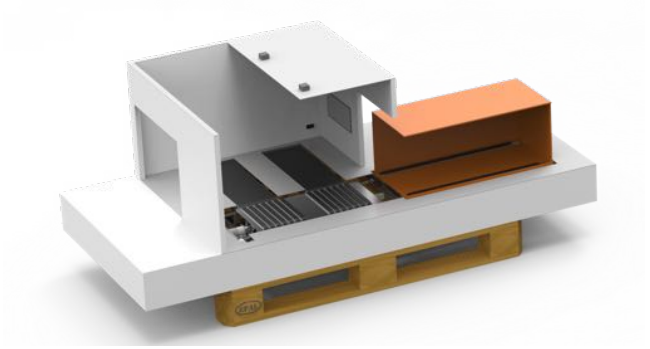
ROYAL INSTITUTE
OF TECHNOLOGY

MF2059
Mechatronics Advanced Course

Final Report

Carocell

Automated Car Park Loading Dock



William Bruce, Douglas Lundgren, Edvin von Otter, Martin Sjögren, Sara Ström, Alexandra Tang, Rebecca Wikström, Martin Åkerblad

Supervised by
Mahmood Reza Khabbazi (Royal Institute of Technology)
Matthew Whelan (Ocado)

December 22, 2017

Abstract

Automated car parks have been around for some time now, but none that has had the focus of parking the cars as densely as possible. This was the company Ocado's vision, which led to this project. Ocado wanted to expand their automated grocery system to function as an automated car park. One key component missing when expanding the concept was the design of a loading dock for the cars. The goal of the project was to be able to provide a smooth handover of the customer's car and quickly and accurately place it in a container, entering the system.

The goal was achieved by designing a mechanical conveyor belt system, for moving a car, and using a computer vision system together with optical sensors. The driver is guided through a monitor showing it's position and allowed parking area. Once the driver has left the car and parking area, the conveyor belts moves the car into a second station, by using the feedback from the camera vision. During this process, the car becomes aligned with high accuracy. A third station will allow the rest of the automated car park system to receive and drop off the cars.

The results were based on a prototype built in the scale 1:10, and showed success in the alignment and handling of the car.

Acknowledgements

We would like to take this opportunity to thank our supervisor at Ocado, Matthew Whelan, and our supervisor at KTH, Mahmood Reza Khabbazi, for sharing their great expertise and for their generous support, coaching and dedication throughout the entire project. We would also like to give our warmest thanks to the manager of the workshop at the mechanical department at KTH, Tomas Östberg, without whom our mechanical demonstrator would not have been finalized. Finally we would like to thank our friend and colleague Todd Barker for a much appreciated peer review of this report.

Team Carocell
2017-12-22

Contents

List of Figures

List of Tables

Nomenclature

fps	Frames Per Second
GUI	Graphical User Interface
HIL	Hardware In the Loop
KTH	Kungliga Tekniska Högskolan (The Royal Institute of Technology)
LAN	Local Area Network
MCI	Micro Controller Unit
USB	Universal Serial Bus
IR	Infra-Red
ROS	Robotics Operating System
SIL	Software In the Loop
OpenCV	Open Source Computer Vision
WYSIWYG	What You See Is What You Get

1. Introduction

1.1 Background

Few people have managed to avoid the tedious mission to find a place to park their car, during times when they are nowhere to be found, and do so smoothly. It also comes with some other risks, for example a more reckless driver might put a dent in your car, or if someone with other ethical believes might break in and drive away. Needless to say, there is room for improvement for the way parking is done today. With an automated car park, these worries, along with many others, could be put at ease. It gives a higher parking density than conventional car parks and depending on the system, it can be made completely unavailable for other people to gain access to the cars. It also makes the parallel parking obsolete - which likely is well appreciated by many.

Today cars are approaching being fully autonomous and could within a not too distant future be a used without human supervision - it is time for the parking solutions to match our cars' intelligence.

The project was brought by Ocado who wants to implement a scaled and adapted version of their existing automated grocery system to store a wide array of passenger cars in containers instead. To accomplish this, the current system is to be scaled up and a new drop-off station is to be designed to fit the purpose of handling cars.

1.1.1 Ocado

Ocado is the largest dedicated online grocery retailer in the world [5]. They use an automated storage system based on a 3D grid of racks with tracks on top. Boxes with products are placed on top of each other in the racks and the boxes are picked up from the stacks in the racks and transported by robots that travel on the tracks on top of the racks. The boxes are dropped off at a picking station where the orders are packed and shipped.

1.1.2 Automated Car Parks

An automated car park is a parking system that lets a customer leave their vehicle in a designated drop-off area from which it is then automatically moved to a storage space and later returned to the owner at a pick-up area [6].
There are existing solutions for an automated parking garage but this project aims to design a more volume efficient, scalable and reliable solution.

1.2 Project Description

The goal for this project is to design and build a prototype of the Car Park Loading Dock, i.e. the drop-off/pick-up station. This prototype should be used to show and test core functions of the proposed design, such as movement and alignment of a car. The Loading Dock should ensure a highly automated system, where user involvement is minimized.

The final deliverable is a prototype, scaled 1:10, of the Loading Dock. This prototype should be able to achieve the following:

- Ensure parking with a certain accuracy independent of the skill of the driver
- Achieve as high accuracy as possible of the positioning system to minimize the required area of the containers and therefore optimize the required volume of the entire car park
- Perform moving actions as fast as possible without endangering users or damaging the car

The focus will be on ensuring parking with a certain accuracy independent of the skill of the driver. The concepts surrounding this are non-trivial and there are many different approaches, however, the Project Owner would like to have a unique solution.

Another essential part of the project is to ensure that the storage units are as cheap as possible. This means eliminating any moving parts and minimising the size of the containers.

Failing this part would contribute in the sense that the Project Owner could approach the problem differently in the future.

1.2.1 Scope and Limitations

This report focuses on the loading dock for the automated car park currently being developed by Ocado, meaning the part from that the driver enters the drop-off area for the car until the car has been handed off to another system which places the car into the container, i.e. the parking lot.

The largest car that will be considered in this report is an Audi Q7. Since the dimensions of the actual car park loading dock will be a minimum of the

dimensions of an Audi Q7, i.e. 5.052 x 2.212 m and will be handling weights up to 2 tonnes, it has been agreed to with Ocado to develop a model in scale 1:10. This scale was chosen to still be able to successfully carry out proof of concepts.

1.3 Requirements and Delimitations

A set of requirements and delimitations have been set to provide guidance in the project execution so to stay on course. Below are two kinds of requirements defined;

Strict: these are what **must** be accomplished to comply with the desired system functionality.

Soft: these refer to some implementation that would enchance the systems functionality and **should** be implemented, if time would allow it.

The requirements have been specified below:

1. **Strict Requirements:**
   (a) The system shall be able to extract the cars position (x, y and angle) when inside the system boundary.
   (b) The car shall be transported without contacting any fragile surfaces on the car.
   (c) The driver shall have feedback to assist when parking the car in the drop-off area.
   (d) The system shall be able to complete an entire hand-off cycle, meaning deliver the car to the pick-up station.
2. **Soft Requirements:**
   (a) The project should investigate the feeble subsystems of the concept design.
   (b) A user friendly GUI should be in place for debugging and visualisation of the system operations.
   (c) The project should investigate if the system could achieve centimetre precision in a down-scaled version of the system.
   (d) The project should conclude how the precision scales.
   (e) The demonstrator should be built in a modular way so that components and debugging is accessible after montage.

1.4 Report Disposition

This report is divided into nine chapters. Chapter 2 deals with the State of the Art analysis and literature studies, discussing the findings in previous work. Chapter 3 explains the proposed concept and Chapter 4 describes methods and tools used in the implementation of it. In Chapter 5, parts and systems that are used in the prototype are presented. Chapter 6 explains the process of verifying and validating the implementation and Chapter 7 the results from the verifications are presented. In Chapter 8, the results are discussed and conclusions are drawn. Finally, in Chapter 9, future work and recommendations are presented.

2. State of the Art and Literature Review

2.1 Existing Automated Car Parks

In this section, the current automated car parks will be described, both in terms of the overall type and of the specific ways of operation that are relevant for this report, i.e. how the driver parking guidance is done, how to deliver the car back to the driver with the car's front in the driving direction and how the current systems are transporting the cars from the parking bay to the allocated slot.

2.1.1 Types of Parking Systems

There are several types of automated parking systems currently available with various building structures and different vehicle parking procedures that can generally be divided into elevator typed, rotary typed or rack-and-rail typed parking systems.

Elevator Type

The car is vertically lifted to the level on which a parking space is available and is then moved horizontally into the allocated parking space. This type of system is used by Volkswagen Group in their car storage facilities Car Towers in the visitor attraction site Autostadt, Germany, see Figure 2.1.

Rotary Type

The rotary type systems are similar to Ferris Wheels in the appearance. The driver parks the car directly onto a parking lot which is later lifted by the system to move the parked car away from the bottom central position, hence allowing for next car to enter the system. Such system has been constructed by Parkmatic on Tillary Street in downtown Brooklyn, see Figure 2.2.

Figure 2.1: Volkswagen's elevator typed automated car collection system in Autostadt, Germany. Source: autostadt.de

Figure 2.2: Rotary type automated car park in downtown Brooklyn. Source: Parkmatic

Rack-and-Rail Type

This system is similar to the elevator type but has generally more complicated structures of the parking lots on each level. There are therefore horizontally

moving carts that are taking over the transportation of the cars and place it in its appropriate parking space after it has been lifted to the right level. This type of system has been implemented in the One York building in New York, see Figure 2.3.

Figure 2.3: A rack and rail typed parking system in New York provided by Park Plus. Source: Park Plus

2.1.2 Ways of Operation

In contrast to conventional multi-level parking, an automated parking system is where the driver leaves their car at a parking bay where the system takes over and transports the car to and from a designated parking lot.

Parking Guidance for Driver

The driver is commonly guided to drive to and park at a specific position in the parking bay, e.g. by following tracks on the pallet as can be seen in Figure 2.4 or by placing the car within the outlines of a pallet as shown in Figure 2.5.

Having Car Front in the Direction of the Exit

There are parking bays with either one or two ports into the parking area. When having two, one is for entering and the other for exiting, see Figure 2.7 and Figure 2.6. This is to reduce the amount of time spent in the bay, allowing the car nose to be in the direction of the exit path. In cases where the parking bay share entry and exit port, a turntable is introduced which rotates the car 180°. The vehicle rotation is either done in the parking bay or in a later stage when it has left the area.

Ways of Transporting Car to Allocated Slot

After the driver has left the parking bay, the car is transported by either a non-pallet solution lifting it by the wheels or by a pallet solution.

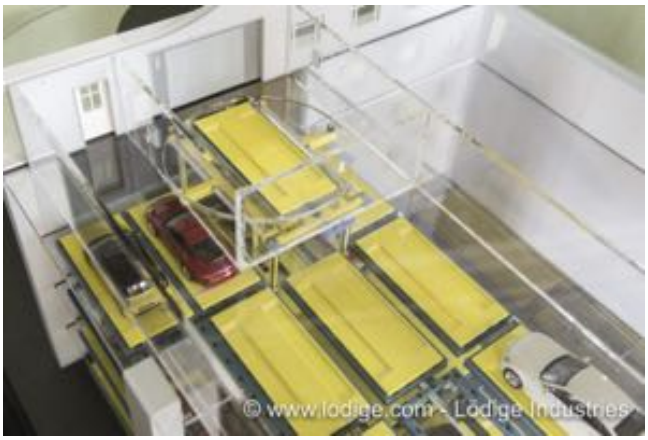


Figure 2.4: Concept by Lödige where driver is guided to follow the tracks on the pallet. Source: Lödige Industries

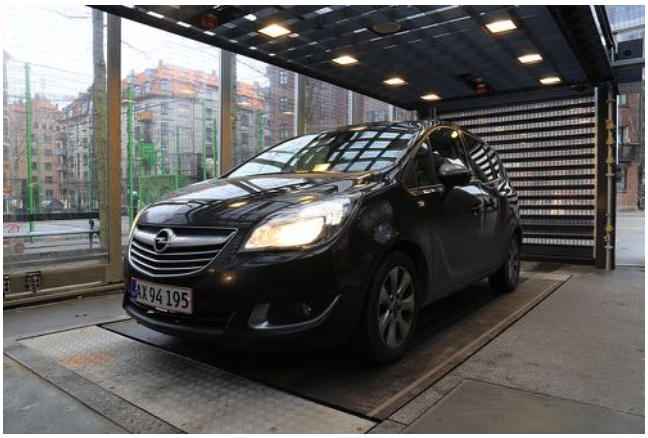


Figure 2.5: Driver is directed to park within the outlines of the pallet. Source: Westfalia Parking Solutions

The American company Park Plus is one company that has implemented the non-pallet solution in their Palletless Comb Exchange Transfer System. The parked cars are standing on a base with a combed surface in order for a robot to be able to access the wheels of the car when it needs to be transported, see Figure 2.8. The driver is parking the car so each wheel is standing on the combed surface and it is then transported by a cart coming in between the combs, lifting it by the wheels [7].

Figure 2.6: The entry port of the two port parking bay in Conrad Hotel & Office Tower. Source: Westfalia Parking Solutions

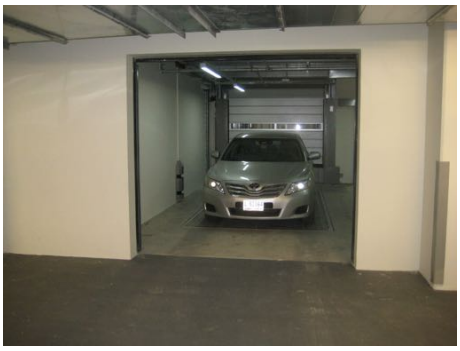


Figure 2.7: The exit port through which the car is driven through when exiting the parking bay. Source: Westfalia Parking Solutions

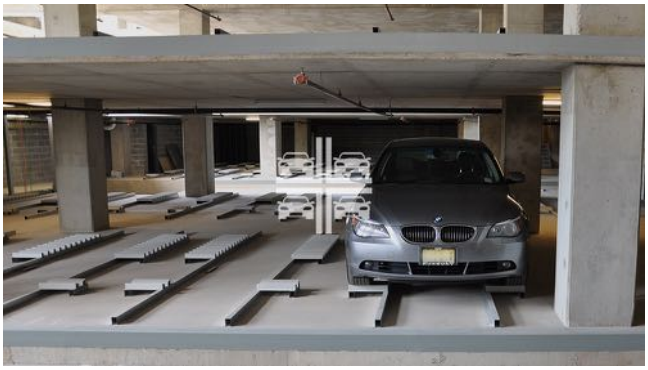


Figure 2.8: Non-pallet solution with carts lifting the wheels of the car by coming in between the combs. Source: Park Plus

Another company that has developed a non-pallet transportation solution is the German company Lödige Industries. The car is lowered down from the bay and collected by an ultra-flat horizontal transfer robot controlled by Bluetooth and Wi-Fi, see Figure 2.9. The robot drives underneath the car and unfolds its arms, squeezing each wheel from two sides, hence raising the entire vehicle [8].

