Faculty of Technology and Society
Computer Engineering

Computer Science: Master Thesis
DA613A

A Self-policing Smart Parking Solution

Yurdaer Dalkic
Hadi Deknache

Exam: Computer Science, Master's Programme (One Year)
Subject Area: Computer Science
Examiner: Bengt J. Nilsson
Supervisor: Johan Holmgren
Date: 13/06-2019

Abstract

With the exponential growth of vehicles on our streets, the need for finding an unoccupied parking spot today could most of the time be problematic, but even more in the coming future. Smart parking solutions have proved to be a helpful approach to facilitate the localization of unoccupied parking spots. In many smart parking solutions, sensors are used to determine the vacancy of a parking spot. The use of sensors can provide a highly accurate solution in terms of determining the status of parking lots. However, this is not ideal from a scalability point of view, since the need for installing and maintaining each of the sensors is not considered cost-effective. In the latest years vision based solutions have been considered more when building a smart parking solution, since cameras can easily be installed and used on a large parking area. Furthermore, the use of cameras can be developed to provide a more advanced solution for checking in at a parking spot and also for providing the information about whether a vehicle is placed unlawfully. In our thesis, we developed a dynamic vision-based smart parking prototype with the aim to detect vacant parking spots and illegally parked vehicles.

Keywords: IoT, Smart Parking, Smart City, Deep Learning, Convolutional Neural Network, Machine learning, Vision-based solution, Sensors, illegally parked vehicles, vacant parking detection

Acknowledgment

We would like to show our appreciation to Johan Holmgren for guiding us throughout the thesis work and providing us with invaluable advice and feedback. We would also like to thank Atea for giving us the opportunity to perform this thesis in collaboration with the company.

List of Figures

List of Tables

Abbreviations

AI	**A**rtificial **I**ntelligence
CNN	**C**onvolutional **N**eural **N**etwork
DBSCAN	**D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise
FSK	**F**requency **S**hift **K**eying
GDPR	**G**eneral **D**ata **P**rotection **R**egulations
GPU	**G**raphics **P**rocessing **U**nit
IoT	**I**nternet **o**f **T**hings
IoU	**I**ntersection **o**ver **U**nion
LAN	**L**ocal **A**rea **N**etwork
LoRaWAN	**Lo**ng **Ra**nge **W**ide **A**rea **N**etwork
LoRa	**Lo**ng **Ra**nge
mAP	**m**ean **A**verage **P**oint
ML	**M**achine **L**earning
NFC	**N**ear **F**ield **C**ommunication
R-CNN	**R**egion-**C**onvolutional **N**eural **N**etwork
RFID	**R**adio **F**requency **ID**entification
TTN	**T**he **T**hings **N**etwork
UHF	**U**ltra **H**igh **F**requency
WSN	**W**ireless **S**ensor **N**etwork
YOLO	**Y**ou **O**nly **L**ook **O**nce

Table of Contents

1 Introduction

The amount of vehicles on the streets today is steadily growing [1]. Consequently, this contributes to multiple negative effects, such as traffic congestion, lack of vacant parking spots, pollution, and noise. A car is on average parked about 95% of the day, which means that the vehicles are most of the time parked [2]. Moreover, finding a vacant place to park a vehicle, also known as a parking spot, when going to work, shopping, or other exclusive events, could result in consuming a lot of time and fuel [3]. Furthermore, other contributions to the limited amount of parking spots today are caused by vehicles that are parked unlawfully, taking up more space than needed or leaving a vehicle or an object on the parking spot without any permissions [4].

According to a French study, 70 million hours are spent each year searching for a vacant parking spot in France, which equals to €700 million lost in one year. Furthermore, it is estimated that from 5 to 10% of the city traffic is generated by vehicles searching for a parking spot and this value can increase to 60% in small streets [5].

All these factors have made the availability of parking lots important for better urban planning. A parking lot is an open area consisting of several parking spots where vehicles can be parked. In large American cities, such as Los Angeles and San Francisco, the number of on-street parking spots is 63% and 75%, respectively among all the parking spots. European cities have on average 37% and cities in Asia, such as Beijing and Tokyo, have 5% or even less. Cities with more on-street parking spots are generally more interested in smart parking solutions. A smart parking system can be defined as a parking solution that helps drivers to find a vacant parking spot. One of the most important reasons for this interest is to avoid drivers cruising for free parking [6].

In addition, cities use smart parking systems in an economic initiative basis. First of all, such systems can reduce the time spent by drivers to find a parking spot, which can help reduce environmental pollution, fuel consumption, and traffic congestion. Second, if drivers can rapidly find a parking spot, the idle time for parking spots can be shortened, thus increase the parking revenues. In addition, smart parking systems can help to detect unauthorized parking that can issue in a penalty charge. Finally, due to fewer vehicles cruising for parking spots, the flow of traffic may be increased as well as urban mobility, which increases the capacity of the city. This mobility can expand the possibility of higher population, and more activities and job opportunities in urban areas [6].

Today, there are multiple proposed systems that are being utilized for finding vacant parking spots, where small sensors are placed on each parking spot or cameras are positioned next to them, keeping track of vacant parking spots [7] [8] [9] [10] [11]. However, these solutions require a power source and often maintenance of each of the sensors. Each solution has some benefits and drawbacks based on the sensor technology that is used to determine the state of the parking spots.

This thesis is performed in collaboration with the Norway based company Atea. Today, Atea has some sensor-based parking solutions and has an interest in a solution that could detect vacant parking spots and unauthorized parking on a parking lot. This is due to the demand from customers that seek a more efficient solution for detecting vacant parking spots and unauthorized parked vehicles on a parking lot. The contribution of this thesis is a proof of concept that shows

a smart parking solution that can detect vacant parking spots and unauthorized parked vehicles can be developed.

1.1 Smart parking solutions

The concept of smart parking can be developed in different ways, using either sensors for measuring if the parking spot is vacant or vision-based solutions for processing images and detecting vacant spots. However, there are always some drawbacks with the one and other developed solutions. To build a system that achieves high accuracy and precision requires to find the most suitable technique, thus trying to construct a low-cost and high performing solution. Below, we provide a broad overview of existing research of smart parking solutions, where we categorized the solutions based on the sensor technology used to detect vehicles.

Ultrasonic Sensor

Today ultrasonic sensors are commonly used for a wide variety of applications, such as non-contact presence, motion detection, proximity, or distance measuring. Some of the main application areas for ultrasonic sensors are water and wastewater, mining, general industry, chemistry, and petrochemistry [12]. Ultrasonic technology is among the most preferred technologies for smart parking systems.

Lookmuang et al. [13] propose and develop a prototype using an embedded controller, Raspberry Pi 3, ultrasonic sensor, and a camera, with the aim of localizing vacant parking spots. The ultrasonic sensor and camera are used to detect when a vehicle approaches the parking spot, in order to communicate to the cloud with an image of the vehicle registration plate.

Similar approaches are contributed by Vakula et al. [14] and Grodi et al. [15]. In particular, Grodi et. al. discuss the advantages and disadvantages of multiple sensor types, such as induction proximity, RFID, light/ranging detection, and camera. In both cases, the authors chose to use an ultrasonic sensor for detecting when a vehicle was parked on a parking spot. The choice of ultrasonic sensor solution was based on its low cost.

Another ultrasonic sensor based system that provides a novel parking management solution for various types of parking facility areas is presented by Sadhukhan [16]. Detection of vehicles is done by using an ultrasonic sensor; however, this system also includes cameras. After detecting the vehicle, a camera placed close to the parking spot takes a snapshot in order to extract the licence plate number from the image. This system contains a server which is responsible for image processing and all logical operations based on the sensor values. Drivers are able to display the status of parking lots by using a web-based GUI. The users of the system can make a reservation for a parking spot and make payment based on the duration of the reservation.

In each of the studies discussed above, the authors relied on using ultrasonic sensors. In summary, the ultrasonic sensor in each of these presented solutions performed good enough, but it is not possible to predict if the detected obstacle is a vehicle or not. In the first paper [13], Lookmuang et al. chose to use an additional module, a camera, in order to increase their accuracy of observing the object which was standing in front of the sensor. To conclude this, considering the use of multiple, different sensors may give a higher accuracy of the system in general.

RFID

The RFID technology has been widely used in industry, electronic commerce, credit-cards, ID-cards and so on. Today, RFID is one of the key core technologies for the development of the Internet of Things solutions [17]. In particular, one of the application areas for RFID technology is parking solutions. This technology has been used for vehicle detection and below we present some of these solutions.

Pala et al. [18] present a smart parking solution prototype that contains RFID readers and servers. In this solution, each parking spot has an RFID reader and a barrier, where the readers are connected to a local server. In order to use this solution, the owner of the vehicle needs to have an RFID label on the vehicle. This RFID label is registered in a database with the information of the owner and vehicle. RFID labels were managed to be read from approximately 7.62 cm distance. When a registered vehicle arrives at a parking spot, the readers read the RFID label. If the vehicle is registered, the barrier will lift off for the vehicle to drive in. All check-in and check-out information are stored in a database in order to see the status of parking lots in the city.

Mainetti et al. [19] present another smart parking system, which combines different IoT technologies, such as ultra-high frequency RFID (UHF RFID), Wireless Sensor Network (WSN) and Near Field Communication (NFC). In the implementation, each vehicle has an RFID tag/label and an RFID reader detects the vehicle when the vehicle enters the parking spot. Unlike previous research, the used RFID tags are UHF RFID tags which have a longer reading range compared to normal RFID tags up to 8 meters. RFID readers send the information about the vehicle to a WSN node, which continuously updates a smart gateway about the vehicle's presence and store the data in a database. Users and traffic police are allowed to see the status of parking spots. Additionally, each vehicle has an NFC tag that is used for identifying the drivers by using a smartphone. If the system detects unauthorized parking, a notification is sent to a mobile phone application that is designed for the traffic police. Another mobile application allows drivers to find a vacant parking spot and make payments.

Magnetic sensor

The use of magnetic sensors can be very useful when developing a system for detecting vacant parking spots. Since the magnetic sensors rely on the magnetic field of the Earth, it can detect when there are some anomaly changes, such as when a car is parked over the magnetic sensor.

A parking system that uses magnetic sensors for detecting vacant parking spots was developed by Zhang et al. [20]. The proposed system utilizes a magnetic sensor for detecting the change in the Earth's magnetic field, whether to know if a vehicle was parked on the parking spot or not. As the use of magnetic sensors has a high risk of giving a fluctuating interval of values, disturbance might occur from neighbouring parking spots. In order to handle the fluctuation of sensor values, two different algorithms are used whether the parking spot is empty or not. Zhang et al. identify that the battery life of the sensors is a potential drawback of the solutions based on magnetic sensors. Therefore, there is a need for putting the device into sleep mode, which gives an increase in battery life.

In a similar system, Sifuentes et al. [21] state that the number of cars that have been parked on a parking spot during the day could have a major impact on the battery life of a device. This is due to the amount of time the device is kept in sleep mode. Based on calculations, a parking spot that

is during a day only being parked on 10 times will have a battery life of 33,5 years, while 4320 times will result in a battery life of 1,5 years. However, they state that a common case would be around 500 detections in populated cities. A Light dependent resistor (LDR) is used for waking up the device whenever a vehicle was parked over the magnetic sensor.

Furthermore, the value of a magnetic sensor varies when different types of cars are parked and then removed from the parking spot. In addition, other factors such as dirt can cause the value of the magnetic sensor to fluctuate.

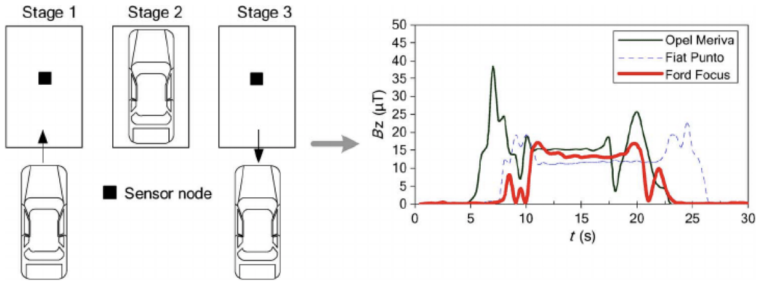

Figure 1: Magnetic sensor values from three different vehicles [21].

Camera

Although sensors are widely used for vehicle detection they have some drawbacks. One of the main drawbacks is scalability, since the need to use one sensor per parking spot may be an expensive solution. To overcome this issue, vision-based systems can be used for vehicle detection [11]. This type of systems can provide a more scalable solution than a sensor-based system. A camera, which is placed in a location that covers a wide view of a parking area can be used to determine the status of several parking spots.

The work done by Patane et al. [22] is an example of a vacant parking spot detection system that makes use of a vision-based system. The system consists of three main parts. Wide angle cameras are used to capture a video stream from which images can be extracted. The captured stream is sent to an embedded box located relatively close to the camera. All computation for image analysis is performed in this box and information about the parking status is sent to the cloud in order to be viewed by users. According to the authors, the resolution of the images should be at least 50px per side for each parking spot. Patane et al. tried to measure the detection reliability of available parking spots by using three different convolutional neural networks (CNN) models: AlexNet, GoogLeNet, and VGG16. To compare the performance of these three different networks, the authors used a dataset with 12471 images with a resolution of 1280× 720 pixels from three different parking areas. Images in the dataset were taken during summer, autumn, and winter at different heights. The results show that all networks perform well, with an accuracy of 97% or more.

A similar approach was contributed by Amato et al. [23], where a collection of approximately 150 000 images were trained for classifying vacant parking spot. The achieved accuracy with the trained dataset using mAlexNet always gave a result of above 90 % when using mAlexNet on a Raspberry Pi 2. In order to get a vast majority of different environments, a dataset for visual

occupancy detection of parking spots (CNR-PARK) was used, where different weather conditions existed in the dataset, i.e., sunny, rainy, and overcast. The training of the model with different weather condition images helps to get an enhanced prediction during the real-time test since the model is trained to understand the different weather conditions. The collection of images from 9 different cameras in different angles, gave a better result when comparing it with another dataset provided by PKLot [24]. In Section 4.2, we provide a deeper explanation of the deep learning models used for vehicle detection.

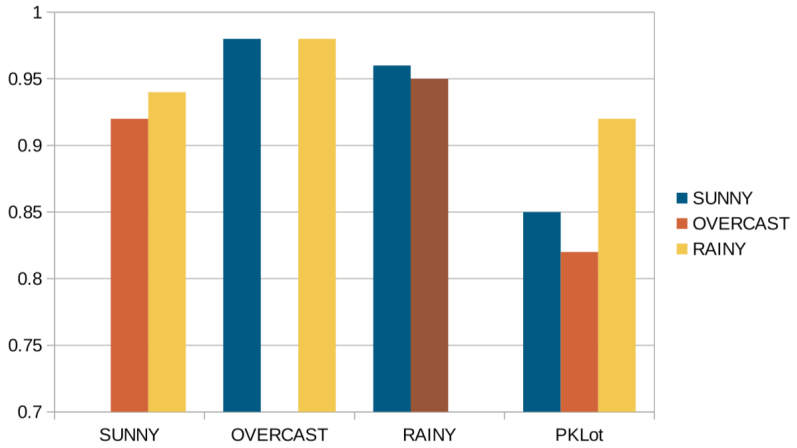

Figure 2: Accuracy result comparison between the two trained dataset (CNRPark-EXT and PKLot) in the approach by Amato et al. [23]

1.1.1 Summary of smart parking solutions

Based on previous research that is presented in this section, smart parking solutions can be divided into two categories: sensor-based and camera-based solutions. The use of sensors for detecting vacant parking spots can provide a relatively fast and fairly accurate result. However, having a great number of parking spots requires installation of sensors on each parking spot. This requirement is one of the main drawbacks of sensor-based solutions; installing and maintaining one sensor per parking spot may be very expensive and time-consuming, in particular for large parking lots. In particular, the cost required for installing and maintaining all sensors might be higher than using one or a few cameras for covering a large parking lot with multiple parking spots. This particular challenge is mentioned by Vakula et al. [14], who states that the maintenance cost is one of the main drawbacks with the use of sensors. In addition, additional costs should be considered if an internet connection is needed for each of the sensors, and also in order to secure the sensors from theft. On the other hand, when looking at the software complexity, it can be much higher for a vision-based solution, since using a camera-based solution requires that a deep learning model is trained accurately to detect vehicles/objects. Furthermore, the camera requires that it is constantly connected to a power source, which requires a higher power consumption than a sensor based solution.

In order to evaluate the smart parking techniques, a summary of the common characteristics of smart parking solutions were derived in table 1. The estimated characteristics presented in the table below may vary depending on the number of parking places that the smart parking system

covers. In small parking areas, e.g., less than 10 parking spots, sensor-based solutions can be a better alternative compared to the vision-based solutions in terms of te total system cost. Big parking areas, i.e., more than 30 parking spots, is considered when creating the table below.

