POLITECNICO DI TORINO

Corso di Laurea Magistrale
in Mechatronic Engineering

Tesi di Laurea Magistrale

Visual Based Control Approach for Autonomous Parking

Relatore
prof. Stefano Malan
firma dei relatori
.....

Candidato
Abdallah Bader
firma del candidato
.....

A.A.2018/19

Abstract

Autonomous valet parking requires roaming the vehicle within the valet and as the vehicle arrives close enough to the target parking spot, a final maneuver is performed. In this research, such a maneuver is performed within a visual servoing framework; the relative pose of the parking spot is estimated relying on a monocular camera feed within a closed loop control system. Two control algorithms are used to solve the motion control problem, one is a cascade proportional controller with which the parking can be initiated from a defined region, another is an optimization-based control law that works well for L curve parking cases.

Control system is developed within MATLAB and Simulink relying on synthetic image of a parking spot such that an online imagery feed is acquired. The simulation is divided into modules mainly related to perception, monocular camera model, the single-track kinematic model of the vehicle and finally the control algorithms. In addition to simulation, parking spot detection techniques of real scenarios are investigated.

Preface

Marelli is an international company that supplies leading car makers with hi-tech systems and components. Part of Marelli's mission is to combine quality, competitive offer, technology, and versatility, with the goal of making technologies available for the end user at a competitive price.

Marelli invites graduate students to develop their thesis activities at the company's site, this dissertation is written to summarize all the activities done at Marelli related to the research topic. The dissertation is to be provided as part of the requirements for the M.Sc. degree in Mechatronics at Politecnico Di Torino. The dissertation includes the definition of the tackled problem, state of the art analysis, realized innovative methods, results and relevant discussions.

To end with, many thanks to Prof. Stefano Malan for making possible to do the thesis work at Marelli. Much of appreciation for my supervisor from the company Dr. Alessandro Amodio for his great supervision to help me facilitate my thesis work. Many thanks to Dr. Michele Giorelli from control management for his valuable advices to proceed my work closely to the scope of work at the automated driving innovation group. And finally, thanks to everyone at Marelli that helped shaping my professional growth.

Table of Contents

Table of figure

Chapter one: Introduction

1. **Recognition of the need for autonomous parking**
2. **Scope of thesis within Marelli**
3. **Automated parking on SAE4 level**
4. **Final parking maneuver problem definition**
5. **The localization problem**
6. **The visual servoing framework**
7. **Parking spot detection**
8. **Visual perception**
9. **Monocular camera as vision sensor**
10. **Visual servoing approaches**
11. **Automatic Parking in literature**
12. **Commercial status of autonomous parking**

Chapter one: Introduction

1.1 Recognition of the need for autonomous parking

Once a driver reaches his destination, he would leave his vehicle, not having to worry about finding the parking garage, he would simply command to initiate the parking process and the car would park on its own. In this way, the driver would reach his destination worry free of being late, especially in the cases of going to an airport, or catching the train. Another part of the user comfort aspect is that maneuvers can become tricky in narrow parking situations, it is in fact the main reason behind non-fatal accidents. Autonomous parking means no more tricky maneuvers required from the driver and no more accidents in the parking lot.

As the number of vehicles on roads are increasing, it is important to find enough parking space for the vehicles, autonomous parking means the extra parking space for opening and closing the door is no longer needed, which provides extra parking places instead.

Autonomous drive application is catching more public interest in a sense that there is a demand over vehicles equipped with autonomous drive technology, autonomous parking is a convenient first step towards autonomous drive on roads; achieving robust parking systems serves as a cornerstone for further developments in the field.

1.2 Scope of thesis within Marelli

Marelli works on autonomous drive solutions, one of which is the autonomous valet parking over an SAE4 level. First step of the parking scenario is to find the parking spot and stop the vehicle in a close position, while the second step is to perform parking maneuvers. Performing the parking maneuvers requires precision as there is the risk to hit other vehicles or end up in a position occupying more than one slot.

In this thesis, the final parking maneuver is under study, it is to be performed relying on estimated measurements from a monocular camera in a visual servoing framework. Figure 1.1 shows the autonomous vehicle ready to initiate reaching its target spot, this is a typical parking situation to solve for this research.

Figure 1. 1 An autonomous vehicle ready to initiate parking maneuver

1.3 Automated parking on SAE4 level

Automated driving is classified into six levels of autonomy according to the society of automotive engineers (SAE 0-5). On the 3rd level, the driving task is automated while the driver is expected to respond appropriately to any request to intervene. On the 4th level the driving task is automated and the human driver intervention is not required at all, however if for any reasons the vehicle is not able to perform the task autonomously, the vehicle would drive itself into a safe situation, untill the human intervention is available.

1.4 Final parking maneuver problem definition

While autonomous parking within a valet requires a solution of a complex architecture, the final parking maneuver can be achieved in answering two questions:

1. From a parking distance, which location the vehicle is to be parked at?
2. How to actuate the vehicle in order to move it from its current position to the desired one?

In an engineering framework, knowing where to park the vehicle can be done in two ways. First, through a precise positioning system given the map of the valet. Second, using direct sensory measurements such as infrastructure sensors, distance sensor when in a closed garage or that the parking spot is bounded by two vehicles, or a visual system to perceive the parking spot features. As two approaches can be used, a throughout reasoning of choice will be covered in the chapter.

To move the vehicle towards its designated pose, control inputs of steering and longitudinal velocity are to be carefully decided throughout the way towards the final pose. In literature, there exists many solutions which relies on path planning techniques that require precise positioning. Other techniques rely on direct measurements when the parking spot is bounded in between two vehicles or is within a home parking garage.

1.5 The localization problem

In one scenario, the valet parking map is known, and there exists a set of sensors which provides a precise position of the vehicle relying on GPS, IMU, and Gyroscope. To locate the relative distance between the vehicle and the parking spot, the position of the vehicle within the map is to be known within a localization framework. Muehlfellner et al. [1] combines visual features together with odometry from the wheels to localize against a previously built map of a parking

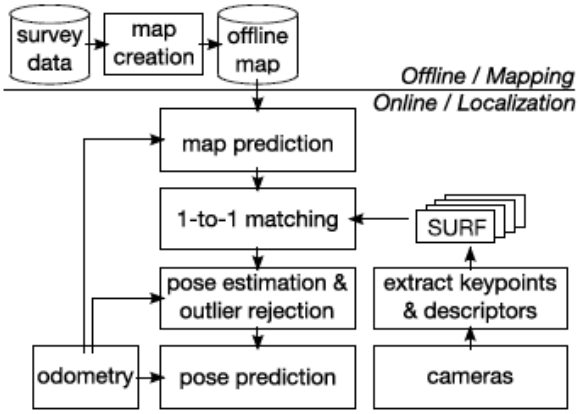

Figure 1. 2 Localization scheme using camera and odometry [1]

valet. The study further shows that within week one, a maximum localization offset error of roughly 20 cm was recorded, while after two weeks, the map becomes unreliable to perform the localization. The data were taken through a two-month timeframe. the technique fails to localize because of the change in the appearance of the environment as concluded by researchers [1].

As shown in figure 1.2, Localization scheme uses previous knowledge of map, measured odometry, and finally camera is used to match the odometry with the map using extracted features.

The localization technique shown in figure 1.2 brings into count three sources of errors. One, the quality of the recorded map, two, errors in odometry, and finally error in the camera features detection due to change of appearance in the environment.