In pallet-based solutions the driver is parking the car directly onto the pallet in the parking bay. The pallet is then lowered vertically out of the bay and moved by an additional lifting unit that can move and rotate in two dimensions. Hence, the car remains on the same pallet throughout the whole parking procedure. Such systems have been used by Westfalia Parking Solutions in their automated parking system in Leifsgade Copenhagen [9], as well as by Park Plus in New York [10], see Figure 2.5 and Figure 2.10 respectively.

Figure 2.9: Non-pallet solution with carts lifting the wheels of the car by unfolding two arms under each wheel. Source: Lödige Industries

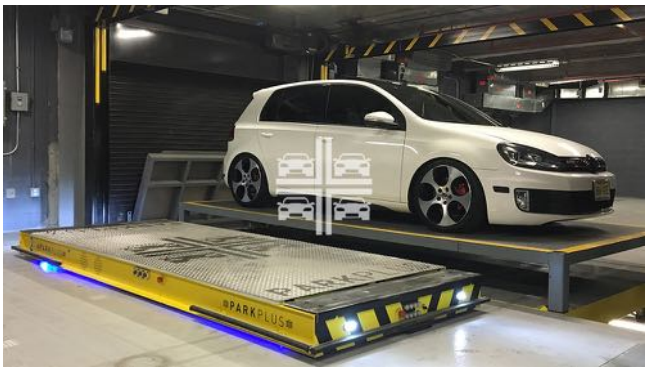


Figure 2.10: Pallet-based solution with carts lifting the pallet on which the car is parked on by the driver. Source: Park Plus

2.2 Conveyor Belts and Design

Conveyor belts are used in many different applications, conveying semi-finished and finished industrial products between different locations. These come in a variety of shapes and sizes, materials and structures. In the simplest setup, the conveyor consists of five main components: a driving pulley, a tail pulley, a conveyor belt, support structure and some kind of tensioning mechanism. Mainly, the design focuses around the support structure and drive system.

2.2.1 Support Structure

The support structure must be rigid, not giving way by flexing from forces when the belt is tensioned. There are two commonly used support structures, slider beds or rollers.

Slider beds, as the name suggests, are beds of material upon which the belt slides, seen in Figure 2.11. The advantage of this is that the belt runs without any influence on the belt tracking. Using this method, the choice of materials on the belt itself and the bed is of great importance. However, if correctly chosen, the friction and noise can be decreased to acceptable levels for the drive system.

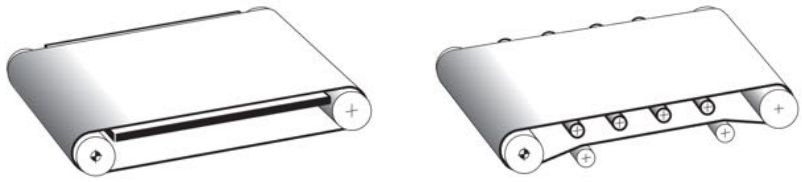


Figure 2.11: Different types of support structures for conveyor belts. Left: Slider bed, Right: Roller support. Source: Habasit

In addition to slider beds, there are support structures that rely on rollers, seen in Figure 2.11. These are used when the transporting distance is long and the weight of the item is high, because of the reduced friction. The rollers are most commonly made out of steel and mounted with roller bearings.[11]

2.2.2 Drive System

The drive system, including the motor drive and tensioning mechanisms, can be found in various implementations. There are three main categories, as seen in Figure 2.12: head drive (drive on the carrying side), tail drive and centre drive. Head and tail drive is most often used in applications where the direction of the

belt is assumed to be the same, whereas when the direction is to be reversed, centre drive is preferred.

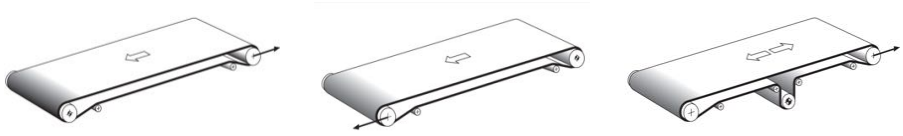


Figure 2.12: Different types of drive systems. Left: Head Drive, Centre: Tail Drive, Right: Centre Drive. Source: Habasit

Tensioning is done preferably by using pulleys that can be arranged so as to get optimal tension over the driving pulley. There are different approaches, either using fixed tensioning, done during installation, or constant force tensioning, enabled by using spring-loaded or hydraulic tensioning. [11]

2.3 Motion Detection Algorithms

Computer vision has been a hot topic for a long time and the swarm of algorithms that has been designed for manipulating images has been more or less bundled up in an open source library called OpenCV. It consists of several modules like memory management, image filtering, motion analysis, shape analysis, feature extraction etc. and to realize the strategy for what modules to use for a specific project is not a trivial task [12]. Some of the ideas that have been researched, and might be relevant for this project, are presented below.

2.3.1 Incremental Learning for Robust Visual Tracking

To cope with large variations in the appearance of objects as well as surrounding illumination, a paper from the university of technology in Toronto have suggested a method where the information about what defines the appearance of the model is updated during runtime. The algorithm is based on a statistical procedure called principal component analysis (PCA), which the main features are to update the sample mean of the model with a forgetting factor which weighs more recent observations more heavily than older ones. In Figure 2.13 one can see a car being tracked and the low-dimensional representation of it being updated continuously.[1]

2.3.2 Superpixel Tracking

When an object is camouflaged by its environment and a person has a hard time to see if something is there, an unconscious strategy is to try to identify

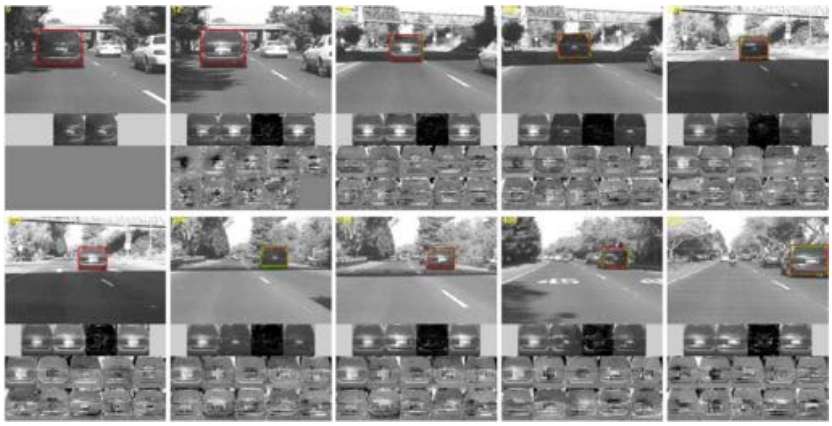


Figure 2.13: Tracking the motion of a car and the corresponding low-dimensional representation of it. Source: [1].

irregularities in the image and then make sense of what object those observations belong to. The Superpixel Tracking method is based on this principle; dividing an image into background and foreground. It does so by computing a target-background confidence map and chooses the best suited pixels for the foreground/background with the help of a Bayesian statistic method called *Maximum A Posteriori Estimate (MAP)*. The process of this algorithm is illustrated in Figure 2.14, where the motion of a teddy bear is tracked when dropped in front of a camera [2].

Figure 2.14: The confidence map of a teddy bear when it's falling in front of a camera. Source: [2].

2.3.3 Integrating Tracking with Fine Object Segmentation

A research project carried out at the University of Crete suggests a method for tracking objects where the object is initially segmented with high accuracy and then a less computationally demanding algorithm takes over to continue tracking its movement. The paper examines kernel-based object tracking in combination with a Random Walker-based image segmentation method to en-

able a probabilistic fusion between colour and spatial cues. A state-diagram of the process can be seen in Figure 2.15 [13].

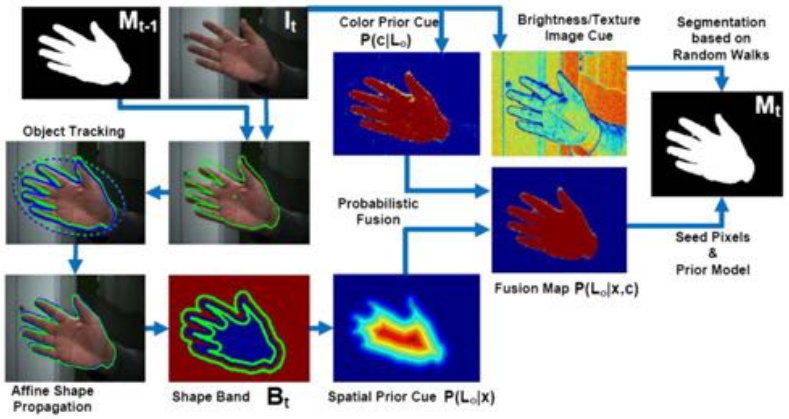


Figure 2.15: State diagram of the algorithm, from a fine tuned segmentation of a hand segmentation to the next sample time update.

2.4 Parking Aid Systems

The numerous parking aids that exist on the market in newer cars today provide precision in parking through an increased awareness of the cars' surrounding areas. The aids that will be considered in this report are various parking sensors and their different applications. These aids are usually combined in parking assist systems [14]. These systems allow parking to be performed almost autonomously. The system can judge the size of an available parking space, inform this to the driver of its decision and handle the manoeuvres required to perform the actual parking, i.e. the driver only needs to control the speed. These systems are capable of both parallel and bay parking manoeuvres. Some examples of vehicles that have implemented parking assists are Ford Focus, Volkswagen Golf and Seat Leon [14]. Theories from this section can be used in the application of detecting cars.

2.4.1 Proximity Sensors

Sensors that have the ability to detect the presence of nearby objects without physical contact is called proximity sensors and are, because of this property, widely used in parking sensors. The sensors are usually mounted on top of the bumpers and deliver a sound to the driver if there are detected objects within its range. The beeping sound increases in frequency as the vehicle decreases its

distance to the detected objects. Some systems even have a graphical representation giving an approximate view of where the detected object or objects are relative to the car as Figure 2.16 illustrates.

Figure 2.16: The graphical interface for a type of proximity sensor system. Source: Team-BHP.com

2.4.2 Parking Sensors

Parking sensors depend on either electromagnetic or ultrasonic technology. Ultrasonic sensors are less costly compared to the electromagnetic ones, but holds many disadvantages in comparison [15], with this specific application in mind. Ultrasonic sensors requires regular maintenance, to take care of dirt or misalignment caused by, for example, the sensors being bumped into, which affects its performance [16]. Furthermore, the ultrasonic sensors also have a tendency to not detect certain objects due to their position, geometry or material qualities as Figure 2.17, 2.18 and 2.19 illustrates.
Electromagnetic sensors on the other hand, requires no maintenance and are placed inside the bumpers of the car and therefore avoid potential damage. The sensor can be disturbed by heavy rain, so its sensitivity has to be adjusted accordingly [15].

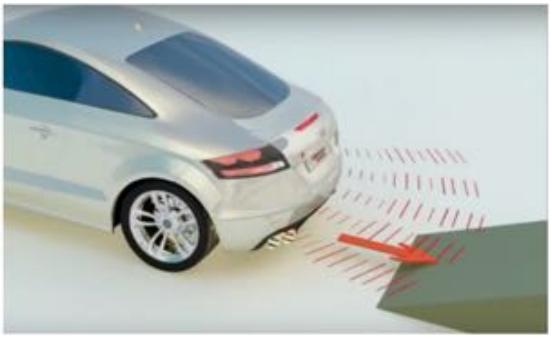


Figure 2.17: The ultrasonic sensors can't detect objects beneath, above or in between the sensors and they are i.e. blind spots. The electromagnetic sensors don't have this issue. Source: Video [3]

Figure 2.18: Narrow objects can fall in between the blind spots of the ultrasonic sensors. Normally a bumper contains six to eight sensors. Source: Video [3]

Figure 2.19: Ultrasonic sensors have a difficulty detecting inclined or poorly reflective objects. Source: Video [3]

3. Concept Design

The concept is limited to the loading dock and is to be constructed so that the car can be picked up by a Storage System that is not concerned within this project. To be compatible with the other system needs the car to be placed inside of a container in the final stage, the container also needs to be accessible for being picked up from above.

The concept has been derived whilst continuously striving towards a mechanical design that:

- Minimises the amount moving parts.
- Avoiding complex movements.
- The scaled up construction should be easily enforced to withstand the heavy load which it will have to handle.

To comply with the points above, the concept relies on having a stage where conveyor belts emerge from slits in the bottom of the container, illustrated in Figure 3.1. Usage of conveyor belts is advantageous because they allow for a simple and robust design and one actuator could be used for several belts if needed.

Figure 3.1: Belts emerging from slits in the container floor.

Combining this with a previous station where the car could be aligned and placed on a plate would allow the system to insert only the car and a simple steel construction. Meaning that each and every parking space would gain a substantial price and complexity reduction.

In order to place the car on a plate while not losing the possibility of making adjustments to the car's rotational orientation, another station is needed beforehand. This station would also need to place the car onto a plate so that the car could be moved. The station will have a two sets of narrow belts that can fit in matching slits made in a transportation plate so that the wheel of the car have somewhere to rest during lifting. This is illustrated in Figure 3.12.

