with transformers. arXiv. 2020 May 26. Available from: arXiv:2005.12872.
50. Fang Y, Liao B, Wang X, Fang J, Qi J, Wu R, Niu J, Liu W. You only look at one sequence: Rethinking transformer in vision through object detection. In: Ranzato M, Beygelzimer A,

Dauphin Y, Liang PS, Wortman Vaughan J, editors. Advances in Neural Information Processing Systems. Vol. 34. Curran Associates, Inc.; 2021. p. 26183-26197
51. Zou Z, Chen K, Shi Z, Guo Y, Ye J. Object detection in 20 years: A survey. arXiv. 2019 May 13 . Available from: arXiv:1905.05055.
52. Duan K, Bai S, Xie L, Qi H, Huang Q, Tian Q. CenterNet: Keypoint triplets for object detection. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV); 2019; Seoul, Korea (South). p. 6568-6577. doi:10.1109/ICCV.2019.00667.
53. Cai Z, Vasconcelos N. Cascade R-CNN: Delving into high quality object detection. arXiv. 2017 Dec 3. Available from: arXiv:1712.00726.
54. Liu L, Ouyang W, Wang X, et al. Deep learning for generic object detection: A survey. Int J Comput Vis. 2020;128:261–318. doi:10.1007/s11263-019-01247-4.
55. Zhang S, Wen L, Bian X, Lei Z, Li SZ. Single-shot refinement neural network for object detection. arXiv. 2017 Nov 18. Available from: arXiv:1711.06897.
56. Law H, Deng J. CornerNet: Detecting objects as paired keypoints. arXiv. 2018 Aug 3. Available from: arXiv:1808.01244.
57. Zhou X, Wang D, Krähenbühl P. Objects as points. arXiv. 2019 Apr 16. Available from: arXiv:1904.07850.
58. Lin TY, Dollar P, Girshick RB, He K, Hariharan B, Belongie SJ. Feature pyramid networks for object detection. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2016. p. 936-944.
59. Singh B, Najibi M, Davis LS. SNIPER: Efficient multi-scale training. arXiv. 2018 May 23. Available from: arXiv:1805.09300
60. Li Y, Chen Y, Wang N, Zhang Z. Scale-aware trident networks for object detection. arXiv. 2019 Jan 7, doi: 10.48550/arXiv.1901.01892.
61. Hoang VD, Jo KH. Path planning for autonomous vehicle based on heuristic searching using online images. Vietnam J Comput Sci. 2015;2:109–120. doi:10.1007/s40595-014-0035-4.
62. Even S, Graph Algorithms. 2nd ed. Tel Aviv University; 1979.
63. Korf RE. Depth-first iterative-deepening: An optimal admissible tree search. Artif Intell. 1985;27(1):97-109. doi:10.1016/0004-3702(85)90084-0.
64. Tarjan R. Depth-first search and linear graph algorithms. In: 12th Annual Symposium on Switching and Automata Theory (SWAT 1971); 1971 Oct 25-27; East Lansing, MI, USA. p. 114-121. doi: 10.1109/SWAT.1971.10.
65. Zhang F, Lin H, Zhai J, et al. An adaptive breadth-first search algorithm on integrated architectures. J Supercomput. 2018;74:6135–6155. doi: 10.1007/s11227-018-2525-0.
66. Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to Algorithms. 3rd ed. 2009.
67. Sahrita AP, Prihandini RM, Agatha AB. Graph theory: Application of graphs to the shortest path problem with Dijkstra's algorithm. June 2024.
68. Popa B, Popescu D. Analysis of algorithms for shortest path problem in parallel. In: 2016 17th International Carpathian Control Conference (ICCC); 2016; High Tatras, Slovakia. p. 613-617. doi:10.1109/CarpathianCC.2016.7501169.
69. Laaksonen A. Competitive Programmer's Handbook. Draft. July 3, 2018.
70. Pandey V, Yadav SC, Arora P. Retiming technique for clock period minimization using shortest path algorithm. In: 2016 International Conference on Computing, Communication