GPS can be used to avoid the drift in odometry coming from wheels. A commercial GPS has an accuracy of 5 m, while a pinpoint GPS has the accuracy of 30 cm though it comes expensive; plug and play pinpoint GPS systems can be easily find online for around 500€.

Visual servoing has less sources of errors than the precise positioning setup, first source of error comes from position estimation of the object within image feed, second source of error may come from the specific controller in use. Monocular cameras are very cheap compared to GPS; an automotive grade camera can be easily found for less than 100€.

For the above analysis, to achieve better accuracy, and economic convenience, the visual servoing framework is utilized for the final parking maneuver.

1.6 The visual servoing framework

Visual servoing utilizes information perceived from a visual sensor within closed loop control scheme. In our case, a monocular camera is used that is mounted above the center of the vehicle. Visual servoing is classified mainly into two approaches, image-based approach which relies on features in the image space, and positional-based approach which relies on estimated features in the 3D space. Other classifications consider mixed techniques.

All visual servoing techniques requires detecting features of the desired object in the image space. Parking spots comes in different forms seen in Fig1.3 [2] and some of these forms comes in different shapes as shown in the Fig1.4 [3]. Four corners of the parking spot provide a complete

Figure 1. 3 Perpendicular parking layout of diifferent types [2]

Figure 1. 4 Parallel parking(left), Slanted (center), and perpendicular (right). [3]

description of the parking spot of different types, or simply two corners in addition to width and length.

1.7 Parking spot detection

To control the vehicle towards reaching the parking spot in a visual framework, the parking spot has to be detected as this makes a cornerstone within the closed loop control scheme. As parking spots are marked with similar to those of lanes, a lane detection approach will be discussed. Further, a learning-based approach is discussed that is designed specifically to detect parking spots.

1.7.a Parking spot detection: a lane detection approach

As parking spots are formed with markings similar to those of lanes, the problem can be tackled with a similar approach. As there is lack of research tackling specifically the parking spot detection, a lane detection approach was first looked upon.

Looking into research on lane detection [4] [5], the lane detection process is usually done in a similar manner; the visual feed is cropped to only those of relevant areas that includes the lane marking, the vanishing point is calculated to identify relevant areas. The original image goes through transformations starting from distortion removal, processing into a greyscale image, as well as smoothing the edges. Edges within the image are identified using Canny method or other edge detection techniques. Finally, the lines in the image are identified using Hough transform.

Nava et al. [4] demonstrates the lane detection technique in day and night situations shown in figure 1.6. It is further shown that detected lines are finally filtered to acquire the correct lane markings, the image is separated into right and left sections where each left and right lane are detected accordingly. The vanishing point is estimated to be in the middle of the image on the horizontal scale, while experimentally approximated on the vertical scale, then a vanishing segment is formed. The horizon line is detected, and the above area is cropped to reduce the complexity of the image processing as shown in figure 1.5.

Figure 1. 5 Nava et al [4] demonstrates horizon line identification and lane detection in day light and night situation

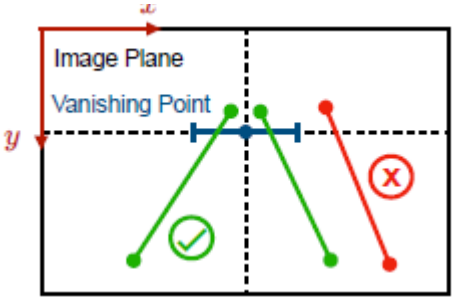

Figure 1. 6 Nava et al. [4] demonstrates filtering of lane lines, and setting up the vanishing segment.

1.7.b Parking spot detection: a learning-based approach

Parking markings comes in different forms, colors, and environment. A detection algorithm has to deal with variety of indoor and outdoor lighting situations. A good algorithm is robust to degradation in the markings color.

From literature, two researches stands out in the parking spot detection domain are [2] [5], a large set of positive and negative parking samples were taken, a set of features from the image are specified, and finally markings are manually labeled where features are correlated with correct marking and an offline learning algorithm is formed, see figure 1.7.

While precision of this technique is very promising ranging from 93.15% for outdoor with slanted park spots to 99.34% for indoor situations, it requires further efforts to prove feasible feature tracking within a video feed, and requires further applicability analysis within a visual based control approach.

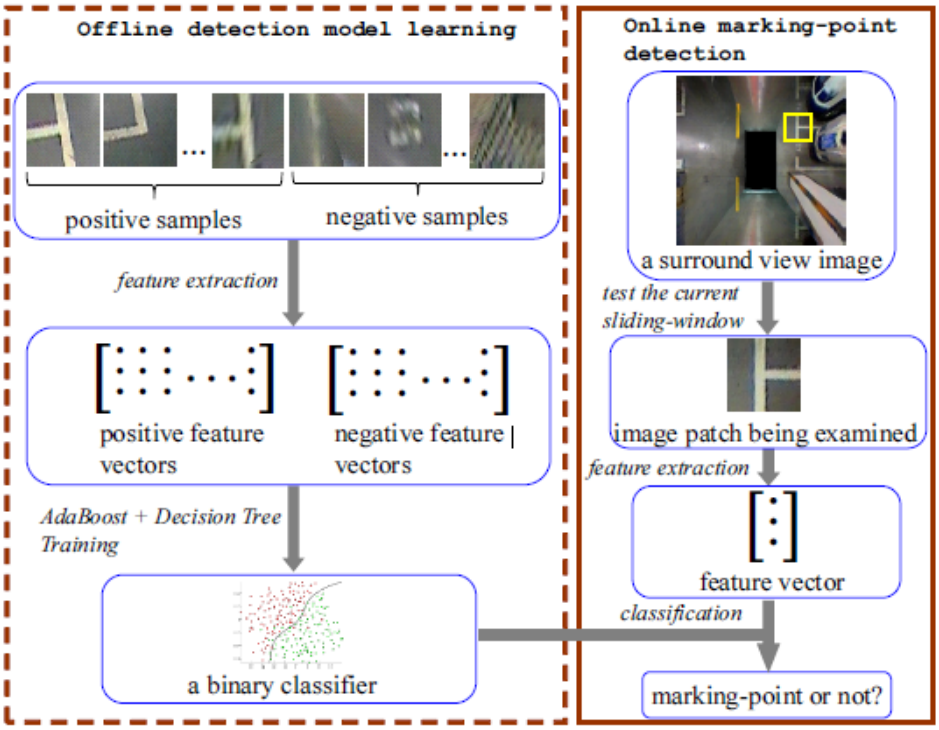

Figure 1. 7 Processing flow of the learning based apparoch to find two marking points of the parking spot. [2]

1.7.c Challenges with parking spot detection

Recent research [2] shows that parking spot detection is a yet to be solved problem, the complexity of the problem is due to firstly, the image vagueness: as there may be occlusion shadows from near buildings, or vehicles. Secondly, there can be variating light conditions throughout the day, and in the cases of indoor versus outdoor parking. Finally, the markings come in different shapes colors, and textures, and those wear out over time [6].

Another challenge is that there does not exist a publicly available dataset of parking spot imagery to benchmark against or use within a learning-based framework.

1.8 Visual perception

Right after detecting desired features of the parking spot within the image frame, in order to perceive information their 3D counterparts, a mathematical model of the camera can be utilized.