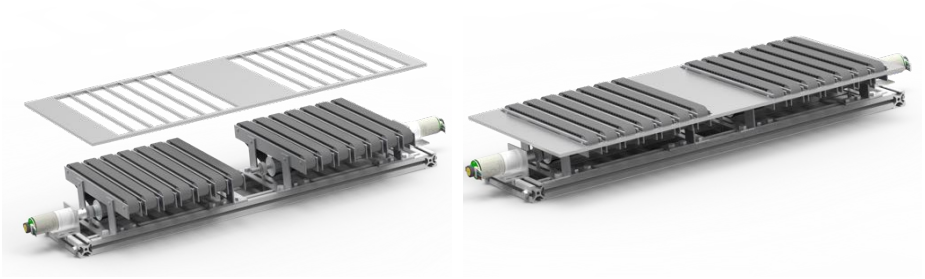


Figure 3.2: Belts and plate

The plate would be elevated by a lifting mechanism which also has conveyor belts for forward and backward motion, further and more detailed descriptions of the mechanical concepts are accounted for in upcoming sections.

With the mechanical systems swiftly explained above, it is in principle possible to achieve a narrow fit into a container without putting cost or maintenance heavy structures - which complies with the stakeholder requirements.

The loading dock will be used for drop off as well as pick-up of the car. With these two stations the loading dock could serve one costumer at the time. By placing two identical stations on each side of the alignment and lifting station this process could be more efficient - this would mean a total of four stations, two of which are identical, see Figure 3.3. By having these side stations the system could allow one driver to park their car while another driver could have their car being brought to them - making the system more effective in terms of costumer flow.

The system relies heavily on knowing the car's position when it is in station 1 and station 2 - this information is extracted by implementing a computer vision system (CVS). This system is then used in combination with distance measurement sensors to send actuation commands to motors which operates the mechanical systems.

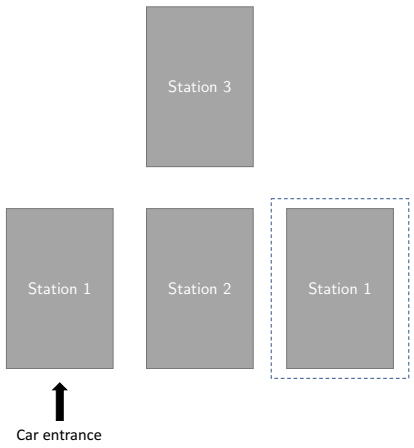


Figure 3.3: Station 1: Driver parking, Station 2: Alignment and transportation and Station 3: Storage system pickup.

3.1 System Overview

Before diving into details regarding how each station is built and their mechanisms, a system overview is here described to give an better understanding of the overall system logic. The system as a whole can be divided into two main subsystems:

Computer Vision System: It is in charge of extracting the car's position as well as supplying the driver with instruction while parking his or hers car.

Mechanical System: Safely align and park the car into a container based on the inputs provided by the Computer Vision System.

How each station uses these systems are shortly described below:

Station 1: Is where the driver will park its car with the help of a guidance system to help the driver park within the area in which the car can be handled by the system. The guidance system is a computer vision system together with two IR sensors [17] providing feedback to the driver, whilst also working as sensors for the mechanical system when the driver has left the parking bay and the mechanical system starts the process of transporting the car into Station 2.

Station 2: Is where the car is aligned to fit into the container in station 3. Station 2 also contains a computer vision system that works as a sensor for determining the cars position and angle, that is used in order to align the car.

Station 3: Is where the container, in which the car should be placed, is situated. When the car has been aligned in Station 2, the car is transported here and then handled by the Storage System.

The work flow and how and when the Computer Vision and Mechanical System is used is described in Figure 3.4.

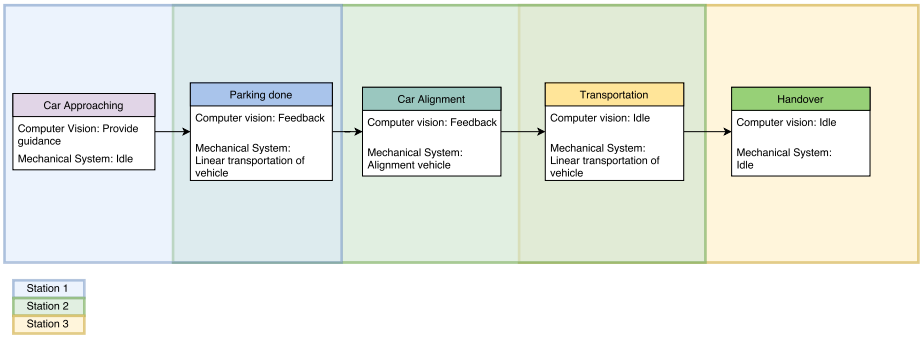


Figure 3.4: The system's work flow.

The system is large and has many complex parts that will work and be controlled in synchronization. In Figure 3.5, the System Architecture and the communication between the subsystems, or nodes, are displayed. In large, the system contains the following nodes:

- **Main Control Node:** Handles the systems overall states, GUI and master communication. This is broken into three controllers: Mechanical and Visual Control as well as UI Control.
- **Visual System Node:** Handles computer vision states, communication between computer vision nodes and Main Controller and image processing.
- **Mechanical System Node:** Handles mechanical system states and communication between mechanical nodes and Main Controller. Actuation commands stem from the Visual System Nodes calculations.
- **Mechanical Nodes (Motor Nodes):** Nodes hosted on the Arduino platform for controlling mechanical hardware.
- **Computer Vision Nodes:** Nodes for handling camera input and streaming to the Visual System Node. Hosted on Raspberry Pis.

Hardware and Software used in each node is further described in the sections below. Communication is realized through Robotics Operating System (ROS), also described in further detail below.

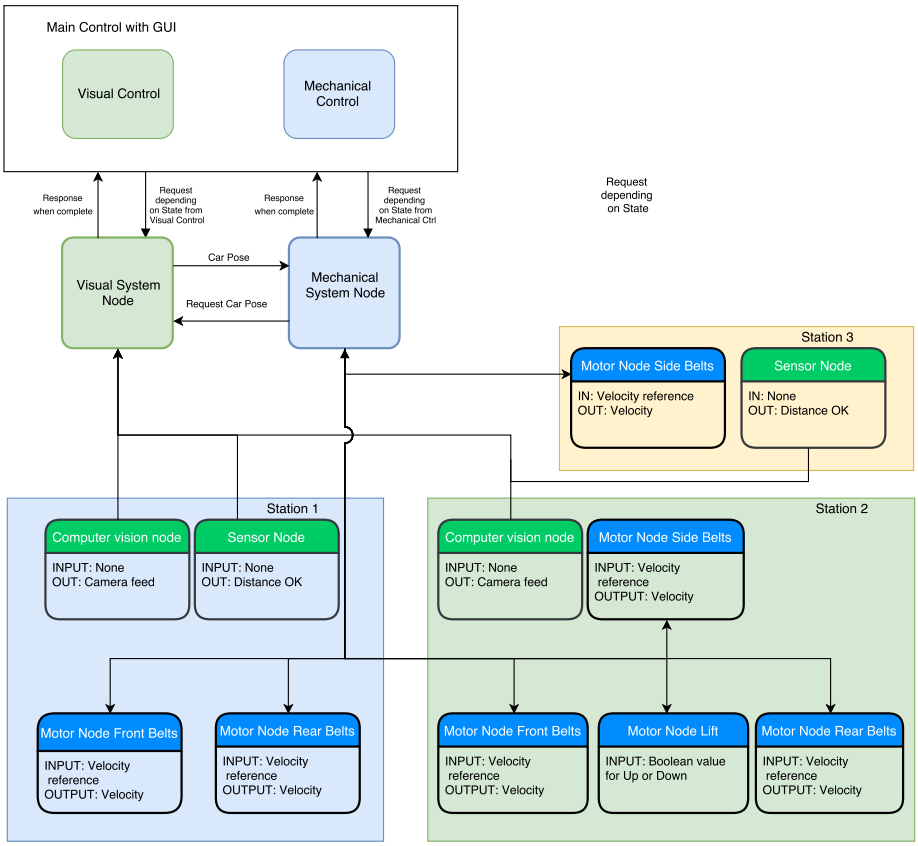


Figure 3.5: System Architecture and Communication.

In Figure 3.6 there is a sectioned view of the concept where one can see all stations. In station 2 there is a plate with slits that will be used to separate the car from station 2 and transport the car to station 3, more detailed descriptions can be found in the corresponding station's subsection.

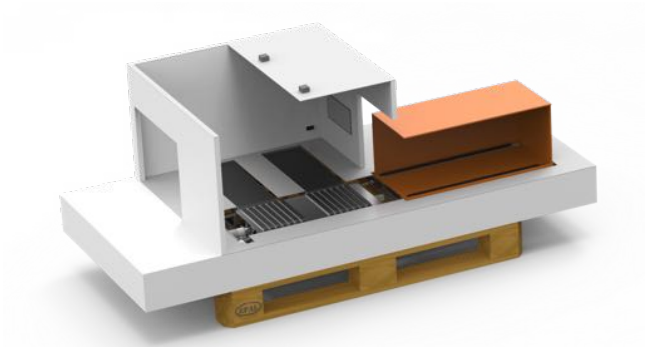


Figure 3.6: Section view of the whole system

The dimensions of the stations in the demonstrator is listed in Table 3.1.

Table 3.1: Dimensions of the stations in centimetres.

Station 1	**Station 2**	**Station 3**
37x61	28x64	25x60

3.2 Station 1

Station 1 is the station in which the driver will park the car. Since it is exposed to customers, the design of the station has to be as non-distractive as possible, with minimised gaps in the floor to reduce the risk of having personal items such as keys disappearing in them.

3.2.1 Design

The purpose of the mechanics of the station is to transport the car into station 2 and perform some initial alignment during the transportation. Therefore, at least two conveyor belts are needed, one for moving the front wheels and one for moving the back wheels. For customer convenience according to the above mentioned preferences, two conveyor belts are chosen. These are designed to cover an as large area in the left-right driving direction as possible to increase the error with which the driver is allowed to park, see Figure 3.7.

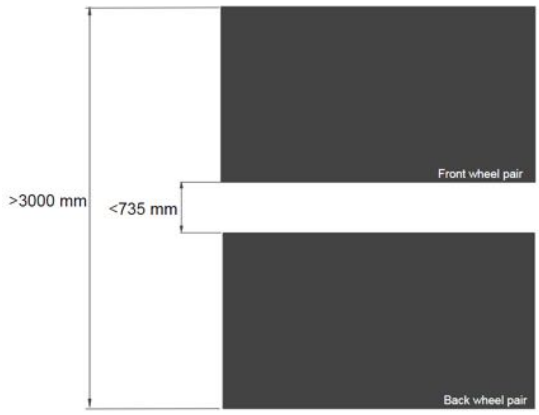


Figure 3.7: The required measurements of the belts based on the size of the largest (Audi Q7) and smallest (Smart Car) wheel pair.

Belt Dimension

To ensure that all cars will be transportable by the belts, the distance between them has to be adjusted to the smallest and biggest car that will be handled by the system. The distances are 735 mm for a Smart Car versus 3000 mm for an Audi Q7. In order to fit the front and back wheels of all cars in between in size, the distance between the belts has to be a maximum of 735 mm and the width of the belt has to be at least 3000 mm from one edge to the other, see Figure 3.7.

Assembly

The whole station 1 is built of two symmetrical parts, one for each belt. Each part contains one belt, two pulleys, four rollers, five axles and one motor. The belt design is standing on a base of aluminum profiles. The whole station can be viewed in Figure 3.8.

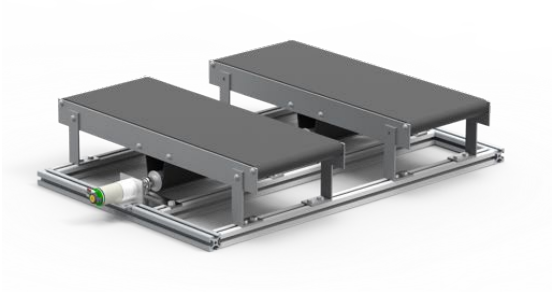


Figure 3.8: Assembled station 1

3.2.2 Drive System

The belts are centre driven due to the fact that they are required to be driven in both directions. Since the two belts have one motor each it is also possible to drive the belts at different speeds and different directions relative to each other, which is how the car is aligned in station 1. This is controlled according to the controller explained in Section 5.2.4.

Belt Tensioning

The tensioning of the belts is done by a shaft going through the pulleys with end rods on each side that can be adjusted up- and downwards and by having additional rollers on both sides of the shaft, as illustrated in Figure 3.9. The fixed tensioning by the rollers is designed to increase the cog teeth contact between the pulley and the belts and thereby increase the efficiency.

Figure 3.9: Conveyor belt design for station 1.

3.3 Station 2

Station 2 is divided into three parts; (a) aligning the car, (b) lifting the car and finally (c) transport the car to station 3. They are all described below independently before showing how there are assembled.

3.3.1 Alignment

Assuring that the car will fit within the container, the system must be allowed to make some alignments in station 2. By having a plate with cut-outs that correspond to the width of a conveyor belt which in turn has been decided based on the wheel diameter for the appropriate scale, the belts can emerge through the plate to make adjustments before the plate is lifted together with the car. The design of the belts that are used for car alignment in station 2 are illustrated in Figure 3.10.

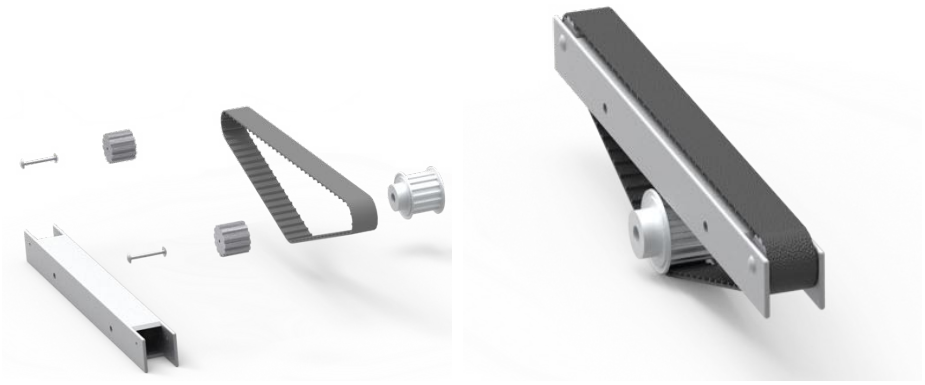


Figure 3.10: Conveyor belt design for station 2.

Seven of these belts are then mounted closely together before being used as a set. Two sets then constitutes for the alignment where one of these are used for the front wheel pair and the other one used for the back wheel pair. The belts are stretched by having a shaft going through the pulleys and end rods on each side that can be adjusted upwards and downwards by tightening and loosening of a nut. A set of these belts are shown in Figure 3.11.