	Ultrasonic Sensor	**RFID Sensor**	**Magnetic Sensor**	**Camera**
Installation Cost	HIGH	HIGH	HIGH	LOW
Maintenance Cost	HIGH	HIGH	HIGH	LOW
Hardware Cost (Per unit)	LOW	LOW	LOW	MEDIUM
Power Consumption	LOW	VARYING	VARYING	HIGH
Relocation Possible	HARD	HARD	HARD	EASY
Hardware Complexity	HIGH	HIGH	HIGH	LOW
Software Complexity	LOW	LOW	LOW	HIGH
Scalability	LOW	LOW	LOW	HIGH
Reliability	HIGH	HIGH	HIGH	HIGH
Vehicle identification	NO	YES	NO	YES
Obstacle Recognition	YES	NO	VARYING	YES
Vehicle Recognition	NO	YES	YES	YES
Additional Requirements	-	Privacy & all vehicles need to have an RFID tag	-	Privacy and Security issues

Table 1: Comparison of different parking detection techniques

The use of RFID sensors is a common approach for vehicle detection. The main drawback of this approach is that all vehicles must have an RFID tag. Another problem with RFID technology is the reading range. A standard RFID tag can be read from a distance of approximately 7-8 cm. By using UHF RFID tags this value can be increased up to 8 meters, however, at a cost of increased power consumption.

Ultrasonic and Magnetic sensors are also used in smart parking solutions for vehicle detection. The common drawback with these detection techniques is that it is not possible to determine whether the detected object is a vehicle or not. More importantly, a smart parking system that requires to identify illegal parked vehicles/objects would need several sensors on each parking spot.

The use of cameras could provide more advanced information, such as getting license plate numbers for automatically handling the check-in when parking or notifying when there is a vehicle/object that is unlawfully parked, hence inhibiting other vehicles to park. In addition, connecting each sensor to a network could create a higher complexity than using one camera connected directly to a local area network (LAN). As mentioned by Nyambal et al. [25], cameras can provide a budget solution from a time and cost perspective, where the need for maintaining the camera-based solution are not required as much as for the sensor-based solution.

Lastly, the privacy aspects are of importance in particular when building camera-based parking applications, where checking in of the parking is needed. Moreover, storing information about the parking status of an individual person can be troublesome, since the storing should only be performed when necessary, and only kept as long as required, due to the GDPR legislation

[26]. In addition, since most of the parking lots usually are located in public places, privacy and security should be taken into account.

1.2 Purpose and Motivation

As mentioned in the previous section, smart parking solutions can be divided into two categories: sensor-based and vision-based. A vision-based system might be advantageous from scalability and cost perspective. Additionally, vehicle recognition and obstacle recognition are important features that can be provided by vision-based systems. These features can be used in order to build a more sophisticated system.

Many vision-based parking solutions have been proposed in previous research, providing information about the state of the parking spots with high accuracy. A common characteristic among current vision-based smart parking solutions is that they have static structure, which means that these solutions require preparation during installation, i.e., each parking spot position needs to be predefined in the system. This feature adversely affects the relocation possibility of vision-based solutions. A vision-based solution, which has a more dynamic structure, can facilitate the installation and use of such a solution. What is meant by a dynamic structure, may be a system that can define all the parking spots based on the position of the vehicles and the frequency of the presence of a vehicle in a particular area.

Additionally, there is infrequency of research about detecting faulty parked vehicles. The performed research in vision-based systems focus mainly on detecting vehicles, thus does not take violated parking into consideration.

The objective of this thesis is to develop a prototype of a dynamic smart parking system in order to detect whether there is any vacant parking spot on a parking lot and if there is any vehicle that is parked in an unauthorized manner. By using a smart parking solution, information about available parking spots can be provided to users through, e.g., an application, as mentioned and discussed in these papers [16] [13] [15] [14]. In addition, a system which could detect illegally parked vehicles would help to solve parking violation issue thereby the efficiency of use of parking spots can be improved by either informing the owner of the vehicle or a parking enforcement officer. The main purpose of this study is to contribute with a sustainable parking solution in urban environments in order to increase the traffic flow and decrease traffic congestion.

1.3 Research Questions

To achieve the goals of this thesis, we have formulated the following research questions:

RQ1 What is an appropriate solution to dynamically detect vacant parking spots by using a vision-based solution?

RQ2 How can a vision-based smart parking system detect illegal parked vehicles?

1.4 Limitations

This research will be limited to developing a system prototype which will be evaluated on only one parking lot. This is due to legislation of installing a camera on public places. Furthermore, the system will not be tested and evaluated in different weather and light conditions, due to the short time period of the study.

2 Theoretical Background

This section describes the theoretical background, with the aim of contributing with a comprehensive theoretical insight related to our study.

2.1 LoRaWAN

Long range wide area network (LoRaWan) is an open protocol, developed by LoRa-Alliance, which regroups more than 500 members. The LoRa-Alliance describes LoRaWAN [27] as:

> "LoRaWAN is an LPWAN specification intended for wireless battery-operated Things in a regional, national or global network. LoRaWAN targets key requirements of the Internet of Things such as secure bi-directional communication, mobility and localization services. The LoRaWAN specification provides seamless interoperability among smart things without the need of complex local installations and gives back the freedom to the user, developer, businesses enabling the rollout of the Internet of Things."

LoRaWAN defines the communication protocol and the system architecture for the network while the LoRa physical layer enables the long-range communication link [27]. The protocol and the network architecture are important factors in determining the battery life of the node, the network capacity, the quality of service, the security, and the variety of applications served by the network.

LoRa (Long Range) is a wireless communication technology which has been developed by Semtech [28]. LoRa is mainly targeted for the machine to machine (M2M) and IoT networks. The main key features of LoRa are a long-range capability, low power consumption, and high robustness. A single gateway may cover the entire city or hundreds of square kilometres [27]. Frequency shift keying (FSK) modulation is used as the physical layer for LoRa. It uses a very efficient modulation in order to achieve low power consumption. LoRa utilizes a spread spectrum technique called Chirp Spread Spectrum, which maintains the same low power characteristics as FSK modulation but this technique significantly increases the communication range [27].

2.1.1 LoRaWAN network topology

A LoRaWAN network contains at least an end device, a network server, and a gateway. In a LoRaWAN network, the nodes are not associated with a specific gateway. A single end device can transmit data to several gateways. The data that is received by the gateway, is forwarded to the network server. Each gateway will forward the received packet to a network server via TCP/IP, which means that the gateway must have access to the Internet. Afterwards, the network server will forward the data to an application server [27].

LoRaWAN network is a star-of-stars network topology, where a central server is a root or the centre of the network. One or multiple gateways are then connected to the central server, creating a network with a star layout. Furthermore, each gateway has its own star-network, where the gateway is the central node and end-devices connect to it. It results in a star-of-stars topology as illustrated in Figure 3 below.

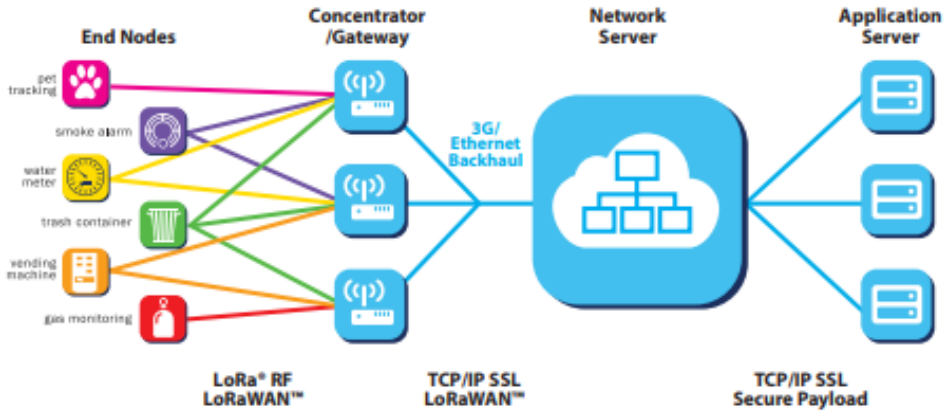

Figure 3: LoRaWAN Network Architecture [27]

2.2 Machine learning

Today, machine learning is used in many areas, such as image processing, speech recognition, and robotics. Machine learning (ML) is a subset of the broader term called Artificial Intelligence (AI). In machine learning, the goal is to build machines that continuously learn by interpreting the collected data in order to discover and understand patterns in the data. There are different types of machine learning methods, which supervised learning and unsupervised learning are the two main types. During the training process in ML, the data is processed by applying an algorithm and classifying the objects. The training process generates a model that can be used to get a prediction of the same type of data that was trained on. For example, this can be to train a fall detection system, which requires data of several different falls. When the model of the different falls is created, then the model can be used in real time to predict if a user falls [29]. In certain applications, the efficiency and time complexity of the algorithm is as important as its accuracy. Some of the machine learning applications are learning associations, regression and unsupervised learning [30].

The aim of unsupervised learning is to learn regularities from the input without using external information from a supervisor. By detecting certain patterns that occur more often than other patterns, it is possible to understand what happens in the input data. That is called density estimation in statistics [30]. One common method for density estimation is clustering and more information about this topic is given in the next section.

2.2.1 Clustering

Clustering is commonly used within ML, and it is useful for classifying objects that are unlabeled. Clustering is an unsupervised learning method, i.e., data that is not labelled, thus the clustering algorithm labels the data by adding similar data together in the same cluster. This is possible by calculating the distance between the different data points and afterwards associate similar data together. Moreover, clustering is useful for detecting anomalies in the data, preventing the clusters from being affected by outliers in the data that might have been collected due to a measurable error [29][31].

2.3 Deep Learning and Neural Network

Deep learning is a subset of Machine learning (ML), where it uses neural networks with multiple layers to learn from patterns. Deep learning is commonly used for applications, such as speech assistant for understanding a human being, or in image recognition for labelling objects in an image. Deep learning is designed to replicate the human brain where multiple neurons can be used for training and understanding the world. This can be exemplified with the child that requires a lot of training until the child can distinguish between the different alphabetic characters and write on his/her own. The neural network in deep learning consists of three main layers: the input layer, the hidden layer, and the output layer. The input layer consumes the information, in the case of image recognition this would be the image data. In the next step, the nodes are weighted in the hidden layer and then processed in the output layer, which return the classification result based on the weights [29]. A representation of a neural network can be seen in the Figure 4 below.

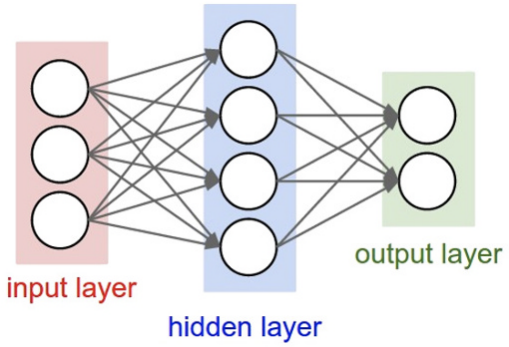

Figure 4: Representation of a neural network [29]

2.3.1 Convolutional Neural Network

Convolutional Neural Network (CNN) is a deep learning algorithm, which is a subclass of the neural network. The CNN architecture is used mostly for detecting objects in images, due to it is constructed to handle large sized images, which the ordinary neural network does not handle. This is possible due to that the CNN architecture handles the images as volumes, consisting of depth, width, and height. When the data transfers between the input layer to the hidden layers, data is being processed and filtered between the layers by using neurons, which try to find patterns in these images [32].

Data training is an important requirement for the CNN approach. Before starting the object detection, a data-set that contains images of the desired object must be collected. The objects in images should be defined by labelling each object separately. After the labelling process, the data-set is ready to be trained. During the training, the data is filtered in the hidden layers in order to find and specify what similar patterns all the images share together. These patterns are parameters that provide a specific description of the object/objects; the parameters which helps to learn how the specific object looks like.

2.3.2 YOLO

You only look once (YOLO) is an approach for object detection which uses an convolutional neural network in order to detect objects and classify what type of object is present in an image. YOLO partitions the whole image into smaller grids and successively search for connections between the different grids to detect objects and find the bounding boxes of the whole object. During this process, multiple bounding boxes are predicted on an object; however, each bounding box has a score which later on is confined into one box for each object with the highest score. Lastly, when the object is found, a classification is made on the object in order to tell what it is [33][34].

2.4 Image Processing

The idea with image processing is to interpret an image in order to extract essential information needed for the desired purpose. This can be compared to the human eye and the perception, which help the human to get a visual image of the world and understand the state of an environment. The aim with image processing is to replicate the same function, but instead use it in machines. Today, image processing is widely used in different areas, such as inspecting faults in electronic circuits, medical diagnosis or object tracking for surveillance purposes. Image processing can usually be seen as a three-step approach where it starts by inputting the image data, then processing the image, and thereby outputting the extracted information needed for further processing [35].

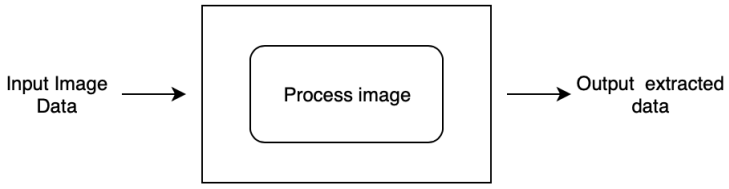

Figure 5: The Three basic steps in image processing

3 Research Method

This section describes the research method used in order to answer the research questions presented in Section 1.3. The chosen approach for the thesis is design and creation, which is suitable for prototype development.

In order to carry out the thesis, we will develop a prototype by following an iterative research methodology. This yielded in choosing the Nunamaker's & Chen's methodology due to its systematical development approach, which is suitable for building and evaluating prototypes [36]. Another reason for choosing this method is that it has an iteration ability, where previous steps of the method can be changed and updated. This feature gives us an opportunity to make improvements and changes during the development process. The workflow for our research method is illustrated in Figure 6.

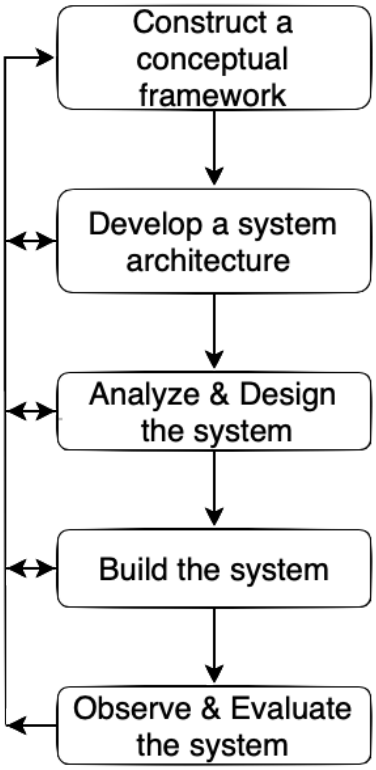

Figure 6: Representation of Nunamaker's & Chen's workflow [36]

3.1 Construct a conceptual framework

The first step of the method consists of defining the problem area, gathering knowledge around the problem, and deriving the research questions. Furthermore, a literature review is conducted to analyze the related work and how it could contribute to this thesis. The literature review helps to identify the gaps that needs to be filled in order to make a contribution within our research field. Additionally, the literature review will help to identify what prevalent solutions

are available today regarding smart parking systems.

Moreover, the literature review will also help to draw a conclusion, based on the criteria, on what type of technology is more efficient when building a smart parking prototype. Lastly, criteria are derived in order to evaluate the different techniques, i.e., what requirements should be considered when building a smart parking prototype. In summary, defining the problem area, identifying the gap and extracting the research questions were performed in this step. Moreover, a requirement specification was defined for the prototype which is presented in Section 5.1.

The databases that were used for the literature study are IEEE Xplore Digital library, Science Direct, and ACM Digital library. The main reason for using these databases is that they are considered reliable sources.

3.2 Develop a system architecture

The next step in the research methodology is to create a system architecture. By defining the system architecture, this step aims to provide guidance during the development process of our prototype. The system architecture describes the various system functionalities that needs to be captured in our prototype. In this step, the requirements should already be defined so that they are measurable and can be validated in subsequent steps in this research process. Based on the requirements, system functionalities and relationships between different parts of the system was specified. A problem breakdown tree was created in which the main functions were defined and then divided into partial functions, as can be seen in Section 5.2.

3.3 Analyze & Design the system

The third step is to design the prototype, i.e., what parts of the prototype are needed in order to have all the required functionalities based on the requirements. This stage is performed iteratively, where the system is analyzed whether it fulfills all the functional requirements. At this stage of the research methodology, some of the relevant components for the prototype were chosen, which includes both a hardware and a software part. The data transmission technique and visualization platform are determined in this step. The interaction between different components and software architecture were designed. This system design was illustrated by using suitable diagrams, which is illustrated in Section 5.3.

3.4 Build the system

The prototype implementation depends on the design that is determined in the previous step 3.3. The prototype design gives insight into what problems that may occur, which is a provided feature in the research methodology. Thus, it is possible to go back to the previous steps and re-design when using this research method.