Mathematical model of the camera basically describes how points in the environment are transformed into pixels in the image space.

Visual servoing techniques rely on information related to the camera model, for example, the positional based visual seroving requires knowing the 3D counterpart of each feature, while image based visual servoing requires knowing the depth of a feature in the 3D space.

To perceive information correctly, the lenses convex nature has to be taken into count. Acquired images from cameras are distorted. Distortions can be both negative or positive. May also be radially or tangentially as seen in figure 1.8. To correct those situations, distortion parameters of the monocular camera are to be estimated.

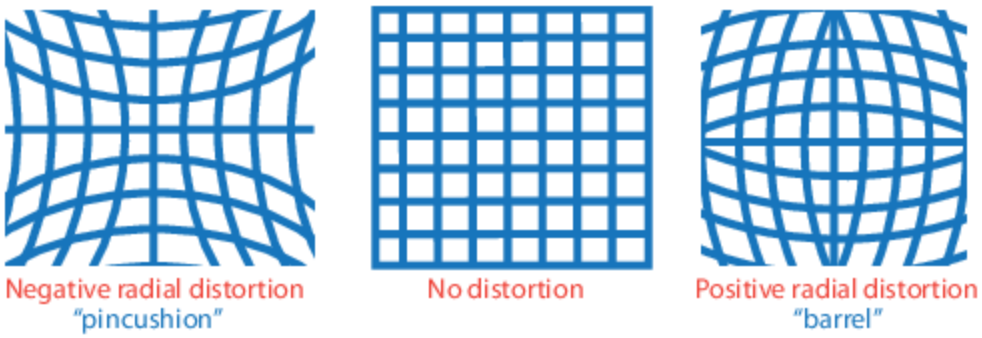

Figure 1. 8 Radial distortion in the image space both negative and positive from MATHWORKS

1.9 Monocular Camera as vision sensor

A monocular camera is a simple cheap device that can be used for visual servoing, while stereovision cameras provide information on depth within the image, the steereo-vision camera is usually 10-20 times more expensive than the monocular camera.

To visualize the monocular camera among its parameters the tool found in [7] is suggested. The tool demonstrates the effects of changing the camera parameters, both intrinsic and extrinsics.

Camera model is basically the mathematical equations which transforms the 3D point into pixel point in the image space. This is done in relation with physical parameters by which the camera can be described. Within MATLAB, the pinhole model is the standard model among MATLAB perspective transformation functions.

1.10 Visual servoing approaches

Visual servoing refers to the use of computer vision data as input of real time closed loop control schemes to control the motion of a dynamic system, a robot typically [8]. Visual servoing is classified into two main approaches, those approaches will be discussed while having in mind the applicability over controlling the pose of the vehicle.

1.10.a Image-Based Visual Servoing (IBVS)

IBVS utilizes pixel features extracted from image as feedback within the closed loop control system. Such features include: points, lines, and ellipses. The overall objective is to stabilize the image towards a desired feature.

Figure 1.9 shows the overall control approach where a desired feature of the object is provided, in our case the parking spot image is desired to be at the center of the image, features of the parking spot is continuously detected within the image and the controller calculates the velocity and steering of the vehicle such that the image finally comes to its final desired position. Figure 1.10 shows a simplified feed of the parking spot overlain with the desired image feature at the center.

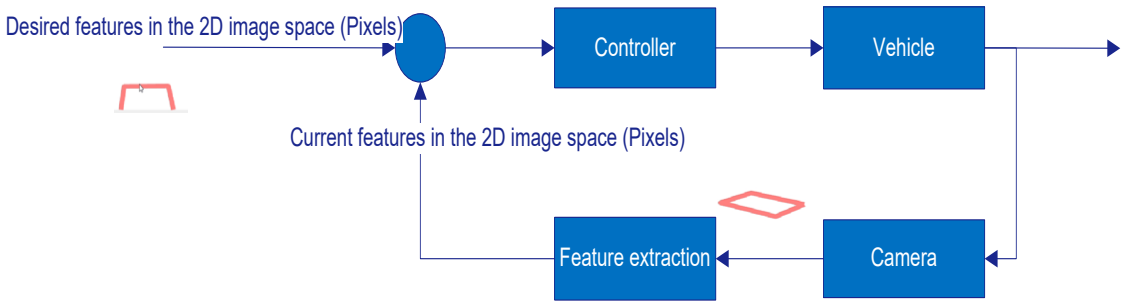

Figure 1. 9 Image based visual servoing control appraoch, overll control scheme

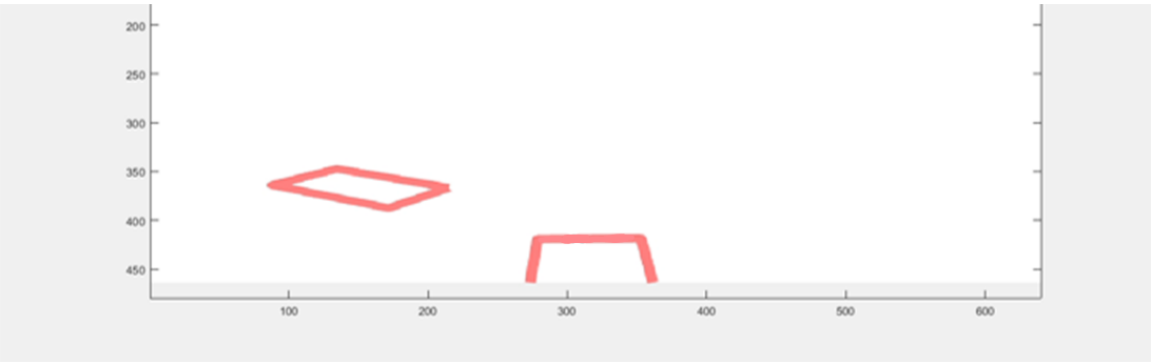

Figure 1. 10 Simplified camera feed (RGB image) of a parking spot overlain with the desired image at the center.

The IBVS approach is tailored for robotic manipulators in which the robot is well actuated. A vehicle can be considered as a non-holonomic mobile robot that is underactuated. This means IBVS cannot be directly applied over vehicle applications [8].

One of the attempts in literature to tackle the under-actuation issue with IBVS approach in a vehicle-like scenario is to actuate the mounted camera as done by Masutani et al.[9].

Another attempt is to build a path in the image plane that passes through the initial and final pose and follows the constructed path. This method however means we are losing accuracy as we are only relying on initial measurements within an open loop framework [10].

Although IBVS faces the challenge of under actuation when it comes to nonholonomic applications, it can offer the advantage of being less sensitive to camera parameters compared to positional based visual servoing (PBVS) [11].