Figure 3.11: Set of belts used for alignment

These belts are mounted on a rectangular aluminium structure with a motor on each side to enable separate control of the belts. How the belts come together with the plate and belts can be seen in Figure 3.12, here it is also shown how the belts will emerge through the plate to make the belts accessible by the car's wheels.

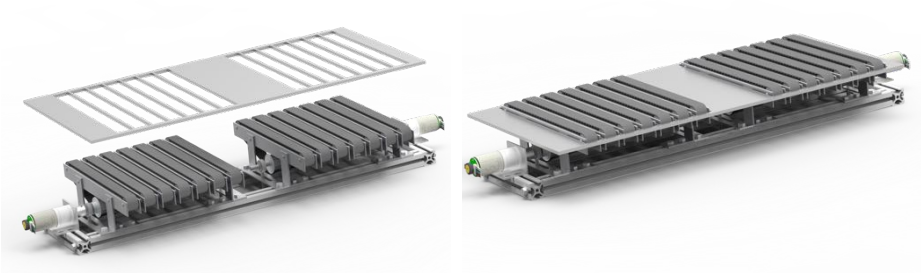


Figure 3.12: Belts and plate

By having this plate with slits and the thin conveyor belts the plate and car can be moved into station 3 without bringing any of expensive or fragile components along with it - in turn this makes for a robust and economical parking space.

3.3.2 Lifting Mechanism

The concept design makes use of a screw mechanism to lift/lower the car and plate. By having two joined screws with different threading (left and right) and their corresponding nut, the mechanism can either move these nuts towards each other or away from each other based on the rotational direction of the actuator. Figure 3.13 shows what this screw and its parts look like.

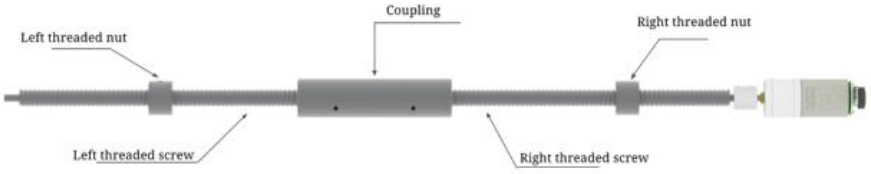


Figure 3.13: Assembly of the screws, with coupling and nuts.

This is combined with a set of guides to transform the movement of the nuts so that the beams on which the plate rests can move in an up- and downwards fashion. This is illustrated in Figure 3.14

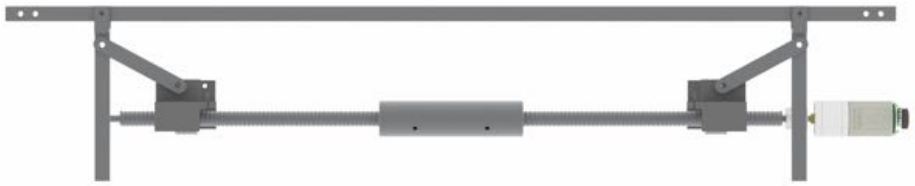


Figure 3.14: Side view of the lifting mechanism with guides

This design makes lifting precise and powerful, via the screws there will be a substantial gearing and therefore there is no need for a powerful actuator.

3.3.3 Transportation

After the car has been lifted and separated from the aligning belts, the car and plate need to be moved. On the beams in Figure 3.14 there is a conveyor belt, hereinafter referred to as "side belts", that will transport the plate and car from station 2 to station 3. These are operated by one actuator and are connected by a shaft on which pulleys have been mounted, Figure 3.15.

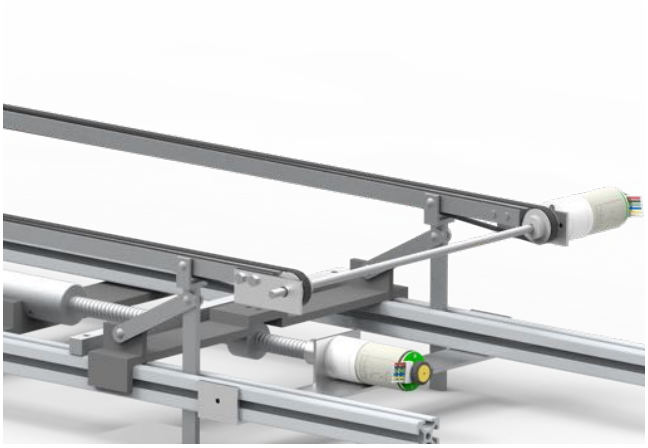


Figure 3.15: Assembly of side belts. The actuator is connected to the belts with a shaft fitted with pulleys

In order to assemble and tension the belts, two plates with slits cut out are fastened at the end of the beam, making the beams length adjustable, Figure 3.16.

Figure 3.16: Tensioning mechanism for side belts

Figure 3.17 displays the lifting mechanism after assembly. This design relies on low friction between a large part of the connecting components, limiting its height adjustability.

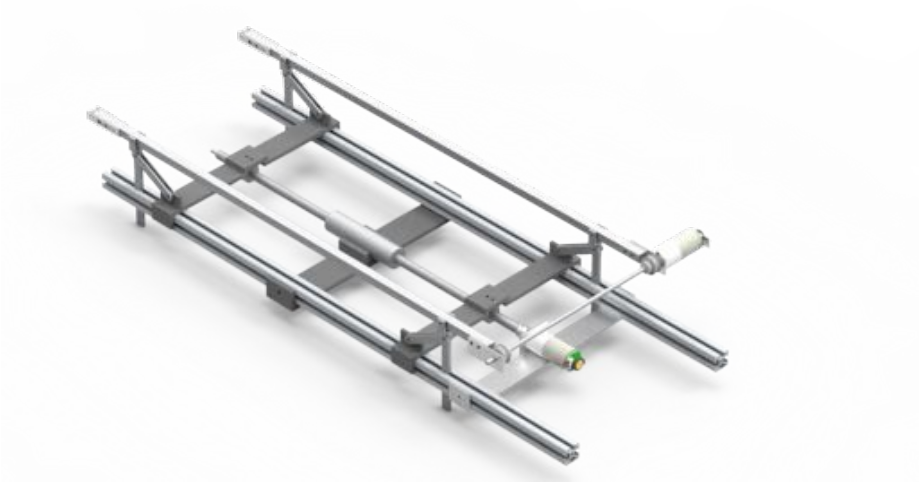


Figure 3.17: Complete assembly of the lifting mechanism

3.3.4 Assembly

In order to work on the *lifting* and *aligning* mechanisms simultaneously they are two separate systems which then can be assembled. Inside of the side belts can the alignment part be inserted so that it rests on two racks. These racks can be moved along on the base frame of the lifting mechanism so to adjust the position of the alignment.
In Figure 3.18 one can see how the alignment part of the station comes together with the rest. They are separated with two adjustable steel beams to enable some assembly alterations to be made if need be.

Figure 3.18: Assembled station 2.

3.4 Station 3

The purpose of Station 3 is to collaborate with Station 2 and safely place and retrieve the car on the plate, to and from the container. This is primarily done by having conveyor belts placed and aligned along the (raised) side belts from Station 2, so that the hand-over can be smooth while minimising the risk of damaging the car, see Figure 3.19.

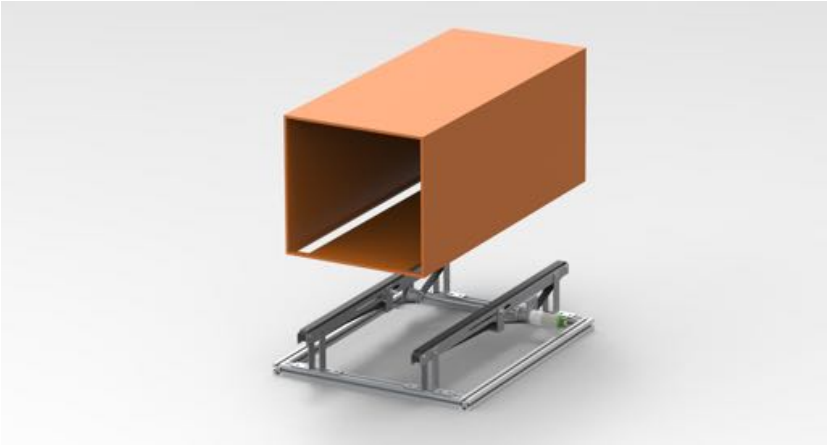


Figure 3.19: Station 3 with the container on the way to be placed at the station.

Figure 3.20: Station 3 with the container ready to handle the car.

3.4.1 Design and Drive System

The design is very similar to Station 1 and 2, but the main difference is that it has belts only in the forward-backward direction. For the required function of having Station 3 both receiving and sending the plate, the belts were designed to be able to move in both directions with the same motor.

The belts are placed as continuations of the Station 2 belts translating the plate to Station 3, so the plate will run smoothly in the transition. As can be seen in Figure 3.20 the container has cut-outs in the geometry of these belts in order to fit when the plate arrives from above. The belts are tensioned by the same kind of roller mechanism as in Station 1 so as to make the contact of the belt's teeth satisfactory for the load during the movement.

3.4.2 Operation

When the car is ready to be transferred into Station 3, it is important that the container has already been placed there, prior to the movement of the plate and car. When the plate has been successfully moved over to Station 3 with the help of the conveyor belts and signal from the IR sensor, the container can be picked up by the rest of the automated parking system. The later on upwards translation of the container will pick up the plate and let it rest on the container's narrow edges and bottom on the containers inside.

4. Method

In this chapter, the system architecture, methods and tools that have been used to realise the project are accounted for.

4.1 Project Development

Throughout the project phase, the development process has been following the V-model, see Figure 4.1, starting with defining requirements from the project description given by the stakeholder. These have later on been narrowed down into a functional specification, high level design and lastly into detailed design before the start of the implementation phase.

Unit tests, such as SIL and HIL, were performed to test the implemented functions separately and validate that those worked according to the specifications in the detailed design. These functions were later tested together with other parts of the same subsystem to validate the overall function that was being implemented. The system in its whole, with all its subsystems was lastly validated against the overall product requirements.

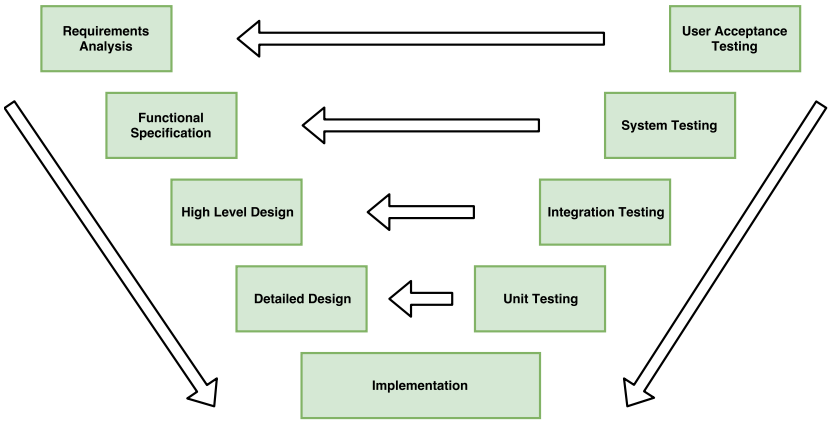


Figure 4.1: The V-model.

4.2 Project Management

The project has been managed according to the agile project development method Scrum. Scrum is a managing method where the work is divided into actions which can be completed within typically two-week iterations called sprints, with daily scrum meetings for tracking progress and possibility to re-plan actions [18]. Each sprint was followed by a sprint review and retrospective to discuss the completed and uncompleted tasks planned for the sprint together with the stakeholder. Continuous process improvement actions were also identified and agreed on.

4.3 Computer Vision

The computer vision system is mainly using classes and functions in OpenCV to process the images in Station 1 and 2, with Python as programming language. The algorithms in OpenCV are all implemented in C++, but have wrappings to support Python. Python was chosen as programming language for coding simplicity and due to the fact that the focus of the project has been to investigate the feasibility of this specific car transportation concept under development rather than on optimizing the system.

4.3.1 Expected Output

One of the objectives of the computer vision system is to feedback the position of the car in the station to the mechanical system. This is to be able to know where and how much to move the car in order to get it in a desired position. To minimize the required computations, the requested variables were reduced to the following two:

- the angle of the car, θ
- the position of the centre point along the horizontal axis from the centre of the station, x, when viewing the car from above,

as illustrated in Figure 4.2.
The parked cars are considered aligned when x and θ are $0 \pm 1cm$ and $0 \pm 1^\circ$ respectively, as seen in Figure 4.3.
Having these two as control variables for controlling whether the car is aligned means that every car will be "perfectly" aligned in order for the system to approve the alignment. This reduces the amount of computations compared to if, e.g. a boundary box control would be implemented when the car position is approved as long as the four outer corners of the segmented car are within an allowed box.

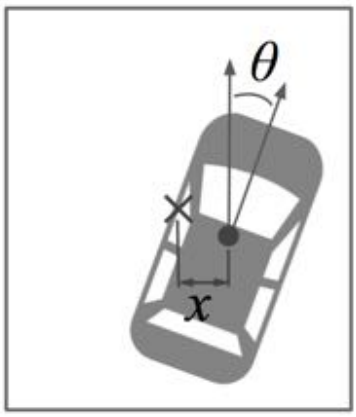


Figure 4.2: The position x in the lateral direction relative to the centre point of the station and the angle θ are the variables that are sent to the mechanical system.

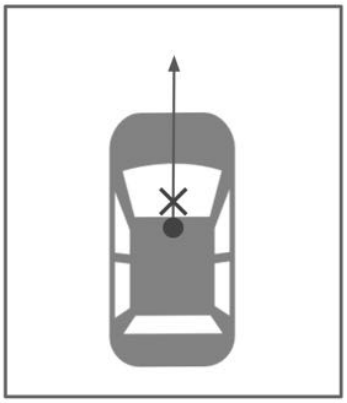


Figure 4.3: The car is aligned when the x-coordinate and θ are zero.

4.3.2 Line Control

The CVS is furthermore used for checking if the car is within the allowed lines (see Section 4.4) for driver parking guidance, with sensors for redundancy. This is described in Section 5.2.6.