This step covers the development of software and connecting different components based on the system architecture, which is mentioned in Section 3.3. For image analysis, it is very common to collect data (images) and create a model based on the collected data in order to reach an accuracy of high level. Therefore, we may need to collect data to be used in our prototype for training a model. However, collecting a new dataset in order to train a model could take a long process. Vehicles are common objects and there are several public datasets that contain thousands of images. Additionally, many people have trained these datasets with different object detection

models and shared their results. Instead of collecting a new dataset and training a new object detection model, we instead decided to use some of the available pre-trained models and choose one of them based on the performance on their vehicle detection accuracy.

3.5 Observe & Evaluate the system

Once a prototype has been built, the prototype needs to be observed and evaluated whether the result of the prototype fulfills the requirements. We will experimentally evaluate our prototype in order to identify if the prototype fulfills the requirements that are defined in Section 5.1.

One important metric that will be evaluated is the accuracy of determining the parking lot status. In order to measure this feature of the prototype, suitable test cases should be defined. All defined test cases should be performed and all evaluation results should be available for the readers. Some evaluation results may fail or inconclusive on the first run, due to bugs, which will be needed to be fixed. The iteration ability of this research method may be advantageous in order to find the problem, correct errors and re-run the failed test cases. Observation and evaluation the prototype confirms that the prototype fulfills the requirements but more importantly the result of this stage will help to answer the research questions.

4 Literature Review

This section mainly aims to give an insight into the current state-of-the-art, regarding which smart parking solutions exist today and what are their advantages and disadvantages. We considered the smart parking concept as a smart city application, where different solutions are presented based on the used vehicle detection technique. Continuing, a more comprehensive description is given on what deep learning techniques have been used together with vision-based smart parking solutions. Lastly, we explain which aspects of GDPR should be considered when building a vision-based solution that requires the handling of sensitive information. In order to conduct the literature review, we searched on IEEE, ACM, and Science Direct, with different terms, e.g., IoT AND Smart parking AND camera, IoT AND smart parking, smart parking AND camera, smart parking, and smart parking AND deep learning.

4.1 IoT & Smart City

Technology is something that today drives the development of a brighter future, where it has contributed to positive outcomes in different areas such as health care, industry, and education. A concept that has gained a lot of momentum and attraction in the past few years is the Internet of Things (IoT). IoT is described by Pureswaran et al. [37] as a network of devices connected to a centralized cloud, with the aim to become a distributed many-to-many connection approach.

The number of devices connected to the Internet is estimated to be more than 25 billion devices by the year 2020 [37]. The reason behind this is mainly the increased number of devices today that are contributing to the smart home and city concept. Today small and cheap sensors are used to automate different processes, e.g. automating the process of ventilation in buildings by measuring the temperature and humidity, or in smart parking solutions for automating the detection of vacant parking spots [38].

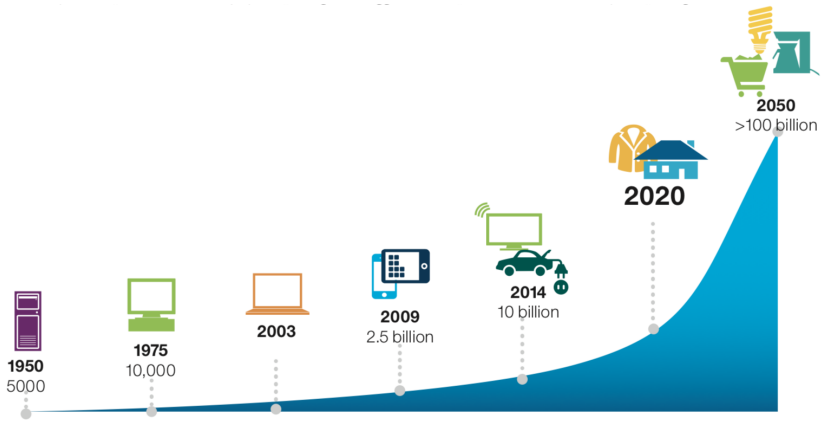

Figure 7: Prediction of the amount of connected devices [37]

Lazaroiu et al. [39], describes the definition of the smart city as something that is dependent on the support from the ”information and communications technologies”. In other words, they mean that a smart city concept can be achieved by integrating smart solutions in different areas, such as electricity, waste management, gas and water consumption, mobility, and public safety.

AlHarbi et al. [40], mention that smart cities contribute with improvements for the cities by solving challenges and contributing with a sustainable environment. Further, the authors state that such improvements for the smart city can be contributed in smart parking by, e.g., decreasing the pollution and traffic congestion or managing the availability and providing guidance towards parking spots.

4.2 Object detection using deep learning

Despite the recent advances of deep learning, there is still much more to contribute and improve in this area. The use of deep learning in vision-based solutions has in the past years been widely used for recognizing objects. Deep learning works in a way that requires a trained model before being able to classify the objects [29]. The same approach can be related to a child, which require time for training until he or she learns the alphabet and distinguish between the different characters in the alphabet.

When handling a vision-based solution, where recognition is needed of an object, convolutional neural network (CNN) can help to effectively recognize patterns in the image. Amato et al. [23] describe that the huge amount of hidden layers in CNN are used in order to handle the input and provide a result as an output. A large number of hidden layers improve the detection accuracy when deciding what the recognized objects are, resulting in the output. Amato et al. further state that, training a model for object classification it can be very expensive from a computational and time perspective. This is due to the number of iterations needed for the network to find and understand the layers in the neural network, requiring a decent graphics unit to perform the training.

Redmond et al. [33] compare their model called YOLOv3 with other object detection models. The YOLOv3 model is an improved model from the previous models: YOLOv1 and YOLOv2. The purpose of the comparison was to evaluate the different models when it comes to the inference time and performance. The models were trained and tested on the same COCO dataset in order to make a fair comparison. The results from the comparison indicate that YOLOv3 has a slightly lower mean average point (mAP) than RetinaNet by 4.3 points. On the other hand, YOLOv3 has a lower inference time than all the other models, which would be an advantage when running locally on an edge device with limited processing power. This small trade-off can be considered acceptable since the difference in mAP is very small while the inference time is much lower.

A similar study was done by Du [41], who compared and evaluated an older version of YOLO called YOLOv2 with other CNN algorithms such as Fast Region-Convolutional Neural Network (R-CNN). The R-CNN algorithm uses the whole image in the layers of CNN when detecting objects by applying region based search in order to get a higher predicting speed while sacrificing the accuracy. On the other hand, YOLO partitions the image before running it through the layers in CNN and assembles the whole image again to keep the performance and speed intact. The results from this comparison indicate that YOLO performs better in both prediction speed and mAP. The overall scoring indicates that YOLO is robust and fast, which is also stated by the author. Moreover, the author mentions that YOLO is suitable for real-time detection which is a requirement when there is a need for fast decisions. In addition, the use of bounding boxes for detecting the objects in YOLO makes it easier to tell where the objects are in the image. Most of the CNN models can only detect that there is an object in an image but does not specify where the objects are located.

4.3 Vision-based smart parking solutions

Amato et al. [42] present an approach for real-time car parking occupancy detection that uses a CNN classifier. According to the authors, in order to achieve efficient use of parking areas the techniques for parking occupancy detection is an important factor and there is a lack of generalization when applied to different parking areas in vision-based systems. The proposed system is based on a deep learning technique, which is known as CNN. CNN differs from other neural networks due to the presence of convolutions layers. All image processing tasks are performed on a Raspberry Pi 2 model B which is mounted in a camera box. Their system periodically captures images of the parking area. Each frame is divided into patches where each of the patches contains a single parking space. The system then classifies each patch using a trained CNN in order to decide if the parking spot is occupied or not. The authors make use of two different CNN architectures, mAlexNet and mLeNet, which they evaluate using the two datasets PKLot and CNRPark. The result indicates that mAlexNet performs better than mLeNet. Using mAlexNet, they achieved an accuracy of 83% on CNRPark and 90% on PKLot. Moreover, the research show that CNN architectures are effective and have high accuracy, even in bad light conditions, and where there is shadows and partial occlusions.

Carrara et al. [23] describe a decentralized and efficient solution for visual parking lot occupancy detection based on a deep CNN specifically designed for smart cameras. The main contribution of this paper is a publication of a CNRPark-EXT dataset. This dataset contains 150 000 labelled images of free and occupied parking spots, where images are taken in different light and weather conditions. The authors use two cameras when collecting data for training in order to have different angles, perspectives and build a mask that allows cropping the full images into patches. Each patch contains only one parking spot. The authors used two CNN architectures, mAlexNet and AlexNet. These CNN architectures were trained with their own dataset (CNRPark-EXT), and another dataset called PKLot. The authors mention that AlexNet performs slightly better than other CNN architectures. The solution is tested in a parking lot which consists of 164 parking spots. Carrara et al. used 9 cameras when testing the system in order to cover the entire parking lot. Images captured by the cameras are filtered by using a manually built mask in order to identify each parking spot. Then, each patch that contains an image of a single parking spot is classified using the trained CNN. According to the achieved result of this study, the proposed solution has a good generalization capability and performs better than other state-of-the-art solutions.

Another similar solution is presented by Nyambal et al. [25]. The used dataset contains 782 images from two parking lots at the University of the Witwatersrand. Each image is cropped into patches that contain a view of a single parking spot. Coordinates of the parking spots are stored in a JSON file. For the experiment, the authors trained the dataset with two CNN architectures, AlexNet and LeNet. Their system captures a frame from a video stream or form a recorded video, and crops the frame into patches based on the predefined coordinates that are stored in a JSON file. The classifier that is built using the LeNet architecture achieved an accuracy of 93.64% and the AlexNet architecture produces an accuracy of 95.49%. During the research, the authors noticed some noise due to light conditions and some elements stuck on the ground. This noise is corrected by improving the dataset based on the output of false detection due to noise.

Valipour et al. [43] present another approach of a vision-based smart parking solution. The proposed solution consists of three components, i.e., a camera, a server, and a front-end application. The camera captures images from a parking lot and sends them to the server through either a local wireless network or over the Internet. The server consists of a database, a detection module,

and a web service. The server is responsible for collecting images from the camera, sending them to the detection module and storing the detection result in the database. The third component is the front-end application that presents the information about vacant parking spots. The server uses a CNN for the detection task. The chosen network is VGGNET-F, which is trained using the PKLot dataset. The Server crops the full image into patches and classifies each patch by using VGGNET-F. The front-end is a smartphone application that requests data from the web service. As a result of this study, the authors claim that their solution performs 8.13% better than the state-of-the-art solutions. The complete system is tested on a public parking lot and showed to have a robust performance.

4.4 Cameras & GDPR

Today, cameras are mostly used for security and safety purposes, e.g. to detect if unauthorized people might have entered a restricted area [44]. Cameras have in the past years been developed to handle more advanced tasks, such as processing images and extracting useful information [45]. Consequentially, in most cases, the information extracted in images contains sensitive information, which could be considered an ethical problem. The policy on how data should be handled and what restrictions there are when it comes to storing personal information have been introduced more specifically with the recent General Data Protection Regulation (GDPR) [26].

Vojković [46], discuss the effects of GDPR and how it could slow down and implicate the development of the smart city concept. The focus of GDPR is mainly on the handling of data, how long the data is allowed to be stored, and what is the purpose of storing personal information. As the author describes, the storage of the data needs to serve a purpose to why the collection of data is needed. Furthermore, the data can not be stored longer than needed; when there is no need for the data, the personal information must be immediately removed. Vojković further describes that the personal information can be identified both directly and indirectly. Objects that can be related to the individual person are considered personal information, e.g. an owner that parks her/his car on a parking spot can be identified by the car he/she owns, hence, camera images of parked cars is typically considered as personal information..

5 Results

In this section, we present the results achieved during our thesis work. In particular, we describe in this section the development process of the prototype by following the method that was presented in Section 3.

5.1 Construct a conceptual framework

In order to develop our smart parking prototype, the first step is to investigate the needed functionalities and requirements that are important for this type of system. Investigating these functionalities and requirements would enable to understand the development process of the complete system.

Installing a camera in a publicly available area requires a permit for camera surveillance. In Sweden, the installation of cameras in public spaces requires authorization from a governmental authority. In European countries, where GDPR is applied, sensitive information that is recorded, such as images taken by a camera, requires consent from the people that might be monitored by the camera. An important factor that should be taken into the account is this privacy issue that comes with vision-based solutions.

Sending images to a server or cloud through the Internet comes with security issues as well. In order to handle this security issue, some encryption process may be used. All of these limitations may affect the usability of the system in a negative way. Performing image processing tasks in the place, where the images are captured without storing or sending any image data through the Internet may help to avert these issues mentioned above. Hence, the privacy and security aspect should be considered as a requirement of a smart parking solution.

Elimination of the obligation to access the Internet improves the relocation ability of the system, since Internet access is not available on most places. Due to this is a smart city application where there is only small amount of data that is transmitted, a long-range communication technology may be suitable for transmitting data.

As a result, we derived the following ”general” requirements on a smart parking system prototype based on these factors mentioned above:

Req1 The processing of images should be done locally without sending or storing any image data.

Req2 The prototype should be able to provide information about the number of vacant parking spots on a parking lot.

Req3 The prototype should be able to automatically define the parking spots without predefining their locations.

Req4 The prototype should be able to transmit data without having Internet access.

Req5 The prototype should be able to detect illegally parked vehicles.

Req6 The prototype should be able to adapt to environmental changes in the parking lot, e.g., changes in the number of parking spots, and the position of parking spots.

Some of these requirements will help to answer the research questions but not all of them. However, they are needed in order to put this study in correct context. A vision-based smart parking system may include different subsystems, e.g., cloud, visualization platform, and database. The focus of this study is to build a prototype that is responsible for capturing images, extracting the parking occupancy information from those images and sending this small sized information to a platform. Choosing the best data transmission technology, visualization platform and data storage are out of scope for our study.

We performed a literature review, where the current state of the art regarding smart parking was discussed, as can be seen in Section 4. The discussed smart parking solutions in the literature review focused on the static solution where each parking spot was predefined with their respective coordinates. Therefore, we targeted our focus on a more dynamic solution where none or only a few configurations of the system will be required when installing this system.

5.2 Develop a system architecture

Based on the conceptual framework presented in Section 5.1, we developed a problem breakdown tree in order to get a general overview of the different subproblems of the system. Each subproblem contains a set of tasks that are needed in order to solve the subproblem. To solve the main problem, we derived four subproblems: parking spot detection, data transmission, hardware choice and configuration, and integration. The main focus of this study is to solve the subproblem parking spot detection. The problem breakdown tree is illustrated in the Figure 8.

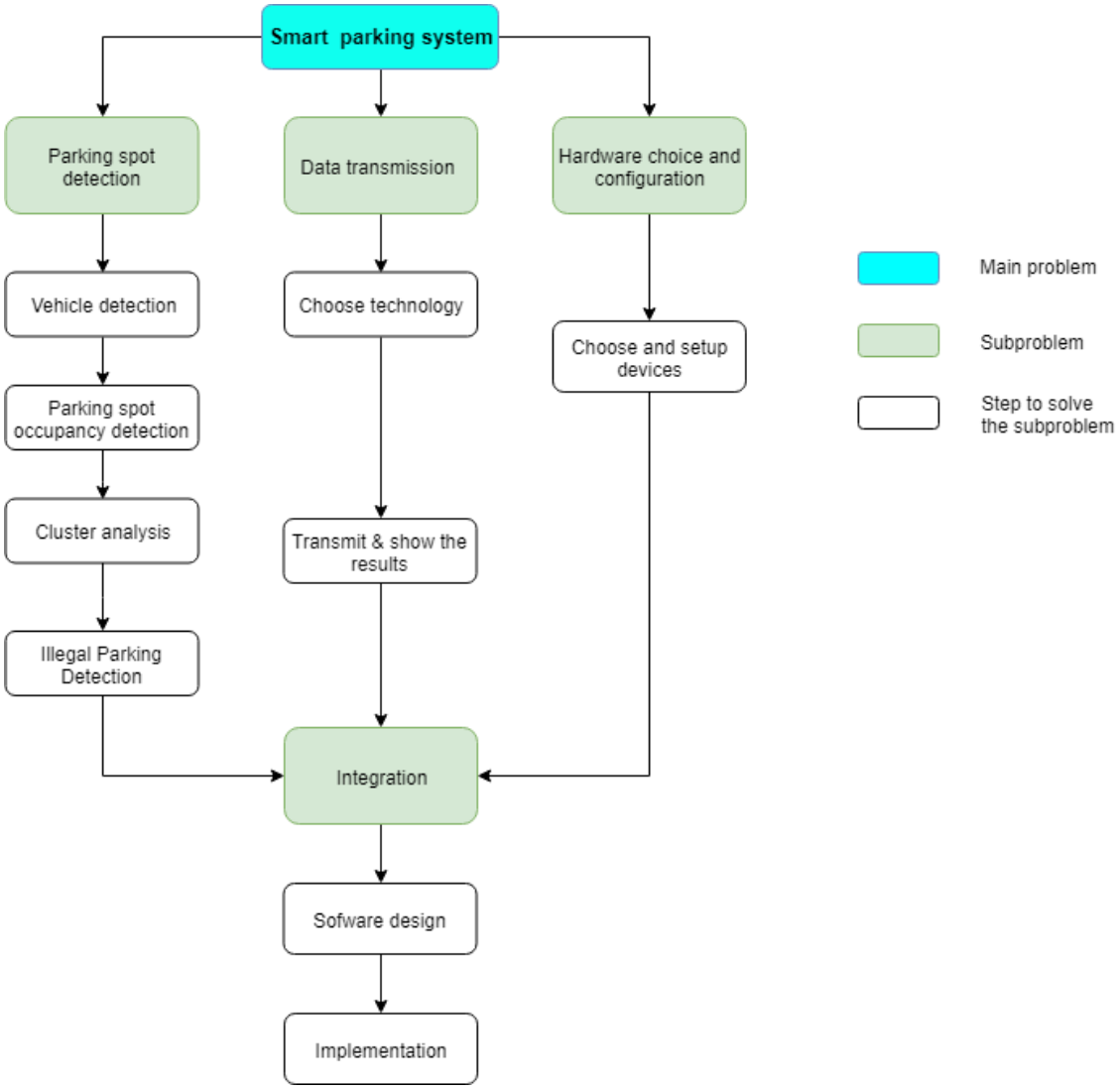

Figure 8: Problem breakdown

- **Parking spot detection**
  In order to build a dynamic smart parking system, parking spots should be detected automatically. This task can be performed by using an object detection algorithm. The chosen approach for detecting the parking spots is vehicle detection using deep learning technology. The chosen approach was based on its simplicity of detecting the parking spots on a parking lot. On a parking lot, the location of vehicles mostly indicates the location of

parking spots when being parked. Moreover, information about the vehicles' position may be useful in order to detect unlawfully parked vehicles.