IBVS requires performing position estimation, so it does not fully rely on image features. The need for positional estimation can be seen through the IBVS theory from which, we need to know the 3D depth of feature points. [12]

A typical choice of image features is 2D points in the image plane, a point $\boldsymbol{x} = (x, y)$, that is corresponding to $\boldsymbol{X} = (X, Y, Z)$, a point in 3D. Those two points are related as follows:

$$x = X/Z \ = (u - c_u)/f\alpha \quad 1.1$$
$$y = Y/Z = (v - c_v)/f \quad 1.2$$

c_u, and c_v are the coordinates of the principal point, f is the focal length, and α is the ratio of the pixel dimension. Finally, $\boldsymbol{m}$ = (*u, v*) gives the coordinates of the image point expressed in pixel units

As the objective of the control law is to choose the correct velocities of the camera, there has to be mapped the velocities of the points in the image plane to camera velocities. Velocities of the points within the image plane can be derived as follows.

$$\dot{x} = \dot{X}/Z - X\dot{Z}/Z^2 = (\dot{X} - x\dot{Z})/Z \quad 1.3$$
$$\dot{y} = \dot{Y}/Z - X\dot{Z}/Z^2 = (\dot{Y} - y\dot{Z})/Z \quad 1.4$$

Velocities of the points in the image plane can be concluded in the following form

$$\dot{x} = L_x v_c \quad 1.5$$

Where $v_c$is the camera velocity, and the interaction matrix L_x organized as follows

$$L_x = \begin{pmatrix} -1/Z & 0 & x/Z & xy & -(1+x^2) & y \\ 0 & -1/Z & y/Z & 1+x^2 & -xy & -x \end{pmatrix} \quad 1.6$$

Eventually, the velocities within image plane can be related with camera velocities through the interaction matrix, or image Jacobian as follows:

The basic idea of how the control law of the IBVS technique work is that velocities of the camera are calculated in such a way to minimize the error of some chosen image features. To exponentially minimize error dynamics, the control law is typically calculated:

$$v_c = -\lambda L_e^+ e \quad 1.7$$

Basically, λ is the control parameter to modify to dynamics of the error, while L_e^+ is the pseudo inverse of the image Jacobian to translate the command from pixel space toward camera space. And finally, the feature error is defined:

$$e = \boldsymbol{x_d} - \boldsymbol{x} \quad 1.8$$

Where $\boldsymbol{x_d}$ are the desired points location, and $\boldsymbol{x}$ is the measured ones. In order to calculate the interaction matrix, information on depth of feature points is required i.e. Z shown in equation 1.6 within the interaction matrix. Therefore, further positional estimation is required, or the use of other depth sensors must be utilized.

1.10.b Position-Based Visual Servoing (PBVS)

PBVS utilizes poses in 3D estimated from the image as a feedback within the closed loop control system. The final goal is to stabilize the vehicle towards the desired pose [13].

This technique requires capturing features of the parking spot in the image space- corner points in our case, and then perform inverse perspective transformation to know their relative pose in 3D.

The overall quality of the PBVS approach relies on:

a. Features detection within image space
b. Tracking features within moving frames
c. Position estimation from detected image features
d. Camera calibration parameters used in inverse perspective transformation
e. Quality of the controller in use.

Figure 1.11 shows the overall diagram for PBVS, while it may seem similar to the image-based approach, the difference is that the desired and estimated features are pose measurements of the parking spot instead of image features in pixels.

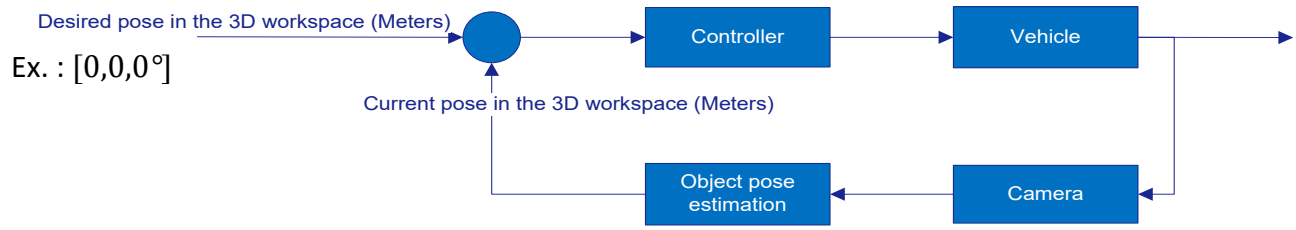

Figure 1. 11 Positional based visual servoing overall control scheme

The objective of this approach is to drive the measurements to reach the desired pose in 3D, that is to [0,0,0°] in our case as shown in the figure 1.11.

One advantage of the PBVS approach is that it separates the control problem from the perception problem. Meaning control techniques can be developed in parallel to perceiving the features of an image feed. This also means that control techniques that works well for vehicle like robot applications are applicable to work well within PBVS approach.

1.11 Automatic parking in literature

1.11.a Automatic parking of vehicles: a review of literatures (2014)

W. Wang et al. [15], reviews automatic parking research by first highlighting commercial applications of automatic parking including: systems-city park developed by Citroen, park4U developed by Valeo, intelligent parking assist developed by Toyota, Parking assist vision PAV concept from Volkswagen, and finally Audi's automatic parking "one key type of parking".

Except for the one key type of parking, none of the above systems are designed to autonomously perform the parking task. Audi's system requires the user to command the parking process from a mobile application, which would initiate the parking procedure into a parking lot, or a garage. The technique used is infrastructure based that requires laser scanners to be installed in the parking location.

W. Wang et al., further discusses important topics for automatic parking applications including: visual perception, ultrasonic and radar technology, path planning, parking control methods, image processing and recognition technology, and finally digital signal processing technology.

W. Wang et al., show that visual perception is used to complement the drivers understanding of the environment, and further eliminate blind zones. Four cameras are conveniently used to acquire around the vehicle view, binocular vision is also used for mapping purposes, and finally rearview mirror cameras are installed for rearview vision.

The use of ultrasonic and radar technology is to recognize the available parking spot and build a map for its location then drive the car into a convenient starting position to perform the parking control task.

Path planning is divided to global, and local path planning. Global paths can be planned using Dubins method, Reeds and sheep, Kanyama and Hartman, Brussel and Schutter, and finally Boer and Albada, those methods use different geometries to plan a global path. Wang et al., further explain the advantage of not requiring real time operations trading off the ability to adapt to global errors.

Local path can be planned using real-time trajectory methods including: Ross (2006), Verma and Junkins (1999), and other nonlinear geometric methods.

Wang et al., summarizes the evolution of fuzzy control methods to solve the automatic parking control problem. It is further shown how image processing and recognition is incorporated to apply the fuzzy control and similar techniques, while it is finally shown how to apply those techniques on hardware.

1.11.b Analysis and review of state-of-the-art automatic parking assist system (2016)

Song et al. [15], elaborates on semi-automatic parking assisting systems that leaves the braking and accelerating task to the driver, researchers also provide a brief over the status of fully automatic parking technologies. Song et al., further summarizes automatic parking assist systems within the modules shown in figure 1.12, in which the target position is perceived, and the current position of the vehicle is estimated utilizing ultrasonic sensors, cameras, lasers sensors and global positioning systems. Once the environment is perceived a path is generated towards the target parking location. After that, the path is followed accurately using a tracking controller.