4.4 Proximity Sensors

There are a total of three IR sensors implemented in this prototype, two in Station 1 and a third one in Station 3. Inside Station 1, an accurate system is required to be able to tell whether the car inside was parked correct enough by the driver. Since the belts in Station 1 cannot translate the car in the forward-

backward direction, there needs to be restrictions on where the driver can park in this particular direction.

4.4.1 Station 1: Driver's Guidance

The communication to the driver is through a driver's guidance system, shown on a screen inside Station 1. It illustrates where the driver is allowed to park, using two fixed lines on top of the video feed from the camera in Station 1. The lines represent the boundaries for the car, one at the front and one at the back, see Figure 4.4, since otherwise it would not be aligned with the belts in Station 2. The colours of the driver's guidance lines are updated to red or green corresponding to if the car has crossed or not crossed the boundary. The signals which switches the colours comes from a collaboration between both the IR sensors and the segmentation from the computer vision, further explained in the Section 5.2.6.

Figure 4.4: Illustration of the position of the two IR sensors.

4.4.2 Station 2: End sequence

The third sensor serves the purpose of knowing when the car has successfully exited Station 2 and entered Station 3, i.e. the container. This is due to the fact that the segmentation becomes more unreliable when exiting the field of view of the camera. It is placed above the passage between the two stations pointing downwards towards the floor. The sensor will there be able to detect the car when it is moving into Station 3 until the car is fully in. When the car has fully entered Station 3, the signal of seeing a car will disappear, thus notifying the system that the car has been transported into its container. The sensor also serves as a backup for situations where the car has not been successfully

transported into the container and still is in between Station 2 and Station 3 and the visual system is sending faulty data.

4.5 Communication

On a system architecture level, all communication is done using ROS. There are two types of ROS communication paradigms present in the demonstrator. These are called ROS Service and ROS Publisher and Subscribers. They communicate in different ways, which are explained below. The information packets sent by both paradigms are called ROS Messages.

4.5.1 ROS Messages

Communication in ROS is built with Messages. A message contains the data that is sent between nodes. ROS supports many predefined message types and also offers the possibility to define custom message types. This allows the user to tailor the message type for every need. Messages can be nested and consist of one or more built-in types, predefined messages or custom messages and are described in `.msg` files. The `.msg` file is a simple text file consisting of rows declaring the data of the message and its names.[19]

4.5.2 ROS Services

ROS Services create a client-server relationship between two nodes, such that the communication client node waits until a reply has been received from the server node.[20] When a service has been set up, the client can call and pass arguments to a routine on the server as easily as if the routine was located in the client's own code. The server can also return a value to the client as easily. The call can look like the following: `reply = serverRoutine(request);`, where "request" and "reply" are ROS Messages that are selected when defining the service. This definition is a description of what type of message the server expects from the client, as well as what type of message the server returns to the client. This is specified in a `.srv` file.

4.5.3 ROS Publishers and Subscribers

Nodes can also set up Publishers or Subscribers for communication. Publishers and Subscribers in ROS communicate using Topics. This is a simpler communication than services, where a node with a Publisher sends a ROS Message on a given topic. Any node with a Subscriber tuned into the same topic can read this message.[21] The user can also subscribe and publish to multiple topics via a terminal window during runtime, which helps debugging.

5. Implementation

The proposed methods and strategies were implemented using software, hardware and custom made electrical designs. The software consists of a combination of open source software and tailored software for the application.

5.1 Hardware

Hardware that was picked out to serve as IR sensors, embedded systems, motors etc. are presented in this section.

5.1.1 VL6180 IR Sensor

These sensors, the VL6180, see Figure 5.1, are proximity sensors [22], meaning they will give an output proportional to the distance to an object. The sensor works by emitting infrared light and then measuring the time of flight (ToF) it takes to reach back to the receiver in order to determine the distance to an object in front of it. This particular sensor has a very good accuracy of millimetre precision and can cover from 0 to 255 mm in front of it. [17]

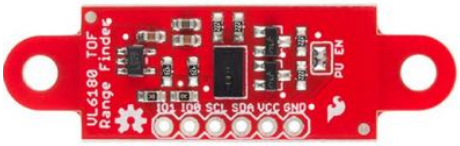


Figure 5.1: The IR sensor VL6180 from Sparkfun.

Programming the initialisation and signal handling of the sensor can be done through the Arduino IDE.

It is implemented with an Arduino Micro [23] for sending and receiving data from the VL6180, using the I2C protocol [24].

As mentioned, there are a total of three VL6180 sensors that are going to be used. In order to make it possible to use multiple sensors using only the one I2C

bus in the Arduino Micro, one has to utilize one of the I/O pins on the VL6180 as a reset pin. The reset pin will have the effect of changing the device's address upon booting, which will enable the Arduino Micro, thanks to the the I2C protocol, to talk with multiple devices. [25]

5.1.2 Arduino Micro

The Arduino Micro is a small microcontroller based on the ATmega32U4. It features 20 digital I/O pins, of which 7 can be used as PWM outputs, a 16 MHz crystal oscillator, 4 timers, I2C-bus and a micro USB connection. Powering can be done through the micro USB connection. Programming, compiling and uploading is done through the Arduino IDE.

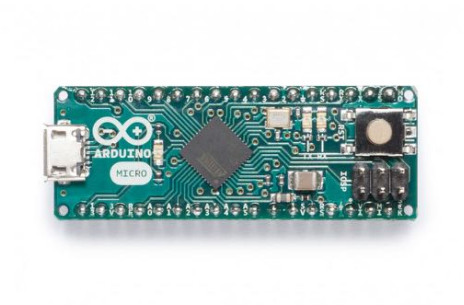


Figure 5.2: Arduino Micro

The Arduino Micro is used because of its size and relative ease of use, suitable for rapid prototyping. Also, availability of I2C was essential, which is used with the IR-Sensor. It is also applied as the MCU for the Motor Nodes, implementing velocity and position control, with reference signal provided from the Main Computer via ROS through Serial Communication.

5.1.3 Raspberry Pi 3

The Raspberry Pi 3 Model B is a small, single-board computer, running the operating system Ubuntu Mate. The computer features, among others, a 1.2 GHz 64-bit CPU, 1 GB of RAM, General Purpose IO-ports, WLAN and CSI camera port for connection with a Raspberry Pi Camera.

In this project, the Raspberry Pi is used to record video via camera stream and broadcast that video via a ROS topic. This was chosen so to relieve the host computer of the data collection process and enable focus on the image processing. Further, it gives better redundancy in a future application, where a back-up main computer would be able to step in if the main computer would fail.

Figure 5.3: Raspberry Pi 3 Model B

5.1.4 Raspberry Pi Wide Angle Camera Module

The camera that is being used for capturing images to the CVS is a Wide Angle Camera Module for Raspberry Pi, see Figure 5.4. It has a 120°horizontal field of view as opposed to the 62.2°offered by the Raspberry Pi Camera Module v2, with which the entire station would not come into view.

Figure 5.4: Raspberry Pi Wide Angle Camera Module

5.1.5 Polulu DRV8833 Dual Motor Driver

The Pololu DRV8833 is a breakout board, based on Texas Instruments DRV8833 chip. The board can supply the motors with 1.2 A continuously in its original configuration and 2.4 A when the motor outputs are paralleled. It operates in the voltage range 2.7 V to 10.8 V. [4]
The Pololu DRV8833 was chosen upon criteria regarding current and voltage for the motors, whilst being easy to use.

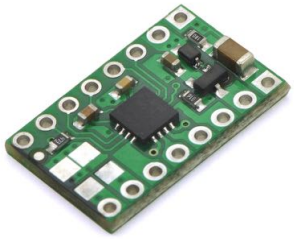


Figure 5.5: Pololu DRV8833 breakout board [4]

5.2 Software

The Car Park Loading Dock is highly dependent on software, especially for Computer Vision and Communication, described in the following sections.

5.2.1 Main Software

The main nodes are all nodes that are not peripheral nodes, i.e. Main Control, Visual System Node and the Mechanical System Node shown before in Figure 3.5 of the system architecture. These nodes are all run on the same computer, although in separate programs. This makes it easier when, in a full-scaled implementation, nodes will be distributed.

The Main Control

The Main control is implemented as a Finite State Machine, where each state corresponds to the different stages in the process of parking a car in the Car Park Loading Dock. A flowchart of the FSM can be found in Appendix A.

Some of these states require a user input before they can transition to another state, whereas others rely on the information from the visual node before transitioning. The user inputs come from the GUI described later in this section. The states of the Main Control call methods in the helper classes Mechanical Control and Visual Control which govern the communication with the their respective nodes.

The main nodes contain sets of states and remain in an idle state until they receive a message from another node. These messages contain an integer corresponding to which state they should transition to. The routines in these states are run before the node returns a message to the caller and transitions back to the idle state.

The Visual System Node

The Main Control sends a message corresponding to the correct state in the Visual System Node when image processing or sensor reading is required. This means that the sensor values from the sensor nodes are handled by the Visual System Node. The Visual System Node also samples the cameras' video feed as soon as they are received from the camera nodes. It is, however, only when the visual node enters one of its non-idle states that image processing begins. When the visual node has received an integer specifying a state from the Main Control and carried out the routines in that given state, it returns a boolean value. This value states if the outcome of the state was True or False. The Visual System Node has the following routines that can be called from the Main Control.

- *updateBackground* - updates the reference images used by the image processing.
- *isInsideBoundary* - evaluates if the car is located within the allowed boundaries.
- *carInFOV* - evaluates if the car has started to enter the field of view of the camera.

Another routine in the Visual System Node can be called from the Mechanical System Node, called *getGeometrics*. This routine returns the pose of the car, expressed in it's 2D-coordinates and it's angular orientation.

The Mechanical System Node

The Main Control sends a message corresponding to the correct state in the Mechanical System Node when an actuation is required in the system. This node is connected to all of the actuation nodes and has the information for how the motors should move for each stage of the parking process. The routines that can be called from the Main Control are as follows.

- *alignStation* - responsible for controlling the motors that drive the lateral belts in stations 1 and 2. Contains the code for calculating the control signals for the closed loop controller based on the car's pose. Calls *getGeometrics* on the Visual Node to receive the measurement of the car's pose.
- *transferCar* - runs the belts in station 1 to move the car between stations 1 and 2, in parallel to calling *alignStation* for station 2 to ensure it is aligned when the transfer is completed.
- *liftPlate* - runs the lifting mechanism that lifts the side-belts in station 2.
- *transferPlate* - drives the side-belts in station 2 and 3 to convey the plate and car into the box in station 3.

The Mechanical System Node also returns a boolean value to the Main Control when a routine has been completed, however this value is not used but is a way to feedback the success of an actuation from a motor node in a future implementation.

5.2.2 Graphical User Interface

In order to simplify the use of the system, both in testing and deployment, a Graphical User Interface (GUI) was created. The GUI was written in Python, using rospy, a client library for ROS to enable applications written in Python, and the UI library GTK+, containing predefined UI elements. GTK+ is XML-based, where the component composition is described in the XML file. To build the GUI, the WYSIWYG editor Glade Interface Builder was used, which generates this XML file and thereby gives a swift development process. Another alternative would be to code the XML file separately.

Figure 5.6: The Graphical User Interface

The GUI, seen in Figure 5.6 uses ROS subscribers and publishers to monitor states and send commands. In the application, each button can change the State Controller, which in turn sends commands to the appropriate nodes.

5.2.3 Communication

In order to mimic the distributed system, communication between the nodes must be established. The communication support of ROS described in section 4.5 was used for this. The communication between the Main Control, the Visual System Node and the Mechanical System Node is done using ROS Services. Services are well suited for this task as they enable two-way communication in the form of request/reply interaction. On the other hand, the communication

between the Mechanical and Visual System nodes and the peripheral nodes, like the actuator nodes, uses Publishers and Subscribers.

Main Nodes Communication

The main nodes are what we call the Main Control, the Visual System Node and the Mechanical System Node depicted in Figure 3.5 of the system architecture. Main Control has two services connected to the other two main nodes. These are called *VisualRequest* and *MechanicalRequest*, and govern the operation of the two other main nodes. The *VisualRequest* service is also set up between the Mechanical System Node and the Visual System Node to directly exchange information.

A VisualRequest call is described by the following `.srv` file.

```
uint8 State
---
VisualResponse Reply
```

It sends an unsigned integer to the Visual System Node and returns a custom message type, *VisualResponse*. It was defined taking into account that the Visual System Node needs to reply with a boolean value as well as the pose of the car. Because of this, a *VisualResponse* message contains the predefined messages Bool and Pose2D. The *VisualResponse* `.msg` file reads as follows:

```
std_msgs/Bool Reply
geometry_msgs/Pose2D
```

This message is nested, where in turn the `Pose2D.msg` file contains the following:

```
float64 x
float64 y
float64 theta
```

which contains three floating point values describing the car's x- and y-coordinates as well as it's angular orientation.

The *MechanicalRequest* call's `.srv` file contains

```
uint8 State
---
std_msgs/Bool Reply
```

where we define that the Mechanical System Node simply receives an integer and replies with a boolean.

Peripheral Nodes Communication

The actuation nodes together subscribe to 7 different topics. These topics are published to by the Mechanical System Node, where it defines a velocity or position that the actuation nodes should achieve. This communication is one-way and the Mechanical Node takes no regard to what the actuation nodes do with this information.

The Computer Vision Nodes publish compressed images going to the Visual System Node via two own topics. The Visual Node subscribes to these topics to update the view of the cameras.

The sensor node has three sensors connected to it. The node reads the sensors' values, processes them and publishes the result to a topic which the Visual System Node subscribes to.

5.2.4 Controller Software

There are three different types of controllers implemented in the demonstrator. Two of them are located on the motor nodes, where one is a velocity controller and the other is a position controller. The third is the controller regulating the car's position and orientation in stations 1 and 2. All controllers are closed loop error feedback controllers.

Car Pose Controller

The input required to regulate the car's pose is its angular orientation and lateral position x. These values come from the image processing in the visual node. A P-controller on the Mechanical System Node calculates the control signals based on the error compared to the reference values. The control signals are sent to the appropriate actuating nodes which use their own controllers, explained below, to interpret these signals. The interaction between nodes for this closed loop regulation is described in Figure 5.7. The reference position and reference angle are both zero, i.e. we want the car centred in the station and oriented straight forward. Due to uncertainty in the measurements, a threshold of $\pm$ 1 degree and $\pm$ 1 cm is used, so that the regulation of each movement stops when its measurement is inside its interval.