Another approach that can be used for detecting parking spots is line detection. However, this approach has some drawbacks, i.e., that lines may be erased or blurred over time. We argue that to determine the state of the parking spots and to detect unlawfully parked vehicle this approach is not enough and that there is a need for combination with another technique. Therefore, the chosen approach is vehicle detection for detecting parking spots. There are different deep learning architectures that can be used for vehicle detection.

- **Data transmission** The purpose of data transmission in the system is to transmit the data to a server where the parking information can be registered. The transmitted data aims to give information about the status of a parking lot, i.e., how many parking spots are available and how many of these parking spots are unoccupied.
- **Hardware choice and configuration** The hardware used in this system consists of three different parts: the camera part, the computer part, and the transmission part. The camera's purpose is to capture images of the parking lot and send them to a single board computer. The computer's purpose is to perform the calculations and image processing on the images that is received from the camera in order to detect the parking spots and identify illegally parked vehicles. Lastly, a transmission device is used to transmit information about the parking lot to a server, e.g., parking lot id, how many available spots, and how many are unoccupied.
- **Integration** Once the parking spot detection, implementation of the chosen data transmission technique and hardware selection are done, we need to integrate these different parts into one system. Protocol for receiving images from the camera and sending data to the server must be determined. After the integration, the software that presents the main logic of the system can be implemented.

5.3 Analyze and design the system

In this section, we describe the analysis and design phase of our research methodology. The subproblems that is presented in 5.2 are analyzed and suitable solution for each subproblems is designed in this section. Furthermore, analysis of what object detection techniques to choose, what hardware to use and how the data will be transmitted is discussed and determined.

5.3.1 Parking spot detection

Vehicle detection

As mentioned in Section 5.2 an object detection model will be used to detect vehicles in the parking lot. There are several deep learning architectures that can be used to detect vehicles, some of which has been already used in other smart parking solutions. One of the most used deep learning architecture is AlexNet. This architecture stands out with its high accuracy when it comes to object classification [42] [23]. The meaning of object classification/recognition is finding out what kind of objects are in the image without giving any information about their location in the image. However, object classification is not enough to define parking spots. Positions of the vehicles are also needed to be known in order to define the parking spots. Therefore, AlexNet is not an appropriate architecture for our system.

There are several deep learning architectures that can be used to both detect and localize an object in an image. YOLO is one of the newest deep learning architectures which stands out with high accuracy and an extremely fast detection process. The fast detection process reduces the need for computing power and it is an important factor when all image processing is performed locally. These features indicate that YOLO is a appropriate choice for vehicle detection. As discussed in Section 4.2, there are different version of YOLO, where we chose to work with YOLOv3 since it is the most recent version of the YOLO architecture and contributes with an increased performance compared to the older versions of YOLO.

In general, the training of YOLO or any other deep learning architectures, requires a large number of images of vehicles, where all vehicles in those images are marked. Collecting a new dataset and training this dataset may take many days of additional work. However, vehicles are common objects and may exist in publicly available datasets. COCO is one of the public datasets that contains 328 000 images of common objects, e.g., car, motorbike, person, and train [47]. Instead of training a new model with the COCO dataset, we made use of an existing, pre-trained, YOLO model, which was trained using the COCO dataset. YOLO returns a bounding box of each detected object in the image as illustrated in Figure 9.

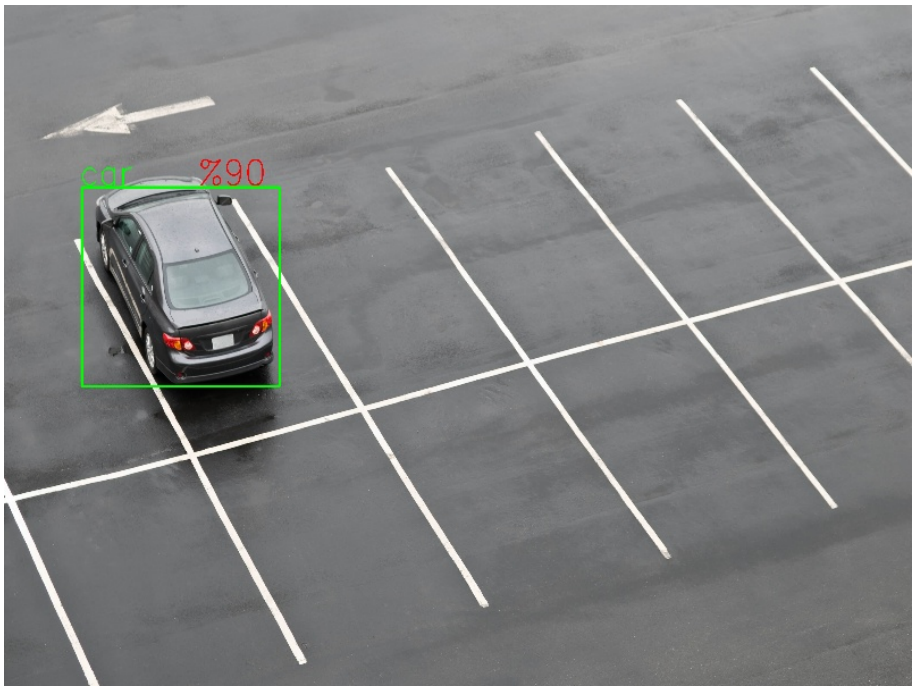

Figure 9: YOLOv3 object detection

As demonstrated in the figure, YOLOv3 detects a car in the image and returns the location, object type and confidence value of the detected object. The accuracy of object detection heavily depends on the dataset that is used for training. A dataset that contains images of vehicles that are taken in different weather conditions, different light conditions and different angles would probably give a better result but requires a long process. Therefore, creating a new dataset is out of scope of out study. Our experimental results show that the used model that is trained with COCO offers a satisfactory result.

The COCO dataset contains many images of different types of objects, e.g., person, bicycle, airplane, car, truck, and train. A filter can be used in order to return information only when the detected object is a car, truck, bus, or motorcycle. These are the main objects that can determine the vacancy status of a parking spot.

After the object detection process is performed, the system has information about the pixel location of each vehicle in the image. Initially, the system assumes that each of the bounding boxes represents a parking spot and stores the location of these bounding boxes.

Parking spot occupancy detection

Once the system has the positions of the parking spots, it can determine the state of the parking spots by measuring how much a parking spot's bounding box is overlapping with a vehicle's bounding box. To measure this overlapping, Intersection Over Union (IoU), also known as, Jaccard index can be used. IoU is measured by finding an area where two boxes overlap and dividing the overlapping by the area covered by both boxes, as illustrated in Figure 10.

Figure 10: Intersection over union

By calculating the IoU value for each detected vehicle's bounding box with the stored parking spot's bounding box, the system can determine whether the parking spot is occupied or not. When the IoU value is low, such as 0.10, the vehicle does not occupy much of the parking spot, thus, the state of the parking spot can be determined as free. However, if the IoU values are high, such as 0.80, the vehicle occupies the majority of the parking spot and in that case, the state of the parking spot can be determined as occupied. These threshold values are based on that the bounding box coordinates are not only covering the vehicle itself, but also a small part around the vehicle. Therefore, a vehicle's bounding box that covers a small part of the next by parking spot would be allowed in this case, however, it will not be allowed when the value is higher since it is considered to be occupying the other parking spot.

Another point that needs to be considered for parking spot detection is overlapping between the parking spots bounding boxes. Due to that the area of bounding boxes are typically larger than the vehicles themselves, there may be an overlap in parking spots' bounding boxes next to each other as shown in Figure 11.

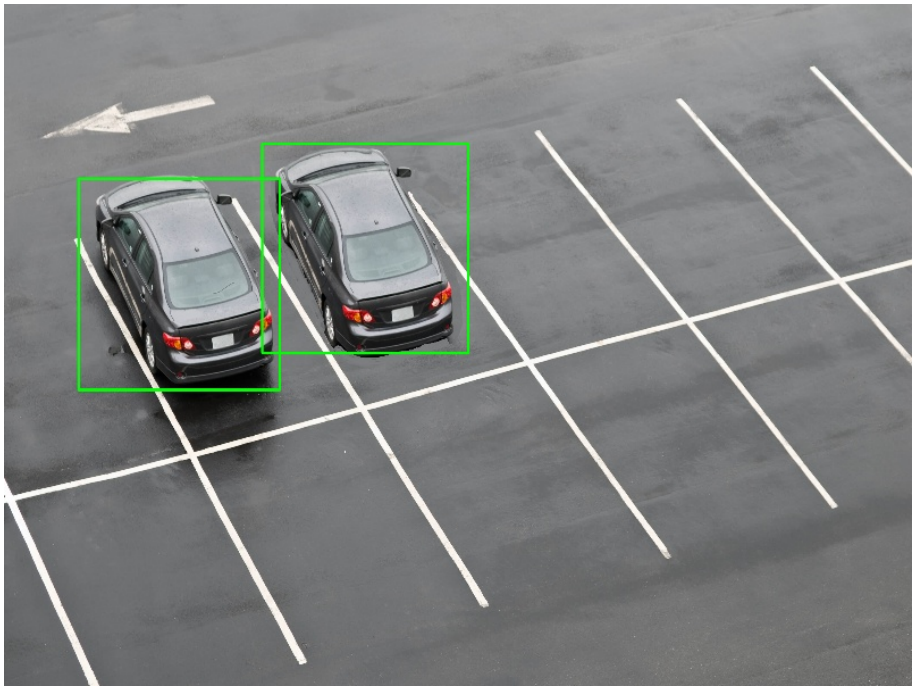

Figure 11: Overlapping in two parking spot's bounding boxes

As shown in the Figure 11, there is an overlap on the detected parking spots. The overlapping area may increase depending on the position of camera related to the parking spots. When one of the parking spots is occupied and the other one is unoccupied, the bounding box of the detected car may overlap with both bounding boxes of these two parking spots. In order to avert this issue, the IoU value of each parking spot with adjacent parking spots should be calculated. This value can be stored together with the pixel location of each detected parking spot and can be used to detect occupied parking spots. In Figure 11, we show that a parking spot should be unoccupied if the IoU value between the parking spot and the detected vehicle is less than 0.10. But, this threshold value must be increased if there is overlap between bounding boxes of parking spots. For instance, if the IoU for two parking spots is 0.30, the new threshold value that will be used for parking occupancy detection should be $(0.10+0.3 = 0.4)$ 0.4. Due to differences between the size of vehicles, the threshold value should be increased, e.g., 0.10 is used in this example. The choice of the threshold value depend highly on the angle of the camera. Capturing an image when the camera is pointed from the front of the vehicles will result in a small IoU value. However, capturing an image from the side of the vehicles will result in a high IoU value since each vehicle that is standing next to each other is covering a part of the other vehicle, thus, the threshold must be increased to not predict incorrectly.

Since the system is capable of detecting locations of vehicles and the occupancy status of the parking spots, all the parking spots will be detected by the time the system have detect all these parking spots and store their coordinates. To improve the accuracy of parking spot detection, the location of the parking spots can be recalculated based on the location of detected vehicles. Each time the system captures a new image, bounding boxes of the detected vehicles can be compared with the bounding boxes of the parking spots. If the bounding box of a new detected vehicle has a low IoU value between the defined bounding boxes of parking spots, e.g., less than 0,2, that means that the detected car is not parked on a defined parking spot. In this case, the bounding box of detected vehicle should be stored as a new parking spot. However, if the IoU value is

high, e.g., higher than 0.8, that means that the detected car is parked on a parking spot that has already been defined by the prototype. In that case, the pixel coordinates of the parking spot that the detected vehicle is parked on, can be updated by calculating the average pixel coordinates of the parking spot and the detected vehicle. The average pixel coordinate represents the average position of the parking spot, when a parking spot have been parked on several times.

An important aspect that should be considered while calculating the average value is that there are different types of vehicles. As mentioned before, the detected vehicle could either be a car, a truck, a bus, or a motorcycle. We assume that cars are more common than any other type of vehicles. For this reason, the calculation of the average position of the parking spots should only be performed when the detected vehicle is a car.

A possible issue that could be faced while running the system is a situation where a vehicle is performing parking at the same time as the system is performing vehicle detection. In order to deal with potential issue, the system can perform a second detection and compare both images in order to confirm that positions of detected vehicles are still the same. Performing an additional check assures that the system does not perform any vehicle detection on the image unless the vehicle is parked. If the location of the detected vehicles is not identical then a new image can be captured after waiting some time, e.g., 30 seconds.

Cluster analysis

The system can continue to work and define all the parking spots and calculate an average size for each parking spot based on the pixel coordinates of detected vehicles. However, it is critical that all the vehicles are parked properly. In a situation where a vehicle is parked incorrectly, e.g. double parked, the pixel coordinates of the vehicle may be defined as a parking spot. The system is not able to detect that which vehicles are parked correctly and which vehicles are parked illegally. In order to achieve this goal, a machine learning approach can be useful. Due to that it is not known which vehicle is parked correctly, an unsupervised learning algorithm will be a reasonable choice to improve the detection of parking spots by not taking into account those vehicles that are illegally parked. The purpose of using unsupervised learning is to distinguish between groups of vehicles that are correctly parked and that are parked in the same spot based on the positions and parking frequency of detected vehicles. In particular, we assume that the frequency of correctly parked vehicles will be higher compared to illegally parked vehicles. In accordance with the goal and the assumptions, an unsupervised machine learning branch cluster analysis is a suitable solution. By implementing a clustering algorithm, the data that has not been labelled, classified or categorized can be grouped.

In order to implement any clustering analysis, we need to have a certain amount of data. In this case, the data will be the pixel locations of the detected vehicles. For this reason, the system will store the pixel coordinates of the each detected vehicle's bounding box. However, this stored data should only consist of cars. Other types of vehicles, such as trucks and motorcycles can occupy an area that is much larger or smaller than an area of a parking spot, and thus can affect the result of the clustering analysis in a negative way.

Different popular clustering algorithms are used in various fields, such as image analysis, data compression, and pattern recognition. Various clustering algorithm requires different parameters that need to be known. One of the unknown parameters in our case is the number of parking

spots which equals to number of clusters that will be defined by the clustering algorithm. Some of the clustering algorithm requires this value as input and some of them don't. Although, this information can be configured in the system setup, in order to achieve a dynamic structure this information should be determined by the system itself. Another potential issue is changes on number of parking spots in the parking spot. Some parking spots may be removed or new parking spot may be added. In this case, the system will require to re-define the total number of parking spots. Due to this reason, we need to consider this parameter as an unknown. That means that the number of clusters should be detected by the clustering algorithm itself.

According to the reasons that are mentioned above, the clustering algorithm to be selected must be robust to outliers and the number of parking spots can be considered as an unknown. Density-based spatial clustering of applications with noise (DBSCAN) is a clustering algorithm that does not require any information about the number of clusters, instead it determines the number of cluster algorithmically. DBSCAN returns the number of clusters as a result. Additionally, DBSCAN is robust to outliers, since data that cannot be connected to any cluster will be considered as noise [48] [49], as shown in Figure 12.