and Automation (ICCCA); 2016; Greater Noida, India. p. 1418-1423. doi:10.1109/CCAA.2016.7813942.
71. Dela Cruz JC, Magwili GV, Mundo JPE, Gregorio GPB, Lamoca MLL, Villaseñor JA. Items-mapping and route optimization in a grocery store using Dijkstra's, Bellman-Ford and Floyd-Warshall algorithms. In: 2016 IEEE Region 10 Conference (TENCON); 2016; Singapore. p. 243-246. doi:10.1109/TENCON.2016.7847998.
72. Candra A, Budiman MA, Hartanto K. Dijkstra's and A-Star in finding the shortest path: A tutorial. In: 2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA); 2020; Medan, Indonesia. p. 28-32. doi:10.1109/DATABIA50434.2020.9190342.
73. Ju C, Luo Q, Yan X. Path planning using an improved A-star algorithm. In: 2020 11th International Conference on Prognostics and System Health Management (PHM-2020 Jinan); 2020; Jinan, China. p. 23-26. doi:10.1109/PHM-Jinan48558.2020.00012.
74. Duchoň F, Babinec A, Kajan M, Beňo P, Florek M, Fico T, Jurišica L. Path planning with modified A* algorithm for a mobile robot. Procedia Eng. 2014;96:59-69. doi:10.1016/j.proeng.2014.12.098.
75. Goldberg AV, Radzik T. A heuristic improvement of the Bellman-Ford algorithm. Appl Math Lett. 1993;6(3):3-6. doi:10.1016/0893-9659(93)90022-F.
76. Parimala M, Broumi S, Prakash K, et al. Bellman–Ford algorithm for solving shortest path problem of a network under picture fuzzy environment. Complex Intell Syst. 2021;7:2373–2381. doi:10.1007/s40747-021-00430-w
77. Thorat S, Rahane S. Review of shortest path algorithm. Int Res J Eng Technol. 2016;3(8).
78. Goldberg AV, Harrelson C. Computing the shortest path: A* search meets graph theory. MSR-TR-2004-24. Microsoft Research; 2004 Jul.
79. Tretyakov K, Armas Cervantes A, García-Bañuelos L, Vilo J, Dumas M. Fast fully dynamic landmark-based estimation of shortest path distances in very large graphs. In: Proceedings of the 20th ACM Conference on Information and Knowledge Management (CIKM 2011); 2011 Oct 24-28; Glasgow, United Kingdom. p. 1965-1968. doi: 10.1145/2063576.2063834.
80. Botea A, Harabor D. Path planning with compressed all-pairs shortest paths data. In: International Conference on Automated Planning and Scheduling (ICAPS-13); 2013 Jun;
81. Ramadhan Z, Siahaan APU, Mesran M. Prim and Floyd-Warshall comparative algorithms in shortest path problem. In: Proceedings of the ICASI; 2018 Apr 23-24; EAI. p. 1-5. doi: 10.4108/eai.23-4-2018.2277598.
82. Abu-Ryash HM, Tamimi AA. Comparison studies for different shortest path algorithms. Int J Comput Appl. 2015 May;14(8):1-7. doi: 10.24297/ijct.v14i8.1857.
83. Risald AE, Mirino S, Suyoto. Best routes selection using Dijkstra and Floyd-Warshall algorithm. In: 2017 11th International Conference on Information & Communication Technology and System (ICTS); 2017 Nov 22-24; Surabaya, Indonesia. p. 155-8. doi: 10.1109/ICTS.2017.8265662.
84. Hautekiet O. Contraction hierarchies for efficient routing in networks: impact of contraction order. [master's dissertation]. Ghent: Ghent University; 2021
85. Madkour A, Aref WG, Rehman FU, Rahman MA, Basalamah S. A survey of shortest-path algorithms. arXiv preprint arXiv:1705.02044. 2017. doi: 10.48550/arXiv.1705.02044.

86. Batz G. Time-dependent route planning with contraction hierarchies [dissertation]. Karlsruhe: Karlsruher Institut für Technologie (KIT); 2014. Available from: urn:nbn:de:swb:90-477592
87. Kim JB. Efficient vehicle detection and distance estimation based on aggregated channel features and inverse perspective mapping from a single camera. Symmetry. 2019;11(10):1205. doi: 10.3390/sym11101205.
88. Girshick R, Donahue J, Darrell T, Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv preprint arXiv:1311.2524. 2013. doi: 10.48550/arXiv.1311.252.
89. Chandiramani J, Gutierrez Cascales P, Sorto A, Hiranandani A, Chen M, Mendoza AC. Detecting brain tumors in MRI images using object detection machine learning models. May 2024. doi: 10.13140/RG.2.2.26822.69443.