The control signals for correcting the cars position and orientation are calculated separately and super-positioned to allow for simultaneous rotation and translation of the car. The super-positioning is done differently for the control signals to the front and rear belts according to

$$u_{front} = u_{rot} - u_{trans} \tag{5.1}$$

$$u_{rear} = u_{rot} + u_{trans} \tag{5.2}$$

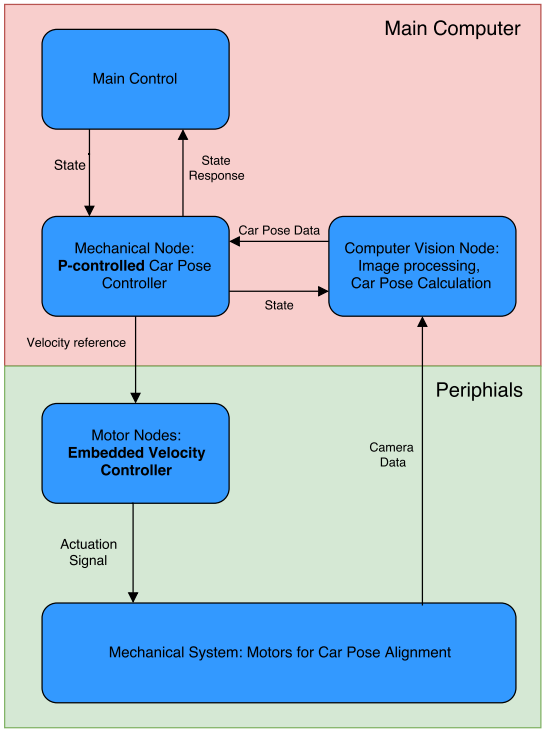


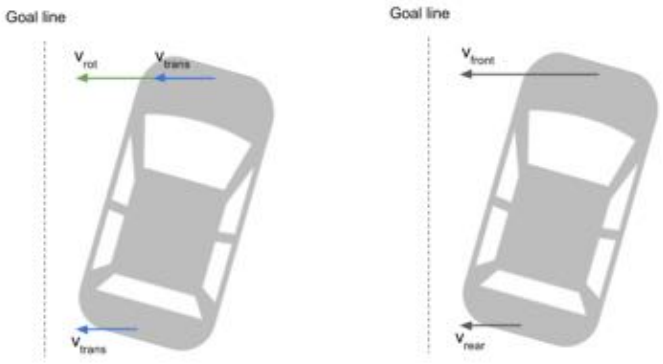
Figure 5.7: Interactions between controllers

where u denotes a control signal, *rot* relates to the rotational control and *trans* to the translational control. With these control signals and equation 5.3 describing the angular rotation ω, the situation depicted in Figures 5.8a and 5.8b is achieved. The difference between the two velocities converts into rotation, while the equal parts only translate the car.

$$\omega = \frac{v_{rear} - v_{front}}{l} \tag{5.3}$$

l here is the distance between the contact points with the belts, i.e. the wheelbase of the car.

The pose measurement of the car is updated before every actuation occurs. With the current setup, a frequency of new actuation commands of 20 Hz is reached.



(a) Velocities of the front and rear separated

(b) Velocities of the front and rear super-positioned

Figure 5.8: Velocity vectors for alignment

Motor Speed Controller

The motors are equipped with quadratic rotary encoders. The encoder outputs square waves on two channels, which triggers interrupts on the microcontroller and the position of the rotor is determined by studying the relationship between the interrupts.

There are two methods in which the motors are controlled in this application, either by position control (encoder ticks) or velocity control (encoder ticks per second). Both of these have been implemented on the Arduino Micros as PI-Controllers.

A PI-Controller implements the following equation:

$$u(t) = K_P \cdot (r(t) - y(t)) + K_I \cdot \int_0^t r(\tau) - y(\tau) d\tau$$

where $u(t)$ is the output signal, $r(t)$ the reference signal provided from the Car Pose Controller and $y(t)$ the systems output signal at time t. Parameters K_P and K_I are so called design parameters, chosen to regulate the system as well as possible. $r(t) - y(t)$ is thereby the error of the system at time t.

When controlling the velocity, there is need to calculate the velocity from the encoder ticks with the following equation:

$$w = \frac{\frac{EncoderPulses}{PulsesPerRevolution} \times 60}{SamplingTime}$$

where w will be the velocity in rpm and Encoder Pulses are the number of pulses during the Sampling Time. The sampling time is set with the MCUs internal timer, which in this case is 1000 Hz.

5.2.5 Distance Sensor Software

The software for handling the VL6180 IR sensors [17] is written in the Arduino IDE [26], since the sensors are programmable with the Arduino language using a package found on their hook-up guide [17] made specifically for the usage of the Arduino hardware. Implementing together with ROS, the outputs of the sensors could be transmitted to the main computing node for the driver's guidance.

The fact that in this project, one Arduino Micro will manage all three sensors means that there need to be a couple of lines in the code handling the address changing at each of the sensor's initialisation. This was done according to a tutorial showing the techniques required for using multiple VL6180 IR sensor on a single Arduino [25]. It could be done by disabling all but one sensor, i.e. drop the power flow to it, and at the same time distributing an address to that sensor. After one of them has been given an address, another sensor can be enabled and the same process repeats for all sensors, and after that, the initialisation is done.

The data from each sensor was chosen to be translated from a distance from 0-255 mm to instead a boolean 1 or 0. This was because of the chosen application, which is: the lines in Station 1 and the telling of when the car is inside Station 3 in the end of the Station 2 sequence. The lines (shown to the driver in a GUI on a screen) only needed to be either red or green, so a 0 from the sensor meant that the sensor value was below a certain threshold (chosen specifically to suite this prototype) and a 1 meant that the threshold has been met, i.e. a car is in front it. The driver will be told through the driver guidance's interface to stay inside the vehicle and try to park inside the boundaries until the car has been parked accordingly.

The same goes for the third sensor. The third and last sensor is pointing downwards towards the floor between Station 2 and 3. The threshold has been calibrated to send the information if a car is below it or not. It therefore serves the purpose of knowing when the transition of the car has been successful or not, and when to close the container for transport into the grid.

5.2.6 Computer Vision System

To be able to detect the position of a car in the station, a car segmentation method is required and the proposed idea is to use background subtraction. This means that an image with the car present is subtracted with an image of the car non-present to get the foreground, i.e. the car. Background subtraction is considered a suitable method for this application since the only new objects in the parking bay are cars ready to be parked. Therefore, pixels that were not in the background will be considered as belonging to the car.

Two specific algorithms for background subtraction are used and compared in this report:

- Algorithm based on subtracting the current image with a reference image, see Image Subtraction.
- Algorithm based on the built-in method for background subtraction in OpenCV, see `BackgroundSubtractorMOG2`.

These have the same pre- and post-segmentation processing procedures, but differ in the actual foreground extraction method.

Pre-Processing

The current frame is polled from the camera via a designated thread. It returns the frame that is most recently captured, which means that the frequency at which the Raspberry sends images over ROS is not compromised. The downside is that if the frame rate is set too high, the same frame will be polled several times and consequently result in a choppy slow-motion effect. So with the frames being sampled at a grey scale resolution of 640x480 pixels the sending frequency of the image publisher is set to 12 fps. To reduce the amount of data that is sent over LAN the images are further compressed to JPEG format. Since a wide angle lens is used, the captured image will be distorted and is needed to be corrected, see Figure 5.9 and 5.10. Before one of the algorithm takes over the frame is blurred with the help of a Gaussian smoothing to reduce high frequency noise, see Figure 5.11.

Figure 5.9: Pinchushion distorted image.

Figure 5.10: Undistorted image.

Figure 5.11: Gaussian smoothed image to reduce high frequency noise.

Foreground Extraction Method 1: Image Subtraction

The car segmentation is initialized by subtracting the blurred frame with the reference image, see Figure 5.12. The small intensity differences are masked with a specified threshold and a Gaussian adaptive threshold algorithm is applied to further reduce noisy colour cues, see Figures 5.13 and 5.14. The intelligence of this extraction method lies within the adaptive threshold algorithm since it's able to mask the colour differences locally in the image and set corresponding threshold values dynamically.

Figure 5.12: The difference between the blurred frame and the blurred reference image

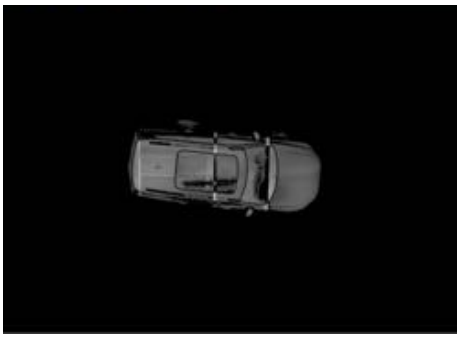


Figure 5.13: The segmentation is masked with a global threshold to reduce noise

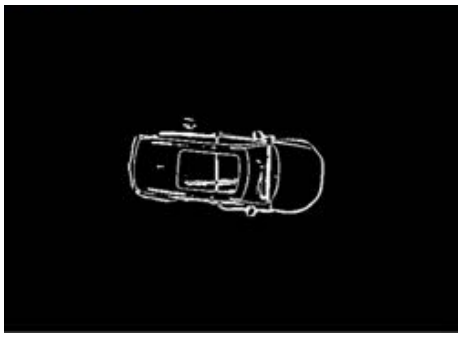


Figure 5.14: Adaptive threshold to intelligently extract what is the car and not

Foreground Extraction Method 2: `BackgroundSubtractorMOG2`

`BackgroundSubtractor` is a background/foreground segmentation algorithm class in OpenCV that copes with excluding the shadow pixels from the extracted foreground and in this report, the specific method `BackgroundSubtractorMOG2` is used.

`BackgroundSubtractorMOG2` is a Gaussian mixture based algorithm and it operates by assigning every pixel in the image to a number of segmentation clusters with a certain probability and then interpreting the clusters that are most apparent as foreground. What is special with `BackgroundSubtractorMOG2` is that it is able to dynamically set the number of clusters, or Gaussian distributions, for every pixel which makes it more robust against illumination. Shadows are detected by examining the brightness of a pixel classified as foreground with the reference image. The result can be seen in Figure 5.15 where the extracted foreground is displayed as white, the background black and the shadows as grey.

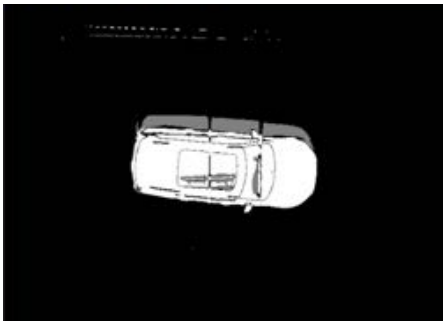


Figure 5.15: The extracted foreground (white) and the detected shadows (grey) by using `BackgroundSubtractorMOG2`.

The shadows are removed by masking the grey pixels.

Post Image Processing

When the foreground is extracted, the resulting frame is filtered by removing contours that are smaller than a predefined number of pixels to eliminate the noise in the image. The results after the filtering operations can be seen in Figure 5.16 and Figure 5.17.

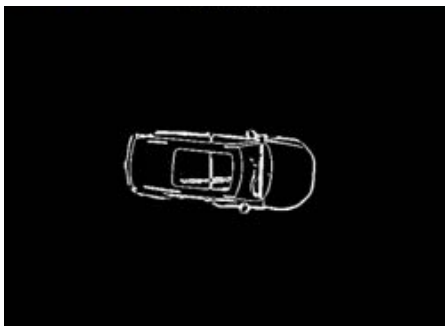


Figure 5.16: Removing noise on segmented image from the image subtraction algorithm.

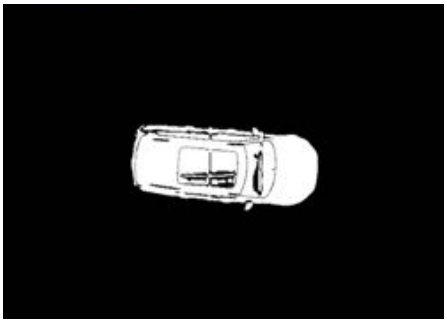


Figure 5.17: Removing noise on segmented image from the `BackgroundSubtractorMOG2` algorithm.

The remaining foreground is dilated to fill the emptiness between the contours and make them merge together into one, as illustrated in Figure 5.18 and Figure 5.19.

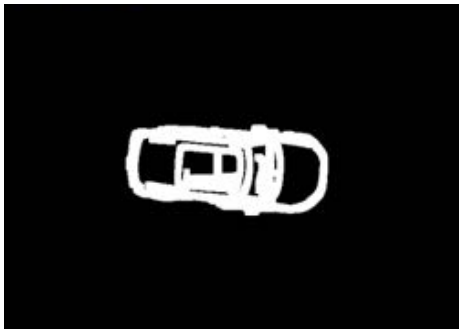


Figure 5.18: Resulting dilation on image using the image subtraction segmentation method.

Figure 5.19: Resulting dilation on image using `BackgroundSubtractorMOG2` segmentation.

When the segmentation is done, the resulting foreground is applied into the original frame. The resulting segmentation with the image subtraction method is displayed in Figure 5.20 and the segmentation with `BackgroundSubtractorMOG2` is shown in Figure 5.21.

Figure 5.20: Resulting segmentation with the image subtraction method

Figure 5.21: Resulting segmentation with `BackgroundSubtractorMOG2`

Calculating Position and Angle from Segmentation

The contour that is given from the segmentation represents the detected car in theory. OpenCV has several functions for fitting geometric Figures onto contours from which information such as area and corner coordinates can be extracted. One of them is *minAreaRect* which finds a rotated rectangle of the minimum area enclosing the input contour. Finding the minimum rectangle was tried out in an early stage of the project, but was however excluded as a solution. This was due to the fact that the bounding rectangle was too sensitive to the shape of the contour and the smallest error in the segmentation affected the resulting position and angle substantially. In other words, a perfect segmentation would be essential in order to get accurate values.

A method proven to generate more accurate results is *fitEllipse*, as can be seen in Figure 5.22, which fits an ellipse around a set of input contours.