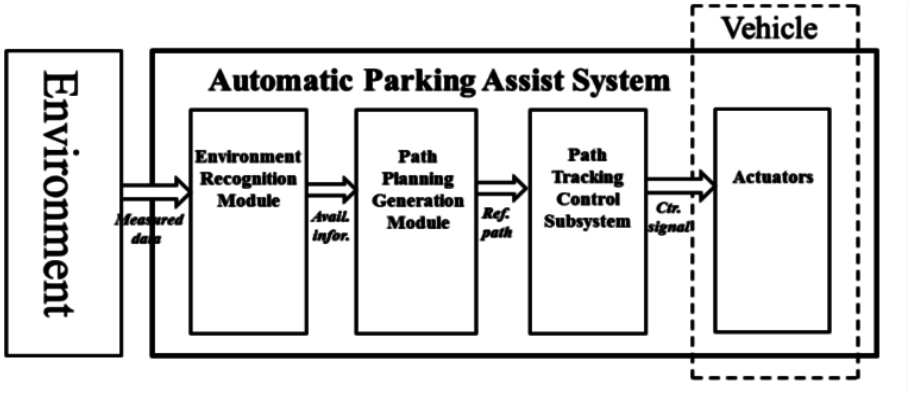

Figure 1. 12 An overall control scheme for automatic parking [13]

Song et al., elaborate further demonstrates image processing of a parking spot in which the image goes through multiple processing stages of smoothing, ridgeness, thresholding, filtering, and finally line assignment shown in figure 1.13.

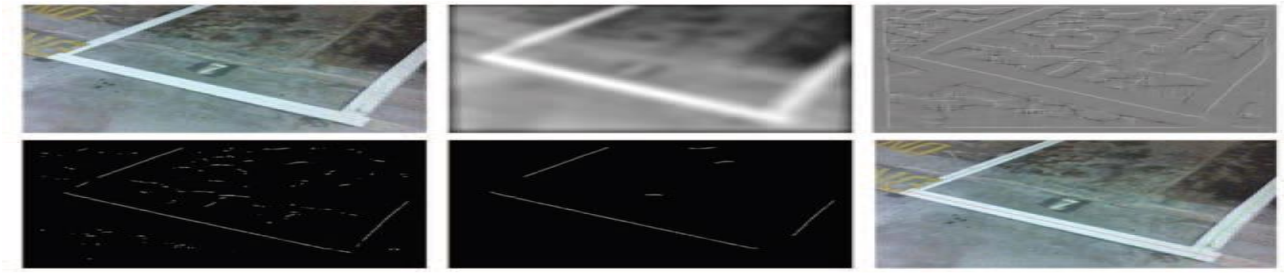

Figure 1. 13 Parking spot image processing: soothing, ridgeness, thresholding, filtering, & line assignment [13]

1.11.c Visual servoing parking with limited view angle (2003)

Muriere et al. [16], suggests a feedback control method for parking application, the problem is approached as a set point stabilization problem, and it is solved through feedback control law which uses the Lyapunov criteria to solve the motion reaching problem. The paper also suggests a hybrid control technique derived from the fact that Lyaponov functions are not unique, so multiple Lyapunov functions are used and are being switched in between when there are view limitations.

1.11.d Reverse Parking of a Model Car with Fuzzy Control (1996)

Research [17] suggests a fuzzy logic control to perform the reverse parallel parking. It uses the model shown in figure 1.14 as a world model from around the vehicle. The idea is that once the parking space is perceived, simple rules are executed that leads to actuation of the vehicle to reach the desired parking location. figure 1.15 shows an example in which the vehicle is evolving in motion to reach the parking space. Even though the presented technique can work well for typical parallel parking applications, it does not directly apply to our typical parking scenario.

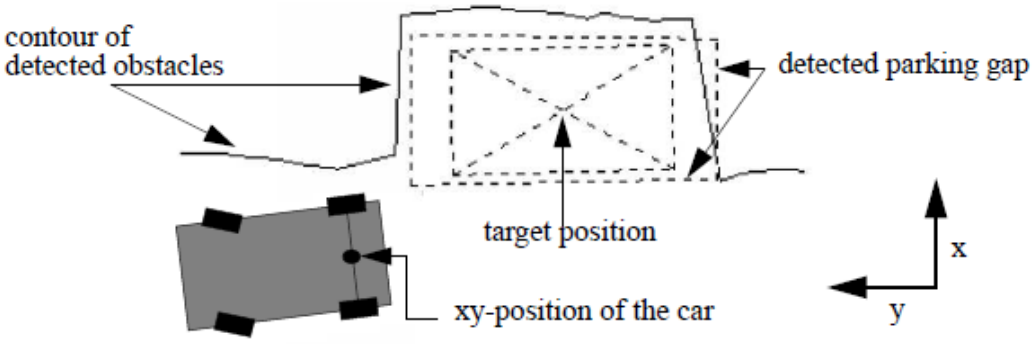

Figure 1. 14 Surrounding environment setup for a fuzzy control parking

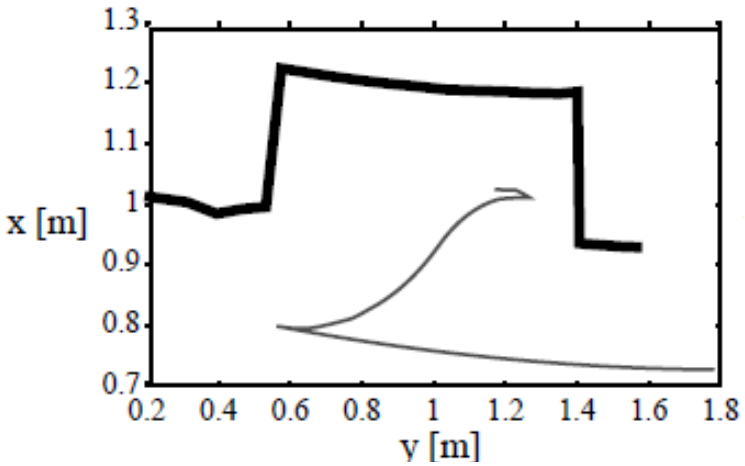

Figure 1. 15 Motion evolution of a fuzzy control parking

1.11.e Skill-Based Visual Parking Control Using Neural and Fuzzy Networks (1995)

[18] Suggests an approach that transfers the skills of an experienced driver to an automatic control algorithm. An expert driver acts intuitively while conventional control methods may fall short of action due to the inherent nonlinearities of the set point stabilization problem. Figure 1.16 shows the setup from which the parking trajectories were collected. While figure 1.17 shows the parking maneuvers done by the expert driver to be implemented into an automatic skill-based parking

algorithm. While this technique makes use of the skills of an expert driver, there lacks enough data sets to fully develop a working solution for all cases.

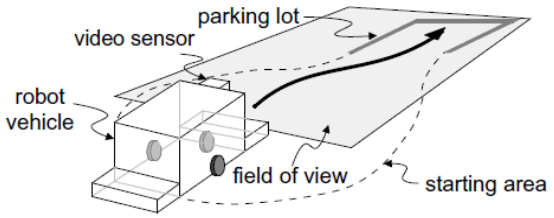

Figure 1. 16 Hardware setup for a skill based automatic parking

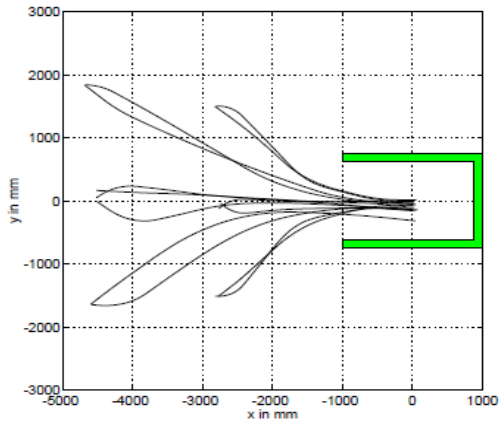

Figure 1. 17 Trajectories done by an expert driver for a skill based automatic parking

1.12.f Thesis: Design of an Image-based Fuzzy Controller for Parking Problems of a Car-like Mobile Robot (2017)

[19] suggests an automatic fuzzy controller based on imagery feedback to solve the autonomous parking problem. Figure 1.18 shows the setup in which the experiments are conducted, while figure 1.19 shows the overall control diagram in which the fuzzy controller is applied. In this research, the parking spot is detected and then a path is built and followed within the image frame, in this way, the full capabilities of the cameras are not utilized, instead an open loop approach is used to follow a built path within the image space.