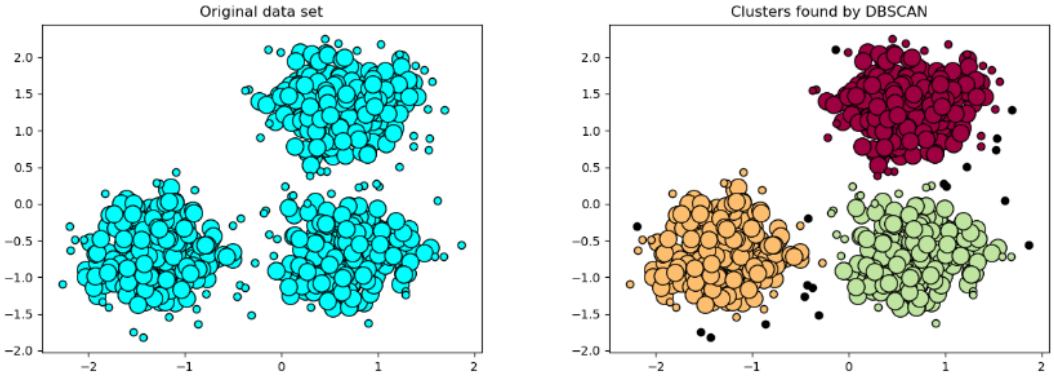

Figure 12: Density-based spatial clustering of applications with noise (DBSCAN). Three clusters are detected by the algorithm and outliers are marked with black colour.

There are two parameters that need to be known when running the algorithm, *min_samples* and *eps*. The *eps* specifies the maximum distance between two samples for one to be considered as in the neighbourhood of the other, e.g., if the distance between two samples are smaller than the *eps* value then the sample can be included in the same cluster as the other samples. The *min_samples* specify the number of samples in a neighbourhood for a point to be considered as one cluster, e.g., if the *min_samples* parameter is set to 7 then in order to define a cluster 7 samples must exist within the distance of the defined *eps* value.

The *eps* can be calculated by the prototype itself. One way to determine this value is by calculating the minimum distance between bounding boxes of the detected cars in each captured image. The size of the bounding box of the parking spot that is farthest from the camera will be smaller compare to the other parking spots that are closer to the camera. This means, that the distance between the two parking spots that are farthest from the camera will have a shorter distance compare to the other parking spots. The *eps* should not be more than half of the calculated minimum distance. If this value is too small, many points can be detected as outliers. On the

other hand, if the *eps* is higher than 50% of the distance between two parking spots, two parking spots may be defined as one cluster, in other words as one parking spot. The estimated value of the *eps* is 40% of the calculated minimum distance. The validity of this estimate will appear in the test results. The value of *eps* is illustrated in Figure 13.

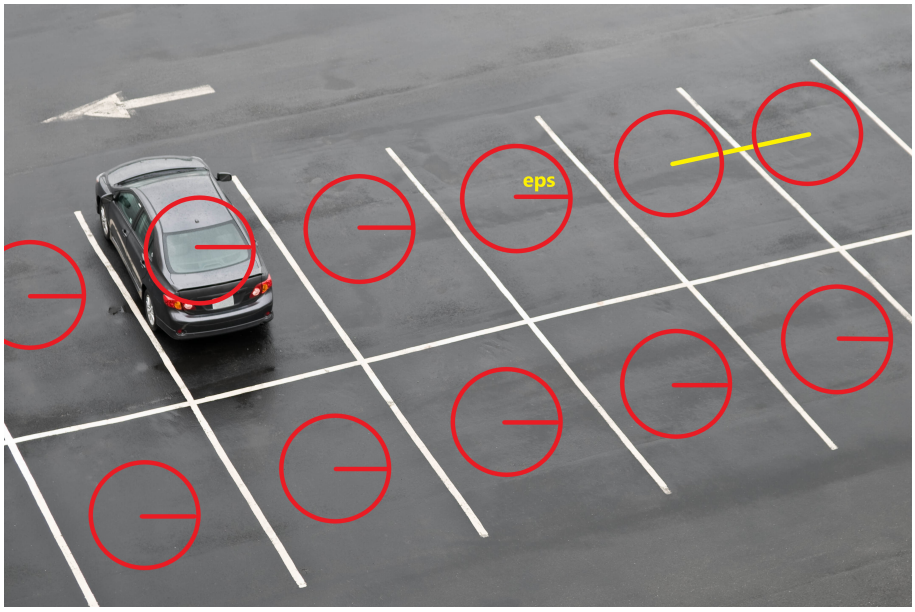

Figure 13: Illustration of the maximum distance between two points for one cluster (*eps*) on a parking lot. The yellow line is the minimum distance between two parking spot in the image. The radius of circles are equal to 40% of the length of the yellow line.

The *min_samples* parameter can be determined in the simplest way by the frequency of use of the parking lot and by how much data has been collected. Each time the system detects a car, the pixel coordinates of the vehicles can be stored, and the clustering algorithm can be run when the total recorded data reaches a certain limit. As an example, in a parking lot that contains 10 parking spots, if the system collects a total of 200 coordinates of vehicles, that would result in average 20 vehicles per parking lot. In this case, the *min_samples* should not be higher than 20. To keep the *min_samples* approximately 40% of the average number of samples on a parking spot should be appropriate. The reason behind this choice is that some of the parking spots may be more popular than others, such as parking spots that are close to an exit. To keep *min_samples* too low, can generate a risk of detecting the location of illegally parked vehicles as a parking spot. If the *min_samples* is too high, some parking spots may be detected as outliers.

After a certain amount of data is collected, the system can perform a cluster task. To simplify this process, instead of using all four coordinates of a bounding box, only the centre point can be calculated and used. Based on the result of DBSCAN, all the groups and outliers can be detected. Afterwards, by calculating the average coordinate of each group, all the parking spots can be determined.

After performing the clustering task, the coordinates of the determined parking spots do not need to be re-calculated with each new detected vehicle. Instead, the re-calculation of the average parking spot coordinates will be performed when a certain number of samples have been collected and the clustering task is performed. Thus, the dynamic structure of the system will be

strengthened.
In summary, to define or re-define a new parking spot with each detected vehicle should be performed until a clustering task is completed. After performing the clustering task, the pixel coordinates of the new detected vehicles should be used for determining the state of the parking spots and the coordinates should be stored for the next clustering task. The result of the clustering task should be considered as correct until a new clustering task is performed.

Illegal Parking Detection

Based on the clustering result, the location of all the valid parking spots will be determined. In order to detect illegally parked vehicles, we decided to use the same method as we used to detect the occupancy status of the parking spots. If the bounding box of a detected vehicle overlap with more than one parking spot and if the overlapping value is more than the overlapping between these parking spots, then this vehicle can be considered as illegal parked (double parked) vehicle as shoen in Figur 14.

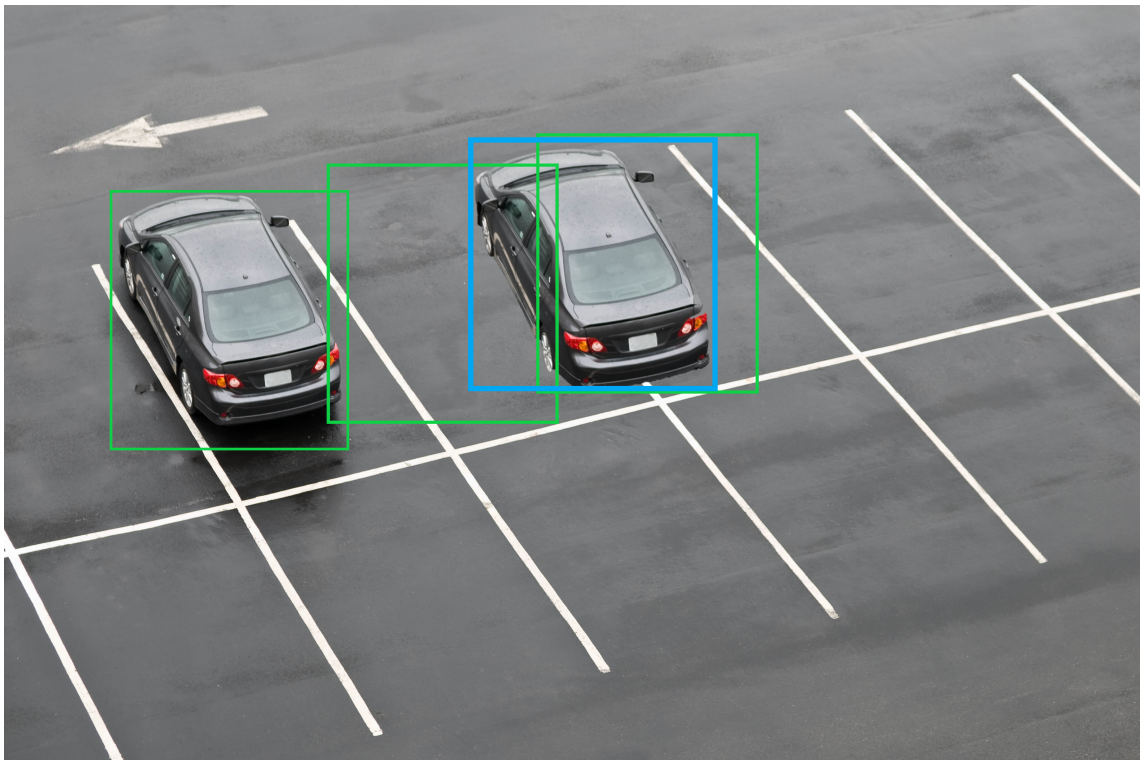

Figure 14: Double parked vehicle. Green rectangles represents the bounding boxes of the parking spots. Blue rectangle represents bounding box of the detected double parked vehicle.

Both parking spots that are occupied by one double-parked vehicle should be defined as occupied because it would be difficult to park another vehicle in any of these parking spots. As illegally parked vehicles are detected based on the area they occupy, doing this determination for large vehicles such as trucks may not give a correct result. Therefore, vehicles such as trucks and buses will not be determined as illegally parked vehicles.

5.3.2 Data transmission

There are several techniques for transferring the data to a server, such as short range and long-range transfer of data. The short range data transfer requires most often a close gateway that performs the data transmission for the device, e.g., Bluetooth or WiFi module on the device that transfers the data to a gateway nearby. On the other hand, long-range techniques, e.g., LoRa, Sigfox or NB-IoT, can be used directly with a device in order to transmit the data directly to a gateway that is placed further away.

The data transmission technique that we decided to use in our system prototype is LoRa. LoRa was chosen due to the need for running the system locally without any need of having a network connection. In addition, it was chosen based on the availability of gateways nearby and the platform, the chosen platform, i.e., *The Things Network* (TTN), which is available free of charge to store and visualize data. Furthermore, the LoRa module can be powered on a single board computer through the USB connection, which facilitates the communication between the computer and the LoRa module by utilizing serial communication.

5.3.3 Hardware choice and configuration

To perform the needed experiments for our study, there is a need for hardware. There are three different hardware parts required as mentioned in Section 3.2.

- **Camera:** The camera that we used in our study is an AXIS P1435-LE camera, which was provided by Atea. We chose to use an Axis camera since, they are easy to configure and use in a local solution. Moreover, AXIS cameras provides high resolution images, which is an important aspect to consider when performin image processing. In addition, the communication through Ethernet cable from the camera to the single board computer contributes with a more reliable and secure image data transfer when performing the image processing.
- **Single board computer:** The computer that we used for performing the calculations and image processing is an Nvidia Jetson Nano. Our choice of computer was based on its small size and the processing power, which is higher than other similar single board computers, such as Raspberry Pi or Google edge tpu, has a higher advantage considering the processing power [50]. In addition, since the computer has a GPU integrated onboard, this would be an advantage when performing image processing, since it increases the performance for detecting the objects slightly faster.
- **LoRa module:** Since the need for transmitting the data is an insignificant part of the system, the choice of LoRa module was freely chosen. The LoRa module used for this purpose was the Badgerboard development board, which is a small LoRa module powered through a USB cable. The module is easy to program and configure with TTN which is an advantage for this purpose. In addition, libraries for programming the board are available free of charge for the community to install and use.

5.3.4 Integration

Software design

To give an overview of the different software parts required of the system, a diagram describing the flow was created as illustrated in Figure 15. We constructed the diagram based on the system requirements that we presented in Section 5.1. In addition, the creation of the diagram was performed iteratively by going back and forth in order to validate that all of the functionalities are included and fulfills their purpose.

The software part of the system can be seen as 6 different tasks: detect vehicles, find parking spots, calculate intersection for determining if a parking spot is vacant or occupied, calculating the vehicle intersection with parking spots for determining if any vehicle is parked unlawfully, transmitting the data, and clustering the data. These tasks can be divided into two different phases: the learning phase that configures and define the parking spots by collecting data and then clustering the data, and the operation phase that based on the result from the learning can use this template to decide whether a parking spot is occupied or a vehicle is parked illegally. The system alternates between these two phases, since it is required that the system prototype is a dynamic solution and can continuously learn if any changes of parking spots would occur on the parking lot.

Before designing the system prototype we needed to decide how often the object detection process should be performed. Further, a decision on when the data should be transmitted was needed to be analyzed. Since the vehicles are mostly parked during longer time periods, the need for checking each minute can be considered over exaggerated. However, since the system serves other purposes, such as detecting violated parking and vacant parking, it would be inappropriate to wait between each detection for more than 10 minutes. Therefore, a time interval of 5-10 minutes between each object detection process might be a reasonable choice for this purpose. Secondly, when it comes to the transmission of the data about the parking lot status, the need for transmitting each time an object detection process has been performed might be unnecessary if it is the same data. Thus, transmitting each time a new car has parked or left the parking lot, might be a better alternative. Importantly, from a smart parking perspective, this is a crucial feature to keep the users up-to-date about the available parking spots. We illustrate in the diagram in Figure 15 the flow of the system.

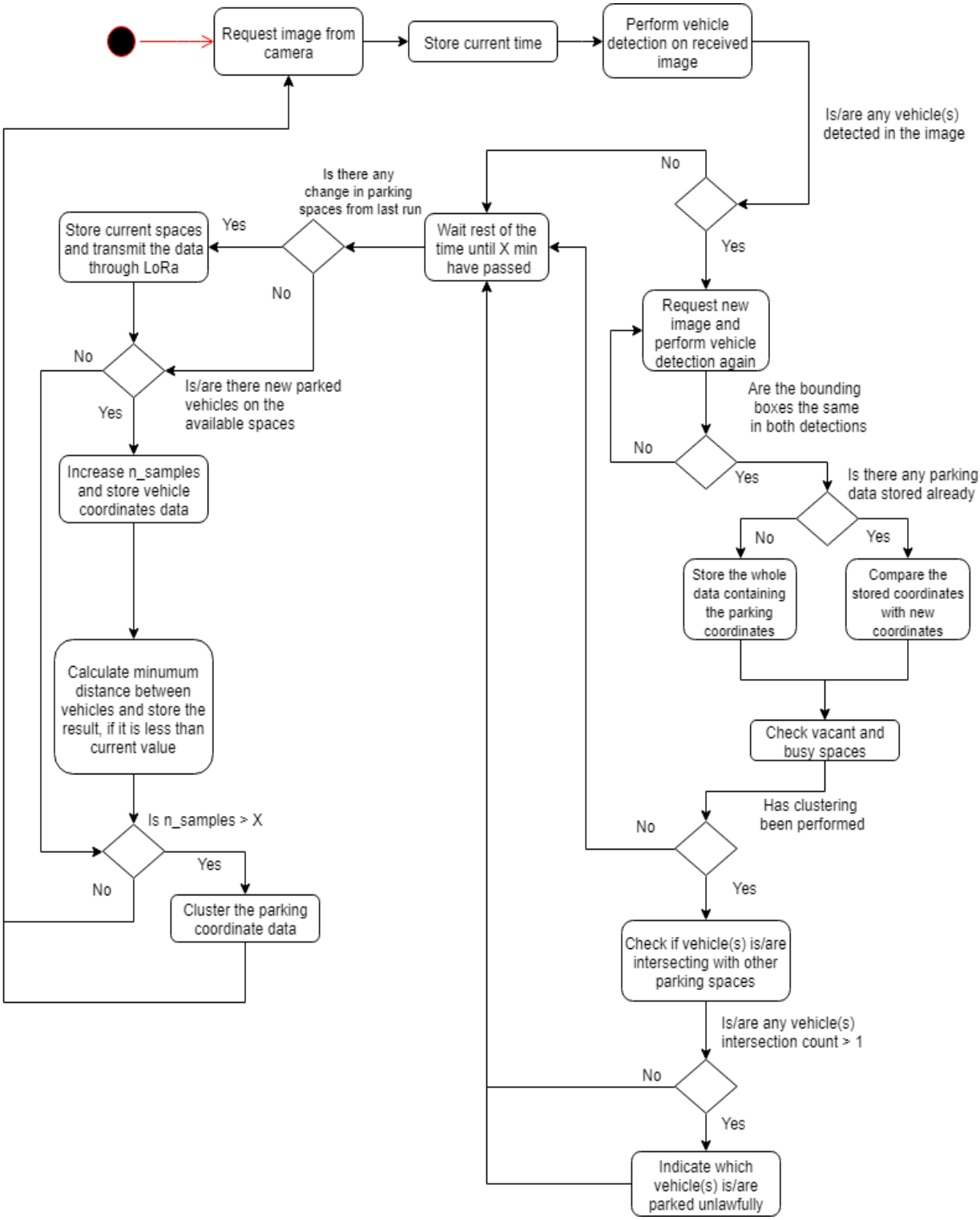

Figure 15: Activity diagram

5.4 Build the system

Implementing the whole system was performed iteratively. The first step was to set up and configure the devices before implementing the code. When the hardware was configured and set up, the implementation of the code could be performed.