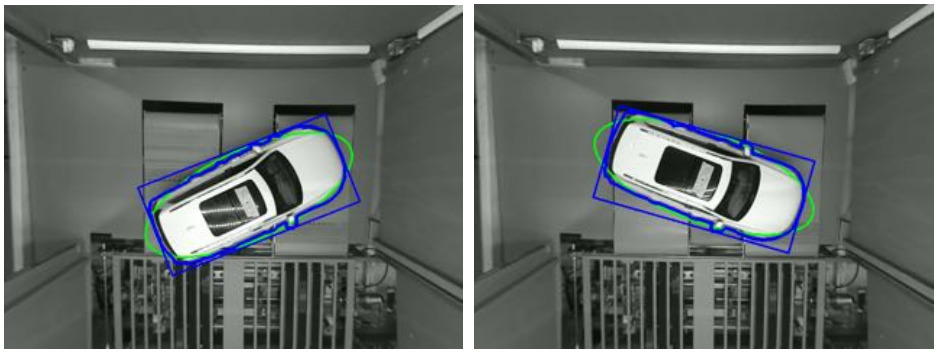


Figure 5.22: Comparison between *fitEllipse* (green) and *minAreaRect* (blue) for contour enfoldment.

Driver Guidance Lines

The driver guidance system explained in Section 4.4 is additionally getting input from the CVS for redundancy. The system uses the pixels from the segmented car image to analyse whether there are white pixels, i.e. pixels belonging to the car, outside the allowed zones. If so, the line corresponding to the affected zone shifts from green to red. If not, the two lines remain green.

5.3 Electrical Design

For this project, two printed circuit boards (PCB) were designed; one for the motor control nodes and one for the IR sensors. The designs were made using the program EAGLE PCB design software [27].

5.3.1 PCB for IR Sensors

The PCB for the IR sensors, seen below in Figure 5.23, was specifically designed for the reset pin on the VL6180, see [17] for reference. The reset pin needs 2.8V which led to the design of a simple voltage regulator using two resistors, R1 and R2 shown in Figure 5.23. The other components are pin mounts for all the other signals and supply.

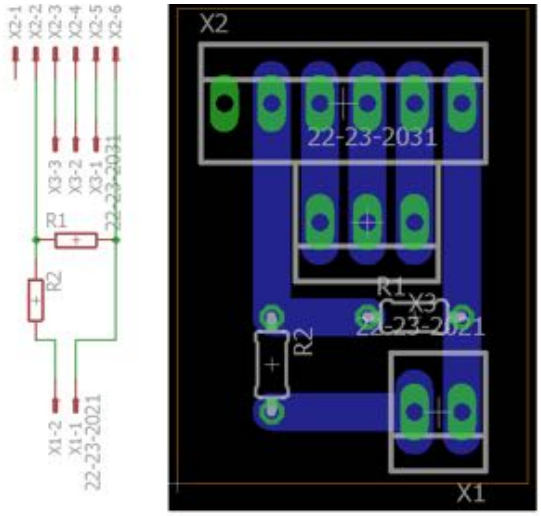


Figure 5.23: Eagle schematic and board of the PCB for the VL6180

5.3.2 PCB for Motor Nodes

To ease the assembly, a PCB for the Motor Nodes were created. It hosts connectors for one Arduino Micro, one Pololu DRV8833 motor driver, pins for motor

input channels and encoder related wiring and input voltage pins, see Figure 5.23 for reference.

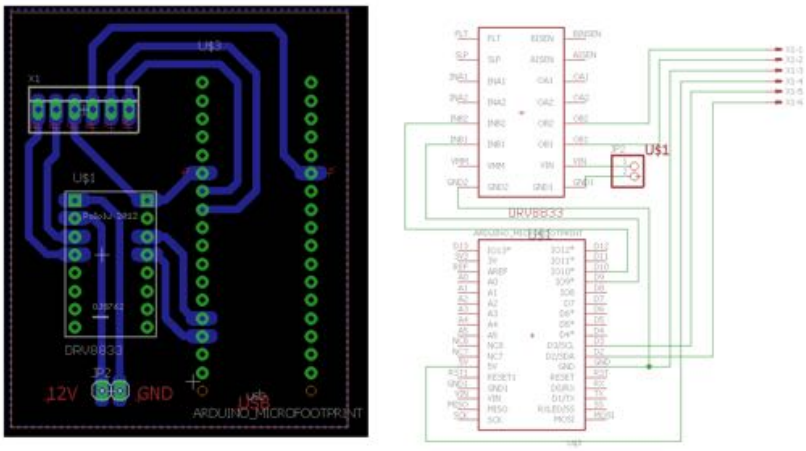


Figure 5.24: Eagle schematic and board of the PCB for the motor node

The driver is controlled by the Arduino with a PWM signal connected to the input channels of the driver, which in turn enables the output ports to power the motors. The Arduino reads the encoder and sends a control signal according to its controller algorithm.

6. Verification and Validation

To verify that the project meets the set requirements that are defined in section 1.3, a set of tests were conducted for the **strict** requirements which are accounted for in the table 6.1.

Table 6.1: Test Matrix

Test ID	Procedure	Req. Tested	Expected Result
1	Car is placed into station 1 and the CVS computes the car's pose and presents it on a monitor.	1a	The car's position appears to be sufficiently accurate through visual observation.
2	Car is placed in station 1 and a complete cycle is performed.	1b, 1d	No damage inflicted on the car.
3	Car is remotely driven into station 1	1c	A video stream is displayed showing a boundary for the car to be within.

For the functional requirements, which are hard to conduct testing on, the verification constitutes of the projects stakeholder deem if the requirements were met or not.
Throughout the project refinements and its development, the stakeholder has been closely involved and consulted when design and function decision has been made. Finally the stakeholder was invited to a demo session to review the design and assembly.

7. Results

7.1 Test 1:

Two tests will be made for the computer vision, one for measuring the accuracy of the angle and position and one for comparing the two algorithms. The accuracy of the computed angle and position of the car is estimated with the help of a protractor and a ruler. 5 test cases for each will be made and the tests are designed for as good segmentations as possible with the intention to see how well the ellipse is able to approximate the cars coordinates. The OpenCV function for displaying images supports zooming and based on where the cursor is placed in the image it gives the pixel coordinates, so by placing the ruler straight up relative to the bottom of the frame, see Figure 7.1, and extracting the pixel y-coordinate that corresponds to the 60 cm marker, the conversion from pixel- to cm-coordinates becomes as seen in equation 7.1

$$x_{calc} = y_{coord} \frac{60}{(480 - 78)} \tag{7.1}$$

The actual position of the car is approximated as the mean of the cars side-to-side coordinates with respect to the ruler, see equation 7.2 and Figure 7.1. The angle however is approximated by placing the car on top of a protractor at 20 degrees, see Figure 7.2. The result can be seen in Table 7.1 for 5 consecutive frames.

$$x_{meas} = \frac{x_{bot} + x_{top}}{2} \tag{7.2}$$

The two algorithms image subtraction and backgroundsubtractor are compared in Figure 7.3 where the the extracted contour is shown together with the foreground segmentation before the post processing stage.

Together with the results from Table 7.1 and the visual observations shown in Figures 7.1, 7.2 and 7.3, the results match the expected outcome.

Table 7.1: Accuracy test for the computer vision system

x_{meas} [cm]	$x_{calc}[cm]$	$\theta_{meas}[cm]$	$\theta_{calc}[cm]$
40	39.586	-10	-8.254
	39.576		-8.280
	39.571		-8.253
	39.587		-8.264
	39.574		-8.264
avg. accuracy 0.419 [cm]		avg. accuracy 1.737 °	
40.75	40.024	0	0.089
	40.017		0.089
	40.017		-0.205
	40.020		0.159
	40.020		0.002
avg. accuracy 0.7304 [cm]		avg. accuracy 0.0268 °	
36.45	34.752	10	9.239
	34.692		9.344
	34.734		9.344
	34.734		9.284
	34.747		9.435
avg. accuracy 1.716 [cm]		avg. accuracy 0.671 °	
45.55	45.869	20	19.549
	45.876		19.543
	45.868		19.530
	45.849		19.530
	45.849		19.460
avg. accuracy 0.319 [cm]		avg. accuracy 0.478 °	

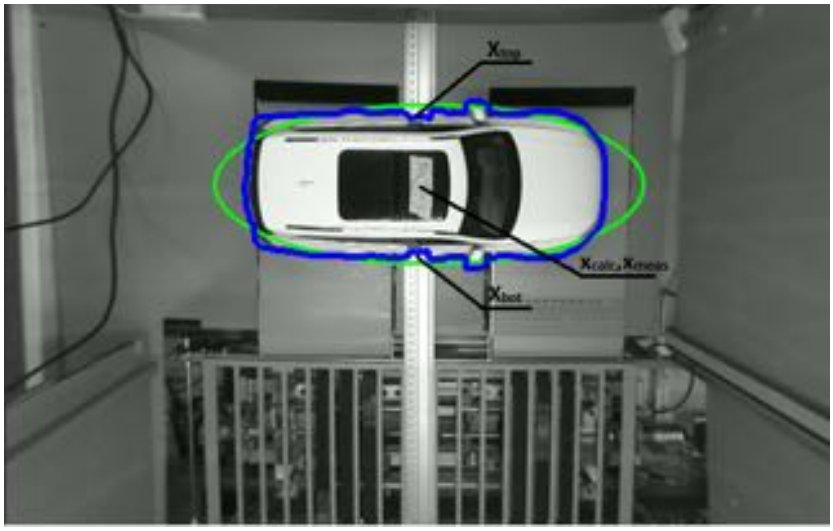


Figure 7.1: Testing the accuracy of the computer vision system for the cars position.

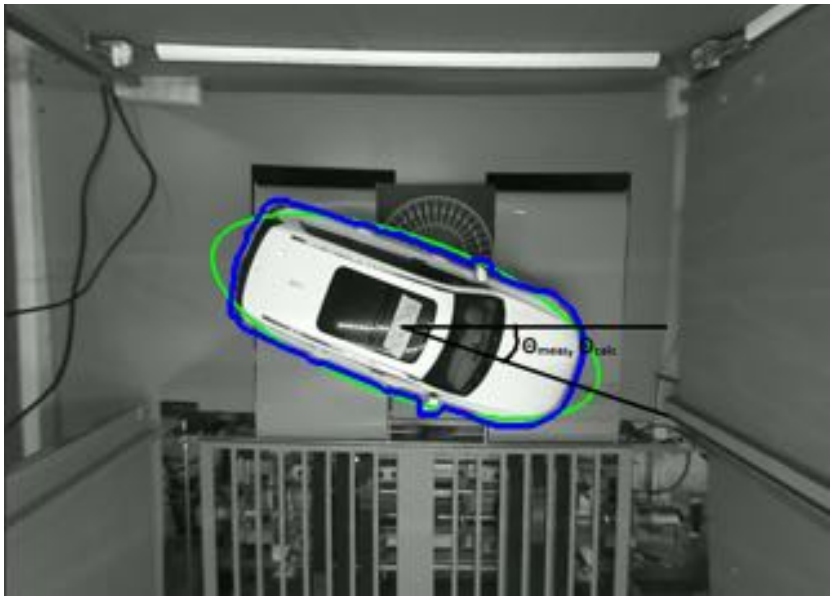


Figure 7.2: Testing the accuracy of the computer vision system for the cars angular orientation.

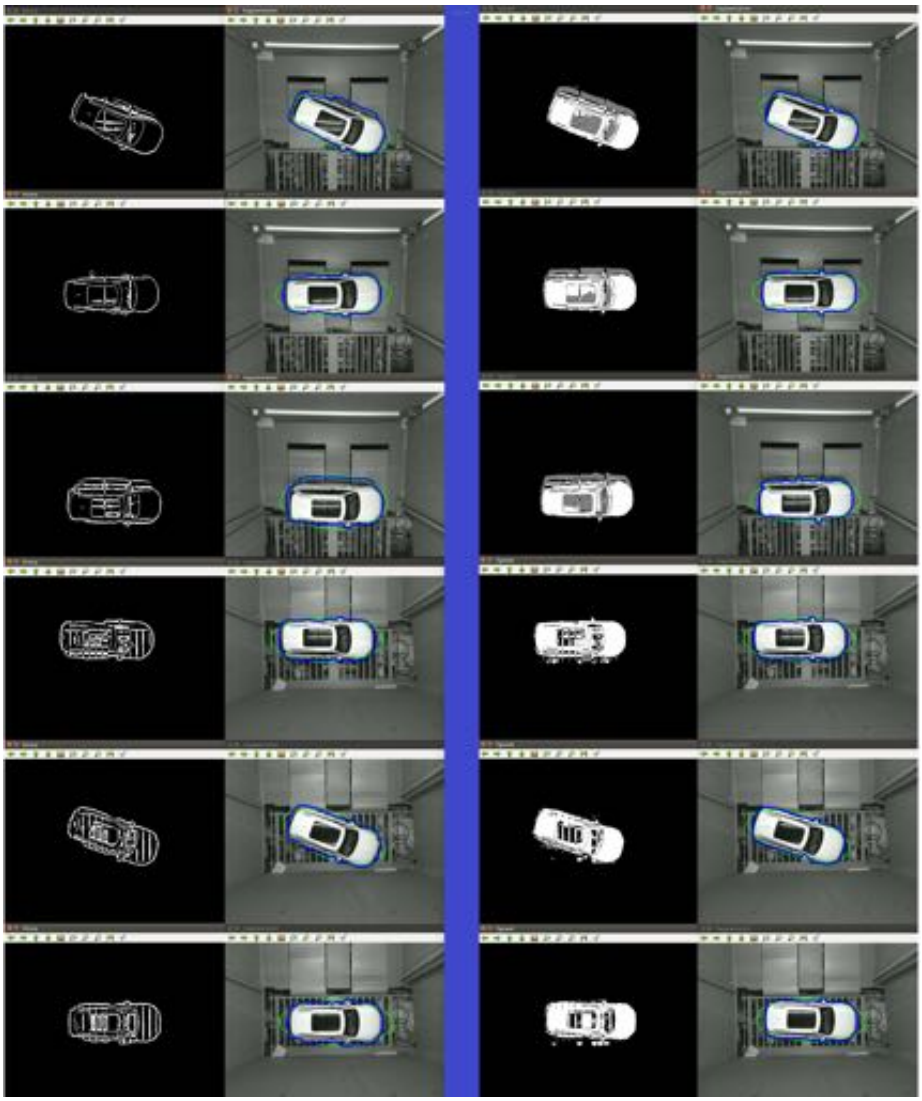


Figure 7.3: Comparison between backgroundSubtractor (right) and image subtraction (left).