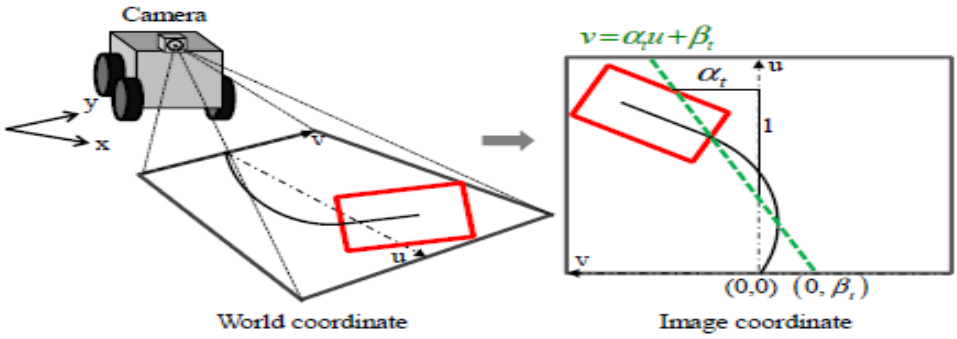

Figure 1. 18 Camera mount on vehicle to perceive parking spot in a fuzzy logic

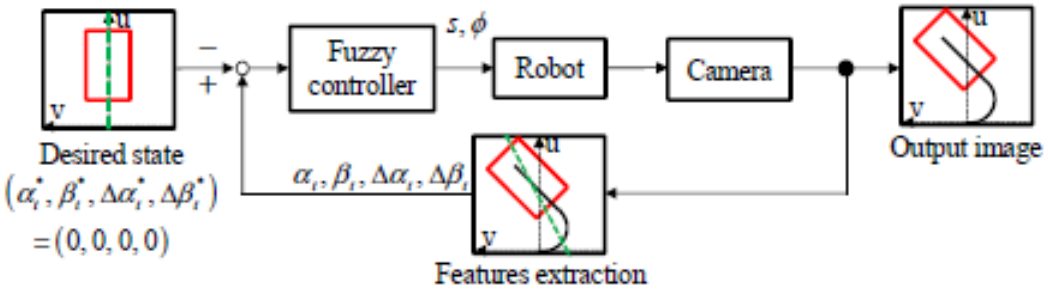

Figure 1. 19 framework Fuzzy logic closed loop control for automatic parking

1.12 Commercial status of autonomous parking

Autonomous parking technologies have a great market potential, in fact some companies are already sending drivers to do the parking for their clients, such as "Valet anywhere" from New York, "Luxe" from Silicon Valley, and many other companies providing similar services. This means autonomous parking has an already proven market potential, with such a technology, the human driver can be eliminated from the business model of those companies.

	Volkswagen	**Mercedes**	**Toyota**	**BMW**	**Ford**	**Citroen**
Parallel parking	✓	✓	✓	✓	✓	✓
Perpendicular parking	✓	✓	✓	✓	✓	×
Sensors	Ultrasonic sensor	Ultrasonic sensor	Millimeter-wave radars and CCD	Ultrasonic sensors	Ultrasonic sensors	Ultrasonic sensors
Velocity requirements	≤30km/h	≤36km/h	unknown	≤35km/h	unknown	≤25km/h
Length of Parking spot (Parallel)	≥Vehicle length +1.1m	≥vehicle length +1.2m~1.3m	≥vehicle length+1m	≥vehicle length +1.2m	unknown	unknown
Price	700 €	867€	1000$	550€	535-599€	850€
Cooperative enterprise	Valeo	Bosch	Aisin Seiki	Valeo	Valeo	Bosch
Vehicle models	Turan,Tiguan, Passat,Magoton, Audi A3	B-class B200, AMGclass, CLS350CGI, A class: A180	Prius hybrid, Lexus, Crown	BMW-5 series:535Li, BMW-7 series.	Lincoln-MKS, Lincoln-MKT	C3, C4

Figure 1. 20 Table of automatic parking system assembly case of car manufacturers[13].

Major automotive companies are putting efforts into automatic parking. Song et al., summarized commercial parking products uptill 2016 seen in figure 1.20. All those commercial systems are complementing the human driver skills and require human attention at all times.

V-charge project has helped advancing the technology for autonomous parking which were funded by European commission through the Community Research and Development Information Service (CORDIS) with an EU contribution of 5.630 M€. the project was coordinated by ETH Zurich, and

carried out by different parties from Italy, Germany and the UK. [16]. The motivations of this project were mainly to lay down on CO2 emissions, while the main idea of the project is that the parking process is initiated from a mobile app and that vehicle would autonomously park.

Most recent commercial solution for indoor valet parking has received regulatory approval as it were announced on July 2019 [17], this was done in corporation between Bosch and Daimler. Recently (2019) NVIDEA announced fully developed solution for outdoor parking spot detection system to be used for autonomous parking applications.

Chapter Two: Methods

1. **Simulation setup within MATLAB/Simulink**
2. **Synthetic camera module**
3. **Perception module**
4. **Single track kinematic model**
5. **Controller design**
6. **CARSIM model and implementation**
7. **Brake and gas control**
8. **Parking spot detection**
9. **Chapter summary**

Chapter two: Methods

2.1 Simulation setup within MATLAB/Simulink

The control problem of autonomous parking is to specify the two control commands of longitudinal velocity and steering angle over time in order to reach the final parking position without any human intervention.

As a first step of designing a controller for the autonomous parking problem, each of the component involved is to be modeled such that the controller can be designed and verified in the software environment without the continuous need to deploy over any hardware.

Simulation environment is created while having in mind that models are to be as close as possible to reality but at the same time, we need to make the simulation as simple as possible when there is no compromise over results. This for example can be seen while using the kinematics of the single-track model as the vehicle model which does not involve any dynamics of the vehicle; the vehicle would be moving slowly towards the parking spot and so we can disregard the dynamics.

The simulation models consist of the single-track kinematic model of the vehicle, synthetic camera model, the control algorithm and finally the perception module. See the overall diagram in figure 2.1.