To receive the image from the Axis camera it was needed to configure the camera with a static IP. The configuration of camera IP would enable to use the same IP each time connecting the camera to the single board computer. Next, the configuration of the view area was necessary in order to capture only the parking lot area. The use of view area would help to receive an image on only the parking area that is of interest to monitor.

To set up the Nvidia Jetson Nano single board computer, we first had to install an operating system, where we decided to use Linux. Additionally, to run the deep learning algorithms, we installed some required libraries and programs, such as OpenCV and python 3.6 [51] [52] [53]. Python and OpenCV are used since they support the use of clustering and also the use of deep learning models. When the installation of these required libraries and programs was done, then the implementation of the code was developed and tested.

Lastly, the configuration and implementation of the Badgerboard LoRaWAN module were done using the Arduino IDE. To connect the LoRaWAN module to TTN, it was required to create an application and registering the device on the platform. A key will be given to configure the device afterwards and connect to the platform. The size of the transmitted payload is 3 bytes. We created a payload decoder that converts the bytes to readable information on the TTN console. The payload decoder is created using JavaScript.

5.4.1 System overview

In Figure 16, we illustrate the whole system with the different hardware and their connections.

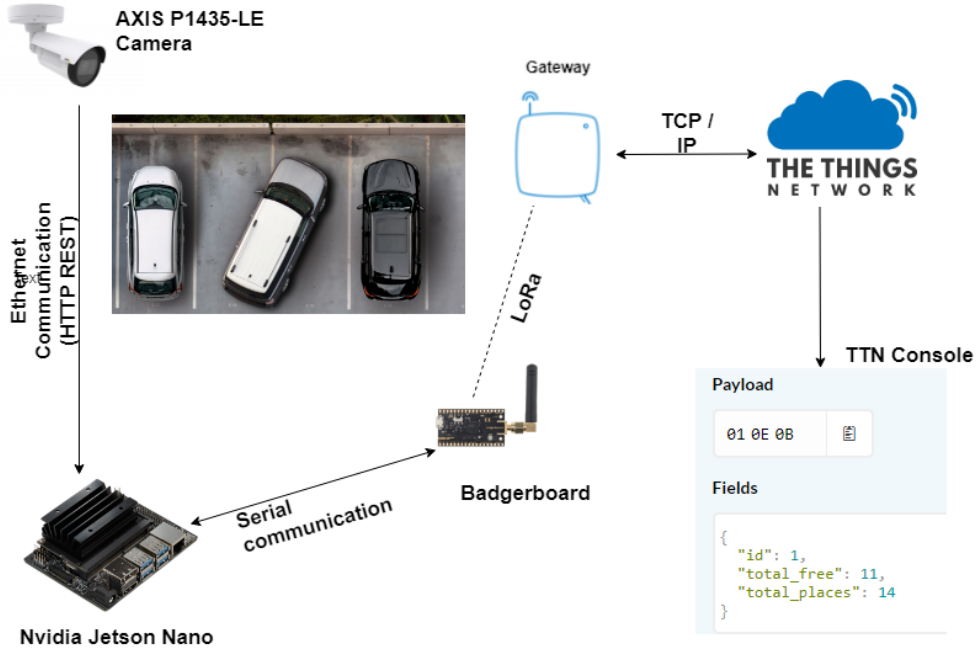

Figure 16: System overview

5.5 Observe and evaluate the system

This section aims to observe and evaluate the system to validate that the system fulfills the requirements presented in Section 5.1. To this end, the test cases in Appendix A were created with the purpose of testing each small part developed, but also the whole system at the end when everything has been integrated. Observing and evaluating the system enhance the development process of the system, where each part can iteratively be further improved and redesigned if any changes would be required.

In order to evaluate the prototype, it was decided to perform the test on a parking lot with 14 parking spots. The tests were performed by using our own vehicles. During the evaluation process we parked the vehicles in every parking spot several times. In addition, illegal parking was also performed several times in different positions. The evaluation of the prototype can be divided into 6 main parts: vehicle detection, parking spot occupancy detection, illegal parking, cluster analysis, and data transmission. Figure 17 shows the parking lot, where we performed our evaluation.

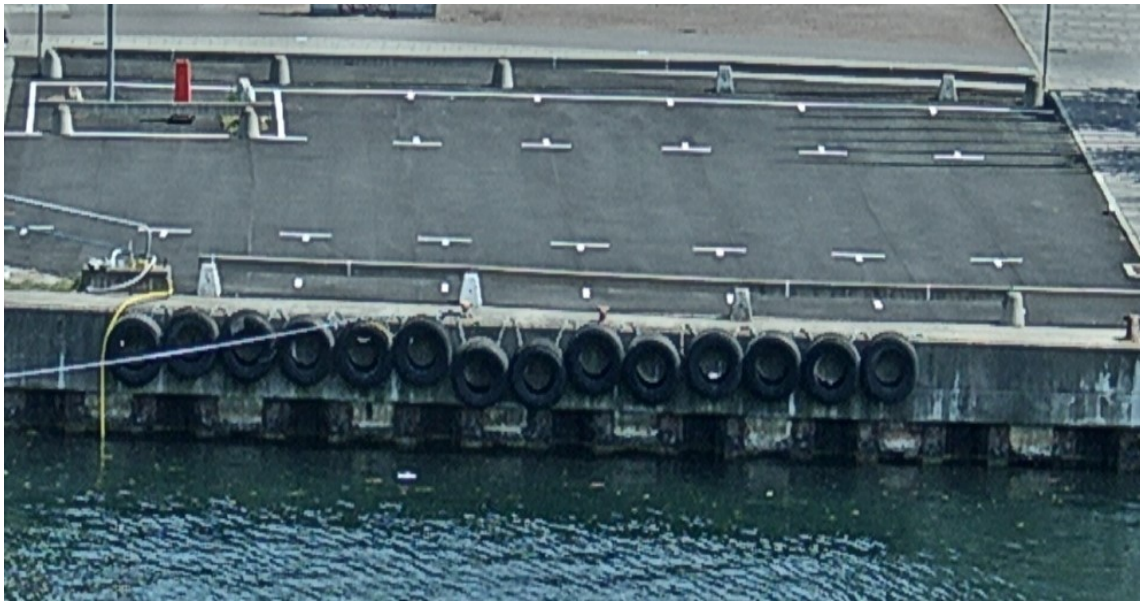

Figure 17: The parking lot that is used for performing all system evaluation

As seen in Figure 18, two vehicles that are parked in the parking lot were able to be detected by the system. The system defines two parking spots based on the location of the detected vehicles and stores their respective locations in two different files. The first file consist of the temporary location of the parking spot and the second file consists of the coordinates of each of the detected vehicles that will be used as a dataset for the clustering analysis.

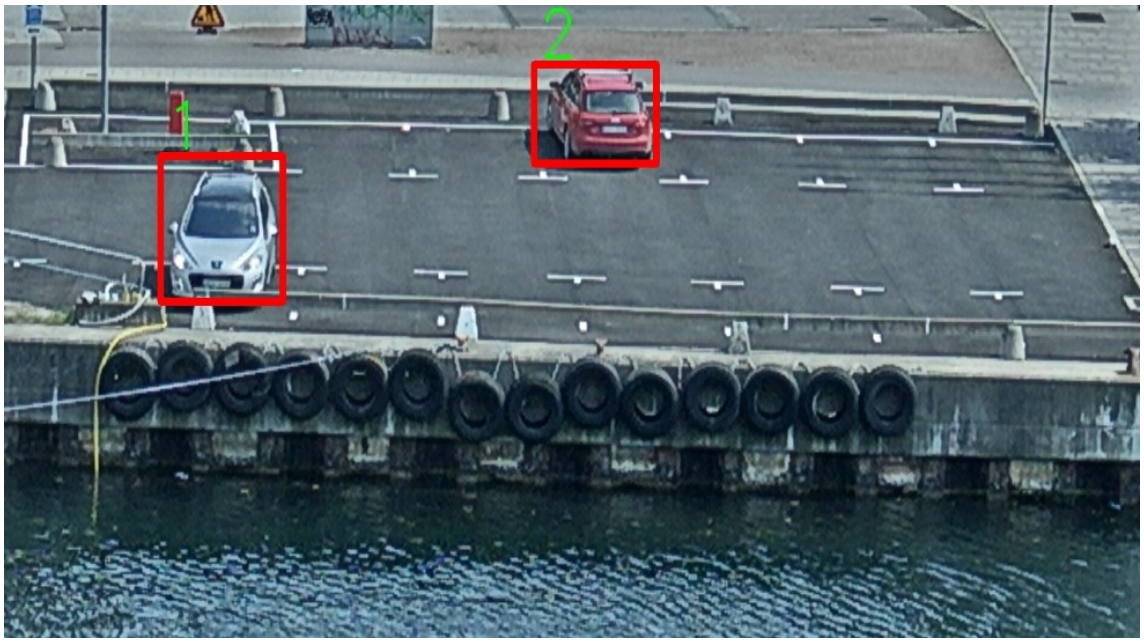

Figure 18: Two detected vehicles in the parking lot

When all of the possible parking spots had been parked on, the system was able to define all the 14 available parking spots as shown in Figure 19. As seen in the figure, the size of the bounding boxes is not the same, although the detected vehicle was the same for all bounding boxes, except

for bounding box number 7. The sizes of the bounding boxes are heavily depending on the area that the vehicle occupies in the image. If the captured image contains a slightly side view of the vehicle, the occupied area will be higher than in another image, which contains only the front or back view of the same vehicle.

In addition, the system was able to detect the occupied and unoccupied parking spots. As seen in the figure, the system marks the occupied spots with a red bounding box, while the unoccupied spots are marked with a green bounding box.

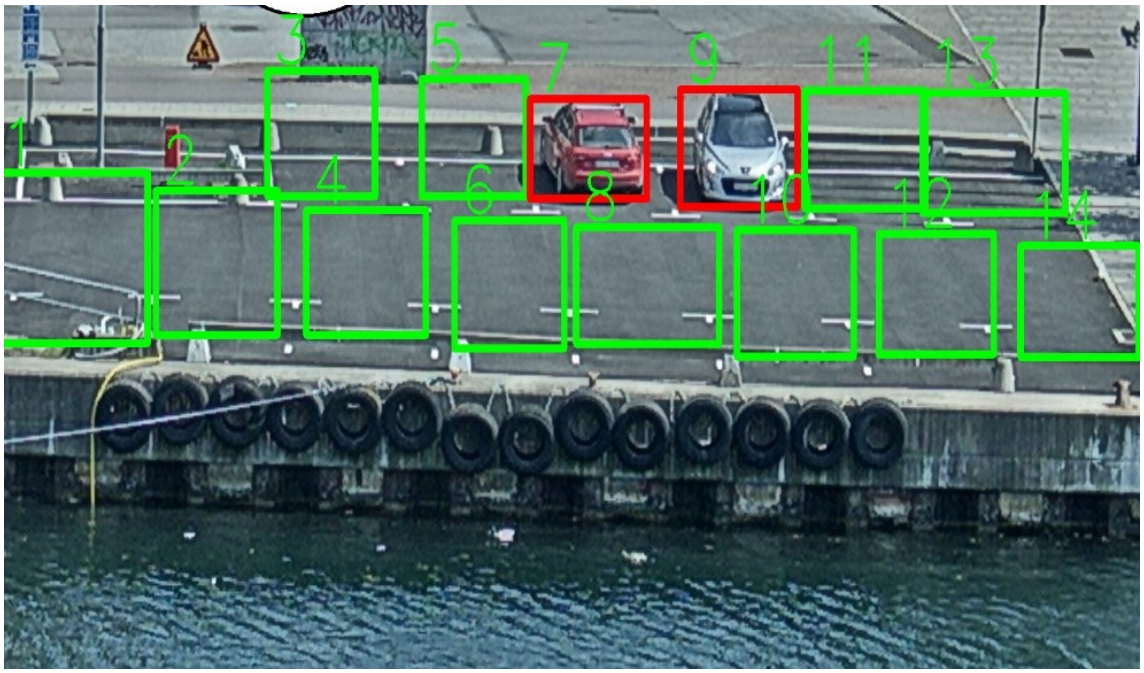

Figure 19: All detected parking spots before clustering analysis

Since the system has not performed a clustering analysis on the collected parking spot data, due to that the system have not collected enough data, the illegally parked vehicle is not defined as illegal parking. However, the system was able to mark both parking spots as occupied as illustrated in Figure 20.

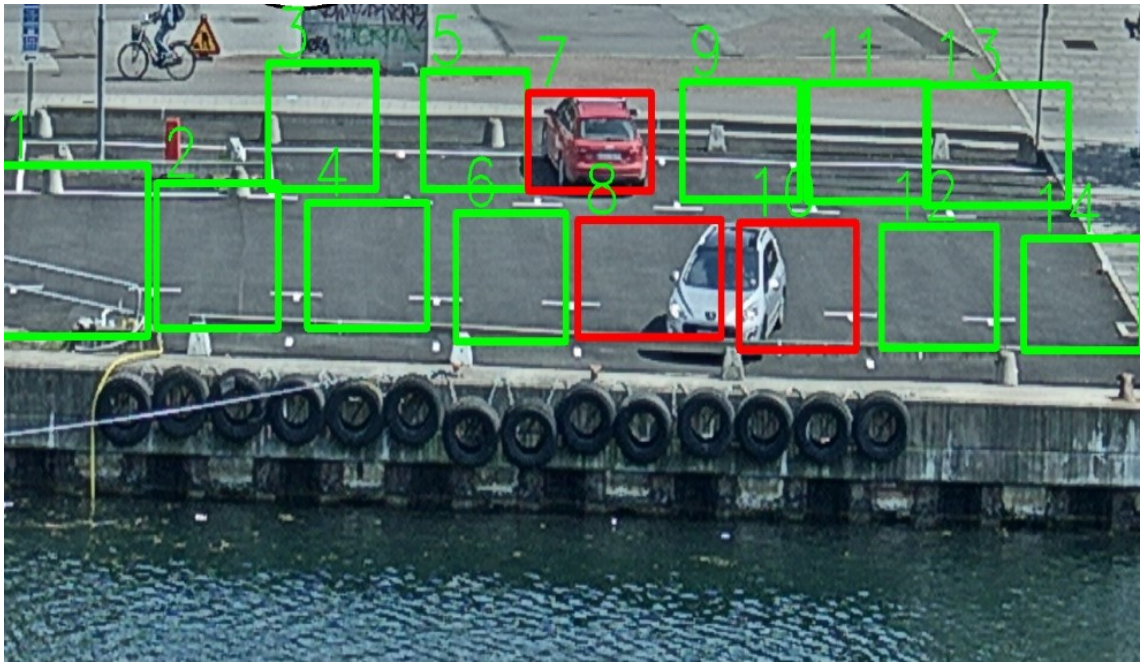

Figure 20: A illegal parked vehicle before clustering analysis

As described before, when the system collected a certain amount of data it performed a clustering analysis. Due to the limited time, the vehicle was parked twice on each parking spot, except parking spot 7, which was out of control. The dataset that is used for clustering contains data from correctly and illegally parked vehicles. We captured 29 images during the evaluation process, which 26 of the images were captured when the vehicle was parked correctly and 3 of the images were captured when the vehicle was parked illegally. As seen in Figure 21, different parking spots are defined as clusters while the three black dots represent the illegal parking data and are considered as outliers. The system defined 14 clusters, which represent the 14 parking spots. Each parking spots in Figure 20 can be illustrated in Figure 21 by rotating and inverting the figure 180°.