7.2 Test 2:

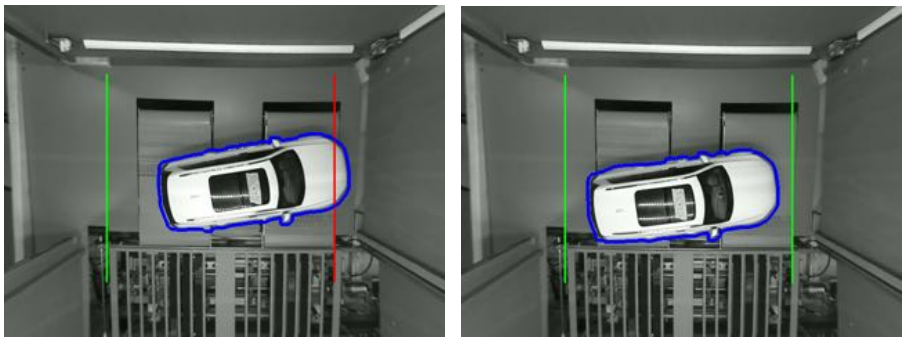
10 test runs were conducted and the result of these are displayed in Table 7.2. The expected outcome was reached 8 out of 10 times.

Table 7.2: Test Results from Test 2

Test run	**Outcome**	**Cause of failure**
1	No damage to car	-
2	No damage to car	-
3	Car's side driven into the wall by the system	Bad segmentation causing controller to send wrong signal
4	No damage to car	-
5	No damage to car	-
6	No damage to car	-
7	No damage to car	-
8	Car's front right corner driven into the wall by the system	Shaft Coupling between motor and pulley axle damaged, causing axle not to rotate
9	No damage to car	-
10	No damage to car	-

7.3 Test 3:

In Figures 7.4a and 7.4b it is illustrated what the feedback to the driver looks like. It shows two lines which colour will toggle between red and green depending on whether the car is between the lines or not, thus only giving the driver feedback to drive forward or backward.



(a) Unsatisfactory parking **(b)** Satisfactory parking

Figure 7.4: Test 3 results.

8. Discussions and Conclusions

In this chapter, the results are discussed and conclusions from the project are drawn.

8.1 Discussion

The estimated accuracy of the computer vision system, seen in table 7.1, is exposed to a couple sources of errors. The angle of the car was measured by placing it on top of a protractor with its long axis aligned with the 20 to 200 deg line, see figure 7.2, which might not represent the true angle since the side of the car isn't completely straight. And the protractor was only relatively aligned to the pi camera. As for measuring the position the same problem arose when aligning the ruler with the camera and the conversion between centimeter and pixels suffered from that as well. The tests were made to give a general idea of how well the system operates, but the results should be taken with caution. Something interesting to notice is that the calculated coordinates from the computer vision system lack reliability below the order of magnitude 100 μm.

From the comparison between the two segmentation algorithms, seen in figure 7.3, both methods struggle with shadows even though the `Backgroundsubtractor` have an incorporated intelligence to reduce the effect of shadows. Furthermore the foreground outputted from the algorithms differs a lot. `Backgroundsubtractor` gives a more finely tuned segmentation while the image subtraction method is less sensitive to small changes in the environment.

When running the whole system cycle, the expected outcome was reached 8 out of 10 times. As seen in table 7.2, failure occurred due to the computer vision system not being able to correctly segment the car, causing the controller to send a bad actuation signal and because of mechanical failures caused by a damaged shaft coupling.

The requirement 1b is therefore not met. The sources of error comes from the controllers being unable to handle bad segmentations that emerge promptly, which would be a logic error in the software. Furthermore, not being able to

identify mechanical faults comes from lack of sensors, for example sensors for measuring sudden changes in motor voltage and current.

8.2 Conclusion

A 1:10 scale of a Car Park Loading Dock was designed, developed and implemented, so to enable easy transition from the human's parking to insertion into the project owner's larger parking solution. The implemented prototype uses a combination of computer vision and mechanical and electrical parts to form a highly complex mechatronic product.

The prototype performs several tasks, firstly aiding the driver in parking the car, then transferring and doing precision alignment of the vehicle and finally moving the car into a container, which then is handed to the larger parking system.

There are several limitations to the prototype as it is implemented today. In general, safety and error handling is lacking. This is in large because of the vastness of the project, with many parts working together in different situations. Developing error handling for each state the system may be in was therefore not feasible within the time frame.

In the following chapter, future development of the prototype is discussed.

9. Future Recommendations

9.1 Sensors

When scaling the prototype up to the real size of the stations, a number of issues occur, and one of them is the choice of sensors. The sensor incorporated in the demonstrator of this report, the VL6180, can only reach up to approximately 255 mm. The real size would instead require both a range of about 2-3 m to be sure to reach a small/thin car and a centimeter accuracy to not detect falsely.

One of the positive aspects of using a proximity sensor, such as the VL6180, that was not implemented in this prototype, is the ability to not only send a boolean signal of seeing a car or not, but instead sending the continuous distance to the car. With the distance to e.g. the upper boundary shown on the user interface, it could give the driver a more clear idea of how off they are, and thus a better initial positioning of the car.

9.2 Belts

In the demonstrator the material of the belts is rubber. In a future work, studies need to be done if rubber is the best solution. When the hand-brake is active and the car is placed on the belts there will be a high friction coefficient between the belts and the wheels of the car while aligning. Since the car is aligned by moving the belts the wheels will slide across the belts and not roll. The friction will highly affect how often changing of the belts needs to be done. Another problem is that in the north of Europe the cars also have stubbed tires, this will affect the abrasion of the belts and also be abrasion of the stubbed tires.

Another critical part is the distance between the belt in station 1 and 2. This distance is a problem since the wheels can easily get stuck between the different stations. The distance can be reduced by minimising the width of the plate and therefore make a smoother transportation of the car. The most common width of the wheels is between 155 mm and 300 mm and the most critical is the 155 mm tires. In a future work calculations need to be done at the plate in order to keep a high strength but with the smallest possible width.

9.3 Computer Vision

The two different algorithms that were implemented were tuned for an RC-car that has a specific shape and color and the robustness of the segmentation methods against different car models hasn't been proven. One thing that is certain though is that based on what car that is being parked, the segmentation will excel at a specific set of parameter values. A suggestion is therefore to find a way of initializing the parameters dynamically. As an example the color of the car could be identified with the help of histograms and consequently used as a basis for what parameter values to choose. Another apparent issue are shadows that become misinterpreted as a part of the segmentation, which could partly be worked around with better lighting conditions. At the scale of the demonstrator there is little to no room to design a lighting setup that lights up the stations and simultaneously eliminates shadows. Furthermore there is one piece of information that could greatly improve the segmentation that hasn't been taken into account for and that is that the car doesn't deform. So instead of initializing a new segmentation for every frame one could just update the position of a finely tuned segmentation right after the car has parked. A further iteration could take inspiration from the state of the art "Integrating tracking with fine object segmentation" report mentioned in chapter XXX and perhaps use one of the two suggested methods in this report as a part of the object tracking fusion strategy.

We also recommend future investigators for the full-scale model to have a more powerful computer doing the image processing. This relates to the resolution of the camera image. While the Raspberry Pi Camera Module v2 is capable of capturing 3280×2464 pixel images, one can infer that a camera with higher resolution and speed could greatly improve the implementation. The dots-per-inch increase certainly would be necessary to capture the details at a farther distance. This would then require a more powerful computer to process the images as the amount of pixels increases quadratically.

9.4 Control Software

The controllers used in the demonstrator are standard P- and PI-controllers. They regulate well in the scaled down version, handling the size and weight of the car with good accuracy. How the controllers will handle the real size and weight of different car should be further investigated.

The P-controller for the Car Pose Control, that is used in conjunction with the computer vision system, could be further developed. Too small control signals to the motors sometimes result in an actuation that doesn't outdo the friction, whereas a I-part in the controller would tackle the static error being caused by this. Another priority should be to implement handling of sudden faults in the segmentation of the car.

The PI-controllers in the motor controllers are quite robust, however implemen-

tation of anti-windup should be a priority since this could be safety critical to ensure that the car is not damaged.

9.5 Hardware selection

The mechanical designs and solutions in the 1:10 prototype are heavily subjected to various restrictions. Some mechanical components that we wanted to use were not available to buy in this scale. This mainly applies to off-the-shelf conveyor belts. Ocado is greatly encouraged to buy or special-order complete conveyor solutions. There are some suppliers that can custom-build tiny conveyor belts complete with tensioning mechanisms and drives. However, these would come with a big price tag which brings us to the next restriction; money.

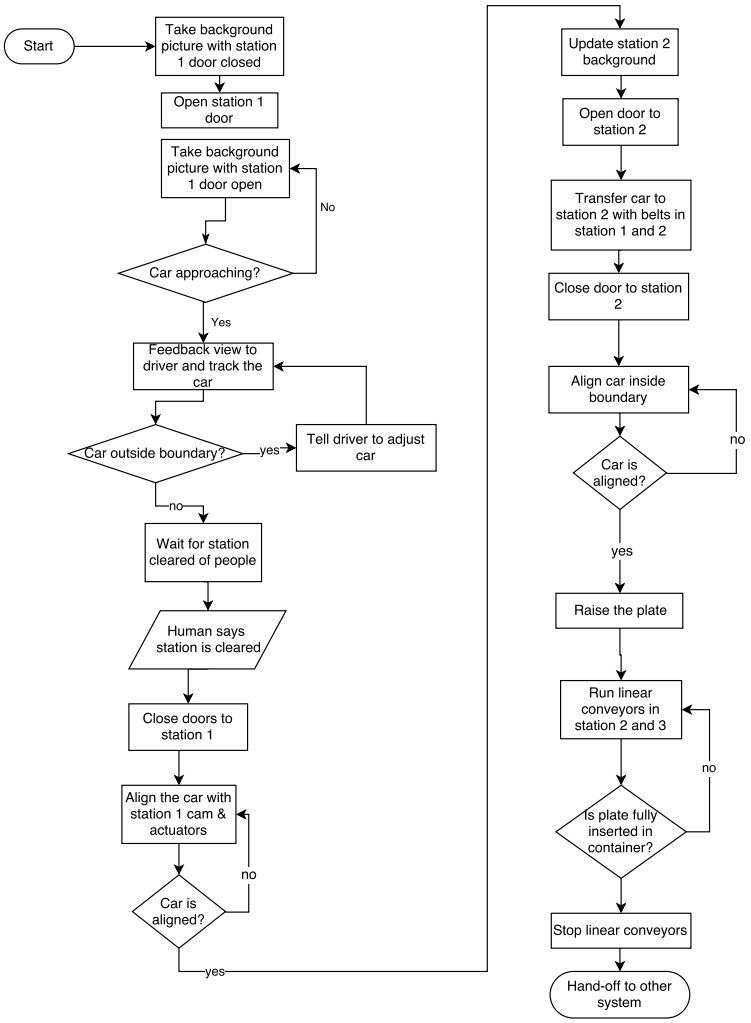
The budget of the entire project was limited to 50 000 SEK. We were reluctant to spend a lot of money on custom-made parts which we beforehand couldn't conclude would fit well in the rest of the system. If we would have special-ordered parts that in the end did not fit, both time and money would have been spent on useless components.

Time is the next restriction. As the number of iterations made on the prototype grows, so does the outcome on the final product. Though a few crude prototypes were made early in the project to validate that the concept would work, we are convinced that yet another iteration after this prototype would have been even better in regard to design and implementation. But there simply is not time.

9.6 Reversed workflow

The prototype is only implemented to be able to park a car in the Storage System, and can not return the car from the system to the driver at the time of pick-up. However, one should only need to do the steps of the prototype in reverse order to get the sought after functionality. This should be investigated and tested to assure it meets requirements for returning the car at pick-up time.

A. Main Control Flowchart



A. Bibliography

[1] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," September 2005.

[2] S. Wang, H. Lu, F. Yang, and M.-H. Yang, "Superpixel tracking," 2011.

[3] G. Limited. https://www.youtube.com/watch?v=ugHZ8CQAnIY, January 2016.

[4] "Pololu drv8833," 05 2017.

[5] "About ocado." http://www.ocadogroup.com/who-we-are/ocado-at-a-glance.aspx, December 2017.

[6] B. Ovrum, "The parking parking dilemma: Innovative solutions for parking and parking requirements." https://web.archive.org/web/20120601213622/http://ladbs.org/LADBSWeb/LADBS_Forms/Publications/BudOvromRoboticParking.pdf, October 2011.

[7] ParkPlus, "Palletless comb exchange transfer system." http://parkplusinc.com/products//automated/, 2017.

[8] M. Köster, "Press release: Lödige industries supplies europe's largest automated car park in aarhus, denmark." http://www.lodige.com/News-and-press-01.html?action=viewNews&news_id=160&element_id=2551&category_id=3, November 2015.

[9] W. P. Solutions, "Leifsgade." https://www.westfaliaparking.com/projects/leifsgade/, 2017.

[10] ParkPlus, "Self charged robotic vehicle transfer system." http://parkplusinc.com/products//automated/, 2017.

[11] Habasit, "Fabric conveyor belts engineering guide."

[12] S. Brahmbhatt, *Practical OpenCV*. University of Pennsylvania: Springer Science, 2013.

[13] K. E. Papoutsakis and A. A. Argyros, "Integrating tracking with fine object segmentation," 2013.

[14] A. Ingram. https://www.carwow.co.uk/guides/glossary/parking-systems-explained, April 2015.

[15] D. I. Yourself. http://www.doityourself.com/stry/how-do-reverse-parking-sensors-work, May 2017.

[16] P. Dynamics. http://www.parkingdynamics.co.uk/Electromagnetic-Parking-Sensor-Buyers-Guide, 2008.

[17] "Proximity and ambient light sensing (als) module," 3 2016.

[18] M. B. Ken Schwaber, *Agile Software Development with Scrum*. Pearson, 2001.

[19] "Ros messages." http://wiki.ros.org/msg.

[20] "Ros services." http://wiki.ros.org/srv.

[21] "Ros publisher and subscriber." http://wiki.ros.org/rospy/Overview/Publishers%20and%20Subscribers.

[22] "Vl6180 hookup guide," 12 2015.

[23] "Arduino micro," 12 2017.

[24] "I2c tutorial," 12 2017.

[25] "Using multiple vl6180xs in a single design," 5 2014.

[26] "Arduino ide," 12 2017.

[27] "Eagle webpage," 12 2017.