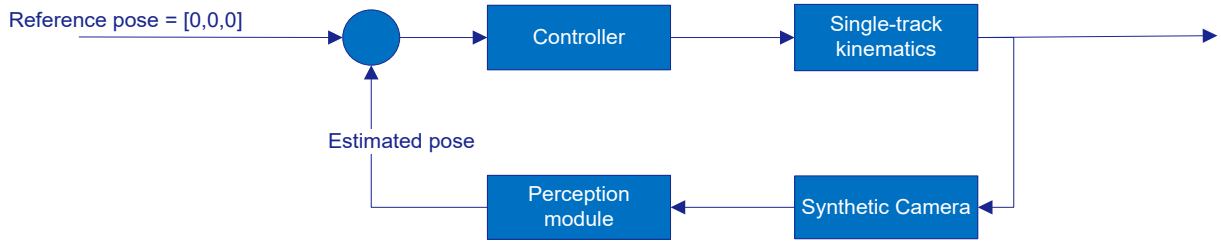

Figure 2. 1 Block diagram of the simulation setup within MATLAB/Simulink for a visual based control autonomous parking

2.2 Synthetic camera module

From the kinematics of the single track model: an update of the vehicle pose is received, the update in pose is continuously subtracted from an initially set parking spot pose in order to acquire the relative pose of the parking spot, this is done in order to finally create a synthetic image of the parking spot. To build the synthetic image, first the corners must be estimated from the synthetically updated pose. Then, having the position of each corner of the parking spot in the vehicle frame i.e. in meters, the perspective transformation is applied so the acquired corners are transformed into pixels. Finally, the corners location in the image are plotted in order to be visualized. The synthetic camera module can be summarized within the submodules shown in figure 2.2. Each submodule will be fully explained in the following sections.

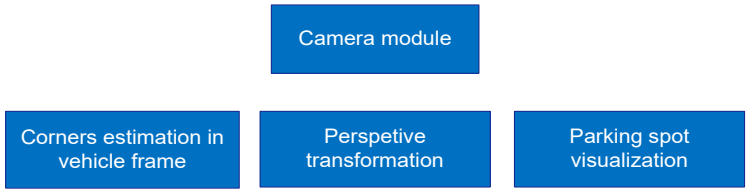

Figure 2. 2 Synthetically modeled camera submoduel

2.2a Corners estimation in vehicle frame

Given the synthetic relative pose of the parking spot in the vehicle frame, corners locations can be estimated using basic rotation and translation relations. First, the parking spot is built around the origin, then it is rotated of an angle θ and finally it is translated to its relative position. At this stage the four synthetic locations of the corners are available for further processing to finally create the picture of the parking spot, see figure 2.3.

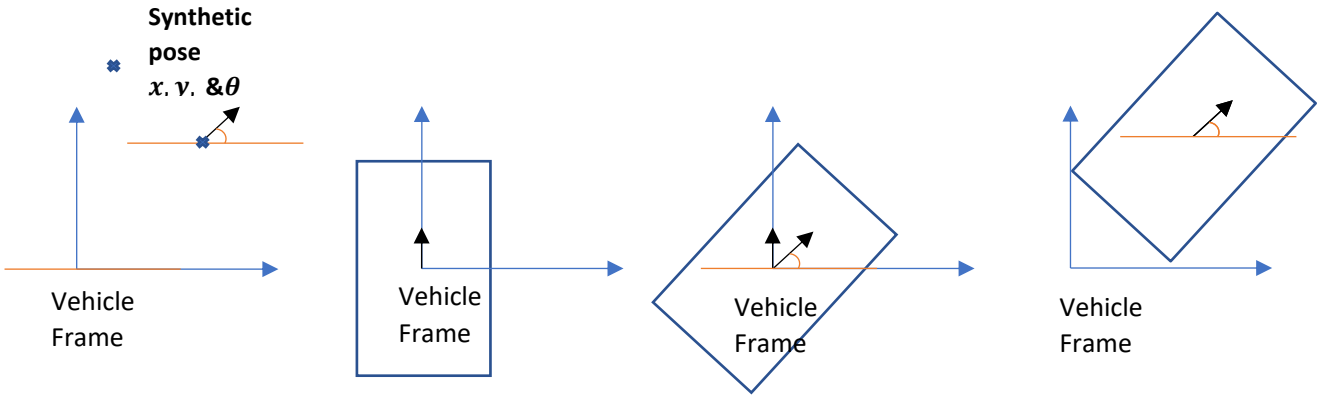

Figure 2. 3 Synthetic corners of a parking spot in vehicle frame

2.2b Perspective transformation

Once the four synthetic locations of the parking spot are computed, perspective transformation is done to transform those locations into image pixel space. This can be basically obtained using camera model, camera is simplified into the pinhole model; the pinhole allows reflected light through camera box towards the image plane which shows an upside shape of the 3D object on the 2-D image plane as demonstrated in figure 2.4.

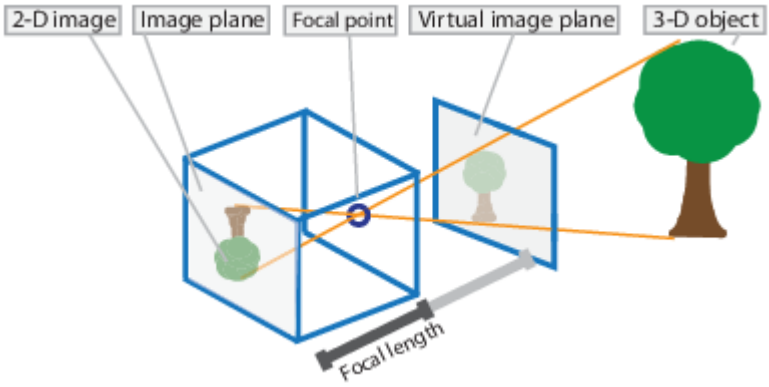

Figure 2. 4 Pinhole model of the camera along with a reflected tree from 3-D to 2-D image plane

A point in 3D can be mathematically transformed from world frame towards camera frame using the extrinsic parameters, then it is transformed into pixels using intrinsic parameters. Figure 2.5 shows the extrinsic and intrinsic parameters involved into perspective transformation. The extrinsic parameter R is the rotation matrix of the world frame relative to the camera frame, R relates to how the camera is oriented with respect to world frame. t is a translation vector which relates to where the camera frame is located with respect to the world frame. In our case, the

camera is mounted with a pitch angle of 14° and it always has the same height of 1.4 m with respect to the ground.

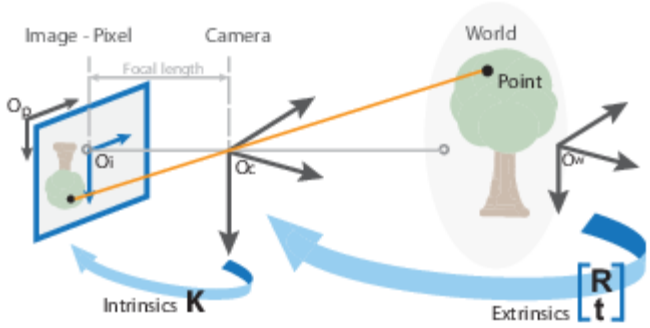

Figure 2. 5 Camera parameters and their roles in transformations from world-camera-image frames

To clearly understand the mapping between world, camera frame and image plane, figure 2.6 shows a block diagram of the transformations between camera frame, world frame and image

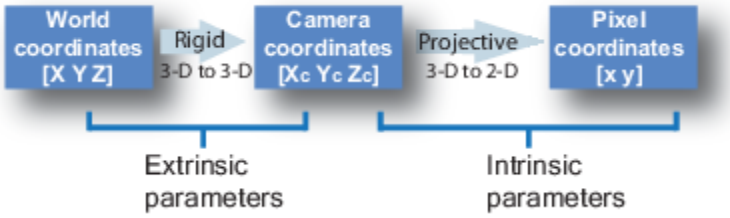

Figure 2. 6 Block diagram of Camera parameters and their roles in transformations from world-camera-image frames

plane.