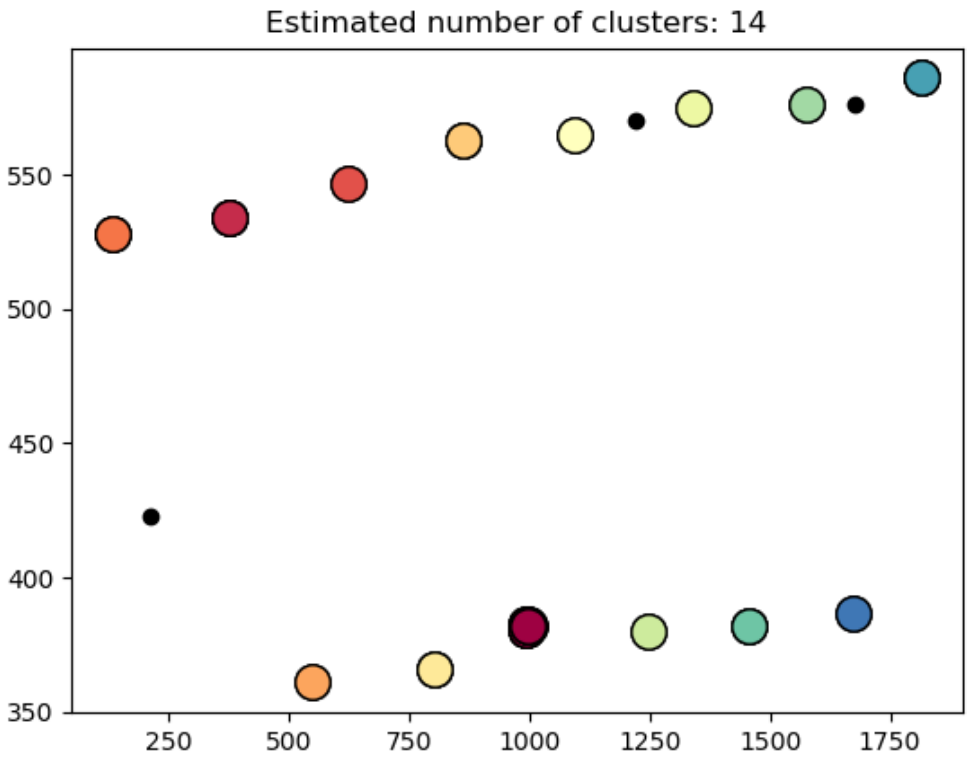

Figure 21: Clustering result

When the clustering has been performed, the parking spots are defined by calculating the average value of all samples that belongs to the same group. In addition, the illegally parked vehicles that were detected as outliers were removed from the dataset. Figure 22 shows the final location of the parking spots based on the clustering result of the colored dots that is illustrated in Figure 21.

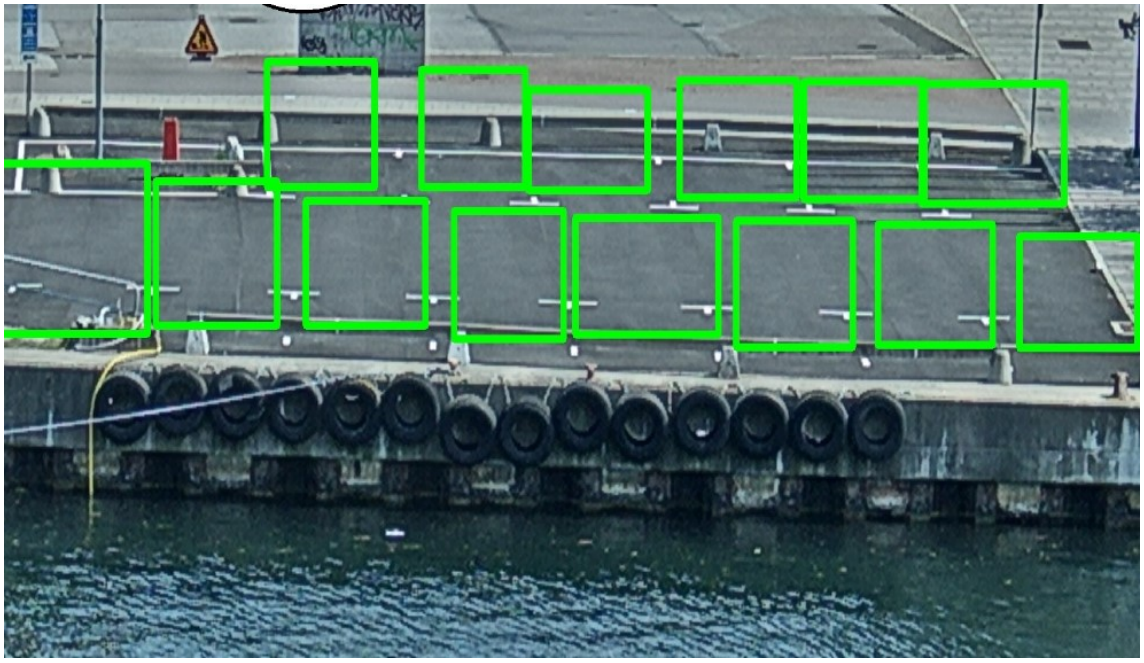

Figure 22: Position of parking spots after cluster analysis

After the cluster analysis, the system continues to detect vacant and occupied parking spots but does not perform any re-calculation with each new detected vehicle. Figure 23 shows two occupied parking spots after the cluster analysis was performed.

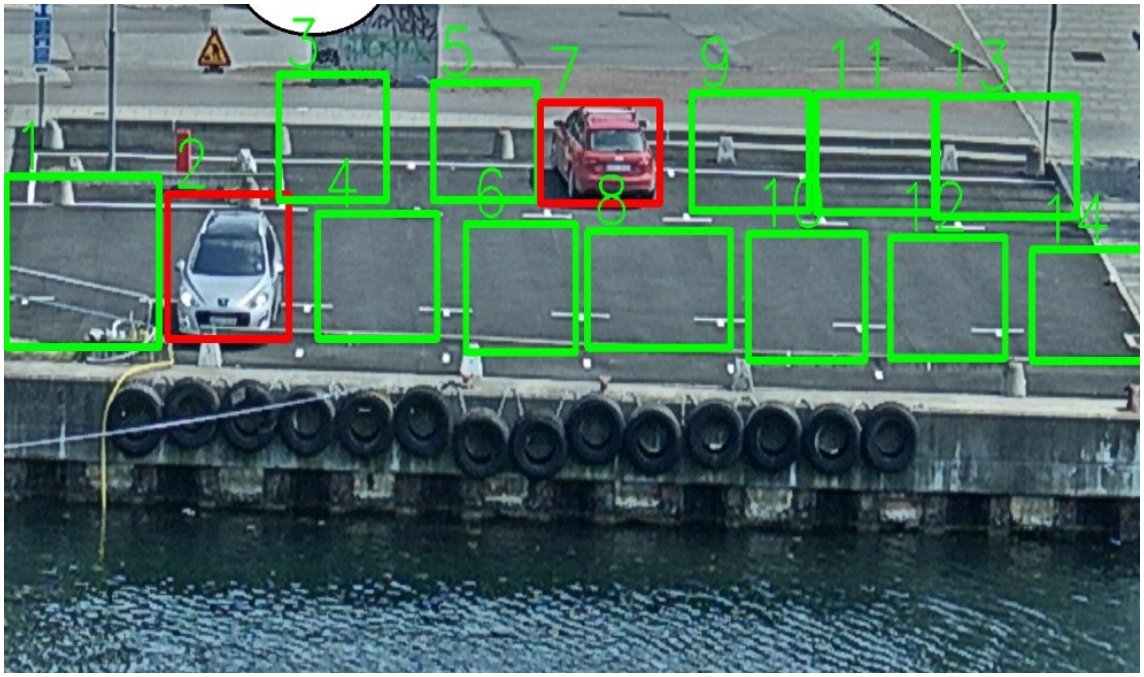

Figure 23: Two detected vehicles after clustering

When the parking spots have been defined and the cluster analysis has been performed, the system was able to mark vehicles which were parked illegally with a blue bounding box. Figure

24 illustrates a situation where a vehicle is double parked. In addition, since both parking spots were occupied by the vehicle the system marked these two as occupied.

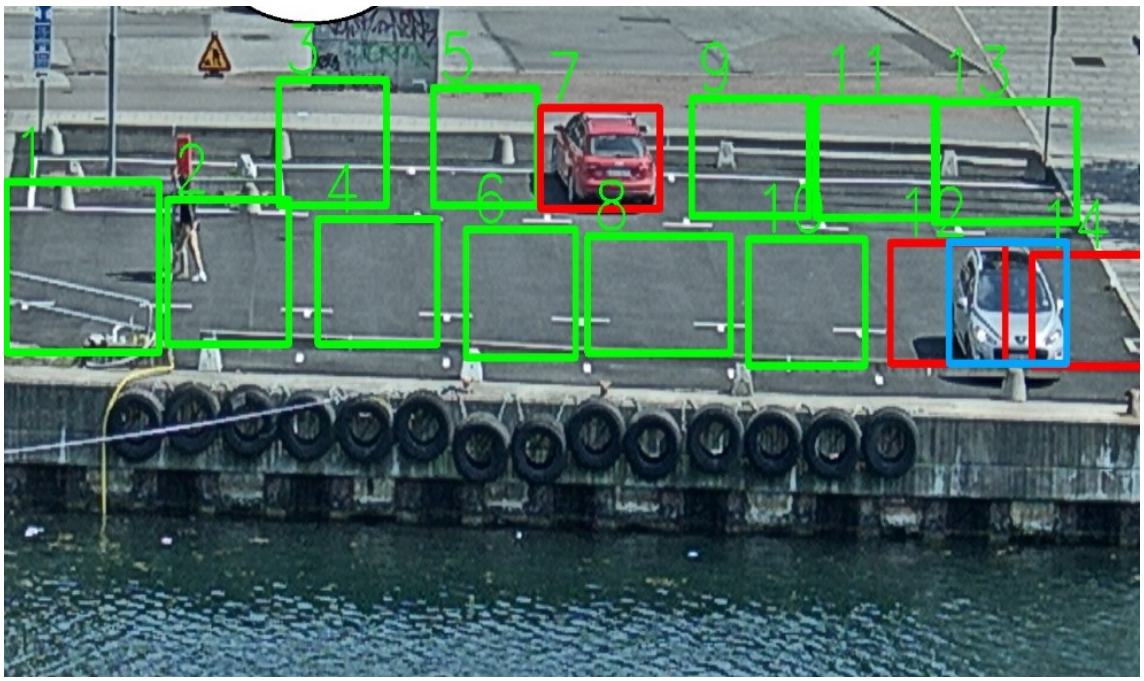

Figure 24: Double parked vehicle after clustering

As shown in figure above, the number of the defined parking spot is 14 and there are 11 vacant parking spots. The system then sends the information about available parking spots to the TTN, which decodes the received data and converts the data to readable information, which can be seen in Figure 25.

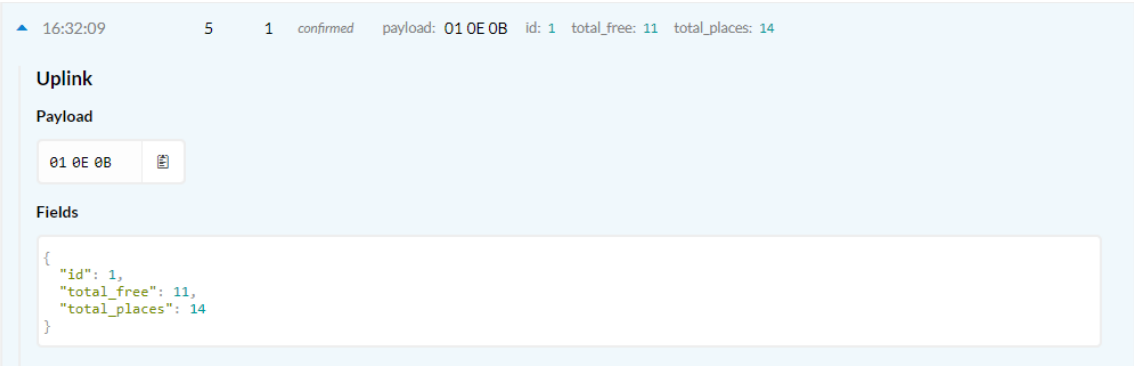

Figure 25: TTN console, number of parking spots is 14 and number of unoccupied spots is 11.

6 Discussion

In this section, a discussion is made on how the method was used and what results were achieved. Moreover, a discussion is made regarding how our study differs from previous research.

6.1 Iteration ability in the method

The chosen approach for this work was design and creation and the used method was a system development method, which provides an iteration ability process. The reason for this method choice was based on the iteration ability in this method. The iteration ability in this method facilitated the process of developing the prototype by being able to go back and forth between the different steps. This was helpful when analyzing and designing the system, where there was a need to observe and evaluate each small part of the system. In addition, being able to test different deep learning and machine learning algorithms iteratively enabled us to choose the right algorithm. Lastly, test cases that failed due to bugs in the implementation code, could easily be corrected by going back and correct these bugs and then run tests again.

6.2 Result Discussion

The aim of this work was to build a prototype of a dynamic smart parking system in order to detect whether there is any vacant parking spot on a parking lot and if there is any vehicle that is parked in an unauthorized manner. The achieved results of our work indicate that it is able to detect vehicles and define parking spots where vehicles are being parked. Further, the detection of an illegally parked vehicle was possible when the vehicle was double parked. The use of the YOLO architecture for detecting the vehicles provided sufficient performance and was able to detect the vehicles without any problem. Moreover, the clustering analysis achieved a positive result, which was able to detect illegal parked vehicles as outliers. The outliers that represent the illegal parked vehicles were detected and eliminated in order to not store this data as a valid parking spot. However, data of valid parking spots was clustered into one cluster without any problem.

An important observation made during the evaluation was that the size of the different bounding boxes of defined parking spots were not the same although the detected vehicle was the same. This was due to the limited amount of data and few cars that was used during the evaluation. By having a greater amount of data on each parking spot with different cars, this would most likely contribute with a better average value of the bounding box. Running the system during longer periods and collecting data of several cars would give a better result. Another point to add is that the angle of the camera plays a major role in how big the bounding box will be. Taking a picture in front of a vehicle would give a smaller bounding box than taking a picture from 45°.

Another aspect to consider is the calculation of the intersection between the bounding boxes. When checking if a vehicle is occupying a parking spot we add 0.10 to the threshold in addition to what the parking spot occupy with the neighbouring parking spot. This value was added due to the different sizes of vehicles which might give a false indication that a vehicle is parked illegally when it is not. In our case, this small value was chosen due to that the camera was pointing straight towards the vehicles, which overlapping between vehicles. However, depending on which direction the camera is installed, this value will have to be changed in order to adapt to the parking lot. Installing the camera from the side of the parking lot where each vehicle covers the vehicle standing next by, requires that this value is increased in order to not predict falsely that the vehicles are parked illegally.

6.3 Relation to previous studies

The smart parking field has been existing for a few years, where there are offered both sensor-based solutions and also vision-based solutions. However, the focus has in the past year been moved towards the vision-based solutions which mainly is due to the adaptation and improvements of image processing in deep learning.

The common divider between the previous work done in [42], [23], [25], and [43] is that all of them depend on a static nature/structure. By having a static solution constrains the possibilities of relocation while also requiring a greater time to install and configure the system. The configuration time taken for configuring the system depends highly on the size of the parking lot. Further, additional time is required if there are changes on the parking lot that require that reconfiguration of the parking spots are necessary. As a result, our thesis focused on developing a dynamic solution that is self-taught with the purpose of eliminating the need for configuring the system where parking spots are available.

Furthermore, no previous research as can be found have used a vision-based system to detect violated parking on a parking lot. Most of the research papers available on smart parking have been focusing on studying how a vision-based system can be built and what accuracy can be achieved. In our work, we have built a system prototype which can detect violated parking, such as if a vehicle is double parked.

7 Conclusions & future work

7.1 Research questions

In Section 1.3, the following two research questions were presented with the aim to be answered during our thesis work:

> **RQ1** What is an appropriate solution to dynamically detect vacant parking spots by using a vision-based solution?
>
> **RQ2** How can a vision-based smart parking system detect illegal parked vehicles?

The first research question is completely answered in our study, since this thesis shows an example of how to build a system prototype that can detect vacant parking spots by using a vision-based approach, where the presented system prototype has dynamic characteristic. To achieve this goal, system requirements were extracted based on the purpose of this study and research questions. The system requirements were presented in Section 5.1. In order to answer the first research question, all the system prototype requirements needed to be fulfilled. Below, we explain how the system prototype requirements were met, based on the result of this study.

Req1 The processing of images should be done locally without sending or storing any image data.

As explained in Section 5.3.3, all image processing tasks are handled on a single board computer (Nvidia Jetson Nano). The single board computer and the camera is connected via an Ethernet cable. The prototype does not store any captured image and information about the parking lot is transmitted to the TTN by using LoRaWAN technology. The transmitted data does not contain any image data from the captured image.

Req2 The prototype should be able to provide information about the number of vacant parking spots on a parking lot.

Evaluation results from parking occupancy detection shows that the prototype is capable of detecting vacant and occupied parking spots. The information about the number of unoccupied spots is sent to the TTN together with the system id and the total number of detected parking spots.

Req3 The prototype should be able to automatically define the parking spots without predefining their locations.

The results presented in Section 5.5 show that the prototype is capable of defining the parking spots each time a vehicle was parked automatically. In addition, after performing the clustering the system could detect the valid parking spots.

Req4 The prototype should be able to transmit data without having Internet access.

Test results from data transmission shows that the prototype is capable of transmitting data by using LoRaWAN technology without having Internet access.

Req5 The prototype should be able to detect illegally parked vehicles.

As explained in Section 5.3, the prototype calculates the intersection over union value for each vehicle and the detected parking spots. According to the test results from the illegal parking detection, the prototype is capable of detecting double parked vehicles and vehicles that are parked in a area that is not defined as a parking spot.

Req6 The prototype should be able to adapt to environmental changes in the parking lot, e.g., changes in the number of parking spots, and the position of parking spots.

As explained in Section 5.3, the prototype is designed to continuously store coordinates of the detected vehicles that are parked, and the parking spot locations are calculated based on the stored coordinates. The prototype performs a clustering task to calculate the new position of the parking spots each time the total number of samples reach a certain predefined limit. This is a continuous process. Performing this clustering task continuously will help the prototype to adapt to changes in the parking lot.

The second research question was answered in Section 5.3, by fulfilling Req5 of the system prototype requirements. By implementing the function for calculating the IoU of the vehicle with the parking spots could be determined whether the vehicle was parked illegally or not. The prototype is able to detect double parked vehicles and vehicles that are parked on an area that is not defined as a parking spot. Illegal parking is, however, not limited to these two situations that the system can detect. Many other situations are defined as illegal parking, such as parking on a spot without a parking ticket or parking in a handicapped spot without any permit but these situations are out of scope for this study.

7.2 Research contribution

The contribution of this work has been mainly a proof of concept on how a vision-based dynamic smart parking solution can be developed, with the purpose of contributing with a sustainable parking solution. Moreover, this thesis contributes with a prototype of a system that can be used for detecting violated parking.

7.3 Future work

The developed prototype in this thesis is still at an early stage. The research field within the vision-based smart parking solution has much more to contribute and improve within this field. The research performed in this thesis has been mainly focusing on the detection of violated parked vehicles and building a solution which dynamically can be self-taught by the time. Therefore, the need for improving the system should be considered. Additionally, the vast majority of available deep learning algorithms might be of interest to be evaluated in order to find a better approach of detecting vehicles than that is used today. Furthermore, the deep learning architectures depend heavily on how the architecture is being trained and what images are being used for training the models; hence, a possible future work would be to train a new model. In addition, since the efficiency of the model depends on the amount of the trained objects, it might be good to train only one type of object, which in this case would be vehicles.

Due to the small time frame for performing this thesis, evaluating the system was only performed during a limited time, thus there is a need for evaluating the system further during a longer time period and in different parking lots. The detections of vehicles in different weather conditions and light conditions, e.g., when it is snowing or raining and when it is dark or light outside, might affect the detection. Thus, evaluating the system during different weather conditions and light conditions is required in order to evaluate the consistency of the system.

The visualization part of this thesis has been an insignificant part, where only data have been transmitted but not been considered to be handled in any way at the platform. Therefore, the transmitted data should be interpreted and visualized by either creating a user-friendly application or integrating the platform with a suitable tool for visualizing the information.

References

[1] D. Sperling and D. Gordon. Two billion cars transforming a culture. *TR News*, pages 3–9, 11 2008. ISSN 0738-6826.

[2] T. Svensson and R. Hedström. Parking : policies, measures and consequences for urban transport. *VTI notat*, 2010.

[3] D. C. Shoup. Cruising for parking. *Transport Policy*, 2006. doi: 10.1016/j.tranpol.2006.05.005.

[4] K. Cullinane and J. Polak. Illegal parking and the enforcement of parking regulations: causes, effects and interactions. *Transport Reviews - TRANSP REV*, 12:49–75, 01 1992. doi: 10.1080/01441649208716803.

[5] E. Gantelet and A. Lefauconnier. The time looking for a parking space: Strategies, associated nuisances and stakes of parking management in france. *Association for European Transport and contributors 2006*, 2006.

[6] H. Rivano T. Lin and F. Le Mouël. A survey of smart parking solutions. *IEEE Transactions on Intelligent Transportation Systems*, 2017. doi: 10.1109/TITS.2017.2685143.

[7] SmartParking. Overhead guidance indicators, 2019. URL `https://www.smartparking.com/technologies/overhead-guidance-indicators`. Online; accessed 22 May 2019.

[8] Trafikia. Parkeringssystem, 2017. URL `http://trafikia.se/parkeringssystem/`. Online; accessed 3 march 2019.

[9] Trafikia. Parkeringssystem, 2018. URL `http://trafikia.se/parkeringssystem-2/`. Online; accessed 3 march 2019.

[10] Parksol. Intelligent parking guidence system, 2019. URL `http://www.parksol.lt`. Online; accessed 22 march 2019.

[11] M. Cesana A.E. Redondi M. Tagliasacchi L. Baroffio, L. Bondi. A visual sensor network for parking lot occupancy detection in smart cities. *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, 2015. doi: 10.1109/WF-IoT.2015.7389147.

[12] H. Vandelinde S. Milligan, M. Cavanagh. *Understanding Ultrasonic Level Measurement*. SIEMENS, 2013.

[13] R. Lookmuang, K. Nambut, and S. Usanavasin. Smart parking using iot technology. *2018 5th International Conference on Business and Industrial Research*, 2018. doi: 10.1109/ICBIR.2018.8391155.

[14] D. Vakula and Y. K. Kolli. Low cost smart parking system for smart cities. *2017 International Conference on Intelligent Sustainable Systems*, 2017. doi: 10.1109/ISS1.2017.8389415.

[15] R. Grodi, D. B. Rawat, and F. Rios-Huitierrez. Smart parking: Parking occupancy monitoring and visualization system for smart cities. *SoutheastCon*, 2016. doi: 10.1109/SECON.2016.7506721.

[16] P. Sadhukhan. An iot-based e-parking system for smart cities. *International Conference on Advances in Computing*, 2017. doi: 10.1109/ICACCI.2017.8125982.

[17] Z. Zhi-yuan, R. He, and T. Jie. A method for optimizing the position of passive uhf rfid tags. *2010 IEEE International Conference on RFID-Technology and Applications*, 2010. doi: 10.1109/RFID-TA.2010.5529867.

[18] Z. Pala and N. Inanc. Smart parking applications using rfid technology. *2007 1st Annual RFID Eurasia*, 2007. doi: 10.1109/RFIDEURASIA.2007.4368108.

[19] L. Mainetti, L. Palano, L. Patrono, M. L. Stefanizzi, and R. Vergallo. Integration of rfid and wsn technologies in a smart parking system. *22nd International Conference on Software*, 2014. doi: 10.1109/SOFTCOM.2014.7039099.

[20] Z. Zhang, X. Li, H. Yuan, and F. Yu. A street parking system using wireless sensor networks. *International Journal of Distributed Sensor Networks*, 2013, 06 2013. doi: 10.1155/2013/107975.

[21] E. De la Hoya, O. Casas, and R. Pallas-Areny. Wireless magnetic sensor node for vehicle detection with optical wake-up. *Sensors Journal, IEEE*, 11:1669 – 1676, 09 2011. doi: 10.1109/JSEN.2010.2103937.

[22] D. Di Mauro, M. Moltisanti, G. Patane, S. Battiato, and G. M. Farinella. Park smart. *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2017. doi: 10.1109/AVSS.2017.8078502.

[23] G. Amato, F. Carrara, F. Falchi, C. Gennaro, C. Meghini, and C. Vairo. Deep learning for decentralized parking lot occupancy detection. *Expert Systems with Applications*, 72:327 – 334, 2017. doi: https://doi.org/10.1016/j.eswa.2016.10.055.

[24] P. R.L. de Almeida, L. S. Oliveira, A. S. Britto, E. J. Silva, and A. L. Koerich. Pklot – a robust dataset for parking lot classification. *Expert Systems with Applications*, 42(11):4937 – 4949, 2015. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2015.02.009. URL `http://www.sciencedirect.com/science/article/pii/S0957417415001086`.

[25] J. Nyambal and R. Klein. Automated parking space detection using convolutional neural networks. *2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech)*, pages 1–6, 11 2017. doi: 10.1109/RoboMech.2017.8261114.

[26] European Commission. 2018 reform of eu data protection rules, 2018. URL `https://ec.europa.eu/commission/priorities/justice-and-fundamental-rights/data-protection/2018-reform-eu-data-protection-rules_en`. Online; accessed 4 march 2019.

[27] Lora Alliance. Lorawan, 2019. URL `https://www.lora-alliance.org`. Online; accessed 22 December 2019.

[28] U. Noreen, A. Bounceur, and L. Clavier. A study of lora low power and wide area network technology. *International Conference on Advanced Technologies for Signal and Image Processing*, 2017. doi: 10.1109/ATSIP.2017.8075570.

[29] J. Hurwitz and D. Kirsch. *Machine Learning For Dummies - IBM Limited Edition*. John Wiley & Sons, Inc., New Jersey, NJ, USA, 2018. ISBN 9781119454946.

[30] E. Alpaydin. *Introduction to machine learning*. MIT Press Ltd, Cambridge, Mass., United States, 2009. ISBN 9780262028189.

[31] T. Peter. *Methods for Business Analysis and Forecasting: Text and Cases*. Wiley, New Jersey, United States, 2005. ISBN 9780471123842.

[32] R. Shanmugamani. *Deep Learning for Computer Vision*. Packt Publishing Limited, Birmingham, UK, 2017. ISBN 9781788295628.

[33] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

[34] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *arXiv*, 2018.

[35] T. Acharya and A. K. Ray. *Image Processing - Principles and Applications*. Wiley-Interscience, New York, NY, USA, 2005. ISBN 0471719986.

[36] Nunamaker J. F. and Chen M. Systems development in information systems research. *System Sciences*, 3:631–640, 1990. doi: 10.1109/HICSS.1990.205401. URL https://doi.org/10.1109/HICSS.1990.205401.

[37] V. Pureswaran and P. Brody. Device democracy - saving the future of the internet of things. Technical report, IBM Institute for Business Value, 2015.

[38] Huawei Technologies Co. Ltd. Nb-iot commercial premier use case library. Brochure, 2017. URL https://www.gsma.com/iot/wp-content/uploads/2017/12/NB-IoT-Commercial-Premier-Use-case-Library-1.0_Layout_171110.pdf. Online; accessed 27 May 2019.

[39] G. C. Lazaroiua and M. Rosciab. Definition methodology for the smart cities model. *Energy*, 2012. doi: doi.org/10.1016/j.energy.2012.09.028.

[40] Ahlam A., Bashayer A., Maram B., Zubaida J., and Maram M. A smart parking solution for jeddah city. *International Journal of Computer Applications*, 171:975–8887, 08 2017. doi: 10.5120/ijca2017915084.

[41] J. Du. Understanding of object detection based on cnn family and yolo. *Journal of Physics: Conference Series*, 1004:012029, 04 2018. doi: 10.1088/1742-6596/1004/1/012029.

[42] G. Amato, F. Carrara, F. Falchi, C. Gennaro, and C. Vairo. Car parking occupancy detection using smart camera networks and deep learning. *2016 IEEE Symposium on Computers and Communication*, pages 1212–1217, 2016. doi: 10.1109/ISCC.2016.7543901.

[43] S. Valipour, M. Siam, E. Stroulia, and M. Jagersand. Parking-stall vacancy indicator system, based on deep convolutional neural networks. In *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pages 655–660, Dec 2016. doi: 10.1109/WF-IoT.2016.7845408.

[44] C. Behle. The multiple benefits of video surveillance in manufacturing, 2018. URL https://www.axis.com/blog/secure-insights/video-surveillance-manufacturing/. Online; accessed 4 march 2019.

[45] Z. Doffman. Smarter cities: Will autonomous ai surveillance and iot now automate law enforcement?, 2018. URL https://www.forbes.com/sites/zakdoffman/2018/12/15/smarter-cities-will-autonomous-ai-surveillance-and-iot-now-automate-law-enforcement/#6bc3676d1d55. Online; accessed 4 march 2019.

[46] G. Vojković. Will the gdpr slow down development of smart cities? *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2018. doi: 10.23919/MIPRO.2018.8400234.

[47] Tsung-Yi Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312.

[48] Scikit-learn. sklearn cluster dbscan. URL https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html. Online; accessed 13 june 2019.

[49] I.V. A journey to clustering. introduction to dbscan, 2018. URL https://medium.com/odessa-ml-club/a-journey-to-clustering-introduction-to-dbscan-e724fa899b6f. Online; accessed 16 may 2019.

[50] D. Franklin. Jetson nano brings ai computing to everyone, 2019. URL https://devblogs.nvidia.com/jetson-nano-ai-computing/. Online; accessed 10 may 2019.

[51] A. Rosebrock. Ubuntu 18.04: How to install opencv, 2018. URL https://www.pyimagesearch.com/2018/05/28/ubuntu-18-04-how-to-install-opencv/. Online; accessed 16 may 2019.

[52] OpenCV. Install opencv-python in ubuntu, 2019. URL https://docs.opencv.org/3.4/d2/de6/tutorial_py_setup_in_ubuntu.html. Online; accessed 16 may 2019.

[53] Python. Python for beginners, 2019. URL https://www.python.org/about/gettingstarted/. Online; accessed 16 may 2019.

Appendix A

Test Cases and Test Results

ID	Description	Expected Result	Pass/Fail
V1	Perform a vehicle detection task when the parking lot is empty	No vehicles should be detected on the parking lot.	PASS
V2	Perform a vehicle detection task when one vehicle is parked in the parking lot.	The system should detect one parked vehicle in the parking lot	PASS
V3	Perform a vehicle detection task when several vehicles are parked in the parking lot.	The system should detect all parked vehicles in the parking lot.	PASS
V4	Detect vehicles in the parking lot even if there are other objects in the image	The system should detect only vehicles and ignore other type of objects.	PASS

Table 2: Vehicle detection test

ID	Description	Expected Result	Pass/Fail
P1	Store parking spot coordinates of detected vehicles when a new car is detected.	The system should store only new parking spaces that does not exist already.	PASS
P2	Store parking spot coordinates of detected vehicles when several new cars are detected.	The system should store only new parking spaces that does not exist already.	PASS
P3	Store coordinates and IoU of a new defined parking spot.	All parking spots should be stored in the JSON file with an IoU value that represent the overlapping with other parking spots.	PASS
P4	Detect a vehicle in an already defined parking spot.	The system should not store and define a new parking spot. Coordinates of current parking spot should be recalculated based on the coordinates of the detected vehicle.	PASS

Table 3: Parking spot detection test

ID	Description	Expected Result	Pass/Fail
C1	The system detects a new vehicle.	The system should store coordinates of each detected vehicles in a separate JSON file that will be used for cluster analysis. The system should increase the number of samples.	PASS
C2	The system detects a vehicle that is not moved since previous detection task.	The system should not increase the number of samples or store vehicle's coordinates.	PASS
C3	The system reaches the samples threshold required for clustering.	The system should perform the clustering task.	PASS
C4	The system performs a clustering analysis task.	The clustering analysis should group the vehicle coordinates together and detect outliers of illegal parked vehicles.	PASS
C5	The system performed a clustering analysis task.	Based on the result, the system should re-define the coordinates of parking spots by calculating the average value of each detected group of samples.	PASS

Table 4: Cluster analysis test

ID	Description	Expected Result	Pass/Fail
O1	The system detects a new parked vehicle	The system should mark the parking spot as occupied.	PASS
O2	The system detects several parked vehicle	The system should mark relevant parking spots as occupied.	PASS
O3	A vehicle leaves the parking spot.	The system should mark the parking spot that the vehicle was parked on as unoccupied	PASS
O4	All vehicles leaves the parking lot.	The system should mark all the parking spots as unoccupied	PASS
O5	A person occupies a vacant parking spot in the image.	The system should not mark the parking spot as occupied	PASS

Table 5: Parking occupancy detection test

ID	Description	Expected Result	Pass/Fail
I1	A vehicle is double parked in the parking lot before clustering analysis.	The system should only mark the covered parking spots as occupied	PASS
I2	A vehicle is double parked in the parking lot after clustering analysis.	The system should mark the covered parking spots as occupied and mark the vehicle with blue bounding box.	PASS
I3	A vehicle is parked on an area that is not a parking spot after clustering	The system should define the vehicle as an illegal parked vehicle. The state of parking spots should not be changed.	PASS

Table 6: Illegal parking detection test

ID	Description	Expected Result	Pass/Fail
T1	The LoRaWAN module is powered up.	The LoRaWAN module should connect the TTN with OTAA mode.	PASS
T2	The LoRAWAN module receives data about parking spots via serial communication.	The LoRaWAN module should compose a message and send to the TTN.	PASS
T3	TTN receives payload from the LoRaWAN module.	TTN should decode the payload and show the received id, number of total parking spots, and available parking spots on the console.	PASS

Table 7: Data transmission test

ID	Description	Expected Result	Pass/Fail
S1	The single board computer sends a http request to the camera, requesting an image	The single board computer should receive an image from the camera	PASS
S2	The system detects vehicles in the captured image.	The system should calculate and store minimum distance between detected vehicles in the image.	PASS
S3	The system loads the JSON file	The system should get all available coordinates of parking spots that were stored	PASS
S4	The coordinates of parking spots is updated.	The system should update coordinates in the JSON file.	PASS
S5	The defined delay time between each detection tasks is 5 minutes.	The system should wait approximately 5 minutes before performing a new detection task.	PASS

Table 8: System test