It is important to understand that once we can map from 3D world space to 2D image space we can also map our way back from 2D to 3D space through inverse perspective transformations.

2.2c Camera calibration

The calibration process of a camera is performed to calculate the intrinsic parameters. While for simulation purposes any camera parameters may be assumed, a web cam is used to acquire those parameters, so the simulation is done more closely to reality.

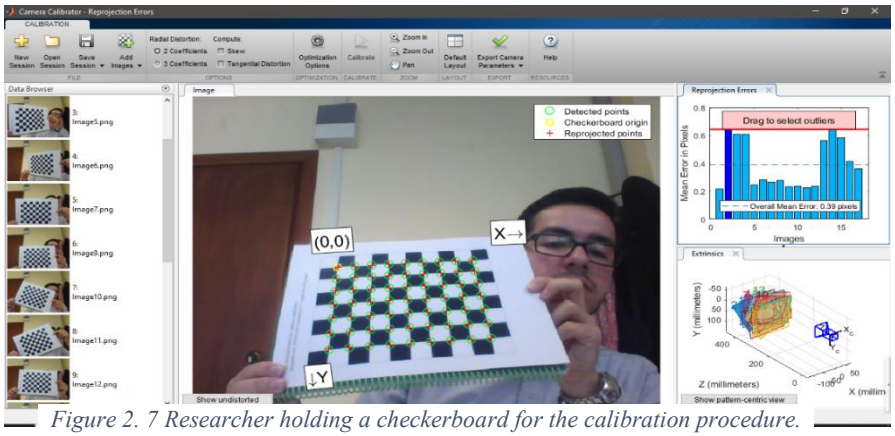

Figure 2. 7 Researcher holding a checkerboard for the calibration procedure.

Figure 2.7 shows researcher holding checkerboard in front of the camera, the calibration app captures both x, and y directions as well as the zero point within the image, the corner points of the checkerboard are also captured, and having given the dimensions of the squares the calibration problem is solved. However, such a process of holding the checkerboard in front of the camera has to be done at least 6 times [18], while the suggested number of pictures is 20 to achieve accurate calibration results according to MATLAB calibration App.

Figure 2.8 shows the manner in which the calibration process was conducted, basically 20 pictures were taken that includes the checkerboard, the board is placed around 40 cm far from the camera and it was oriented in several angles as guided by the calibration app from MATLAB.

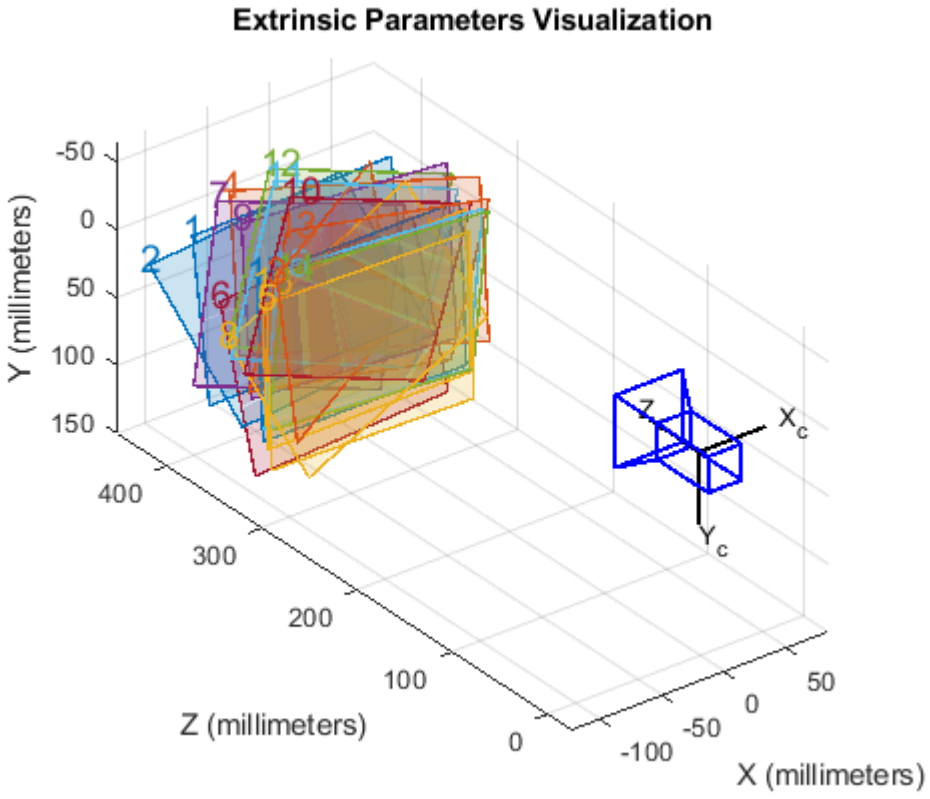

Figure 2. 8 Visualization of the distance of each frame taken through the calibration process.

Extrinsic parameters

The extrinsic parameters represent how the camera is positioned and oriented with respect to world point. In our case the camera is modeled with a mounting angle of 14° towards the ground, and an altitude of 1.4 m.

Intrinsic parameters

The intrinsic parameters make it possible to map from camera frame in meters towards image plane in pixels, figure 2.9 summarizes both extrinsic and intrinsic parameters. $px, and\ py,$ are the coordinates of the principal point, which is the point on the image plane onto which the perspective center is projected.

Intrinsic camera parameters		
Focal length	$fx = 5.151065796789960e02$	$fy = 5.158637963109842e02$
Principal point	$px = 3.168672541305037e02$	$py = 2.288944577963287e02$
Image size	$480x640\ pixels$	
Extrinsic camera parameters		
Height	$1.4\ [meter]$	
Pitch	$14\ °$	

Figure 2. 9 Table of camera intrinsic and extrinsic parameters

Once both intrinsic and extrinsic parameters are acquired, to be able to map back and forth from 3D world to 2D image plane, the camera is modeled using `monoCamera` function from MATLAB.

First, intrinsic parameters are defined within MATLAB and formulated into a single object as follows:

```
>>focalLength = [5.151065796789960e+02,5.158637963109842e+02];
>>principalPoint = [3.168672541305037e+02,2.288944577963287e+02];
>>imageSize = [480,640];
>>intrinsics = cameraIntrinsics(focalLength,principalPoint,imageSize);
```

Then, the camera is defined using the function `monoCamera` given as attributes the intrinsic parameters, as well as the height of the camera and the pitch heading angle of the camera.

```
>>height = 1.4;
>>pitch = 14;
>>sensor = monoCamera(intrinsics,height,'Pitch',pitch);
```

Camera sensor is used to transform points from vehicle coordinates towards pixels space of image points. The following function is used for inverse perspective transformation.

```
>>imagePoints = vehicleToImage(sensor,vehiclePoints);
```

To transform image points into 3D locations in the vehicle frame the following command is used

```
>>vehiclePoints = imageToVehicle(sensor, ImagePoints1);
```

2.2d Visualizing the parking spot in image space

For the synthetic camera model, perspective transformation was used to calculate points in the image plane, those points are plotted and transformed into an image of 480x640 pixels. Figure 1.10 shows a frame taken from the online feed of the camera showing the parking spot in front of the vehicle. Within the simulation environment, figure 2.10 shows a representation of the parking spot in which only the corners of the parking spot are taken into count in addition to lines connecting those corners.

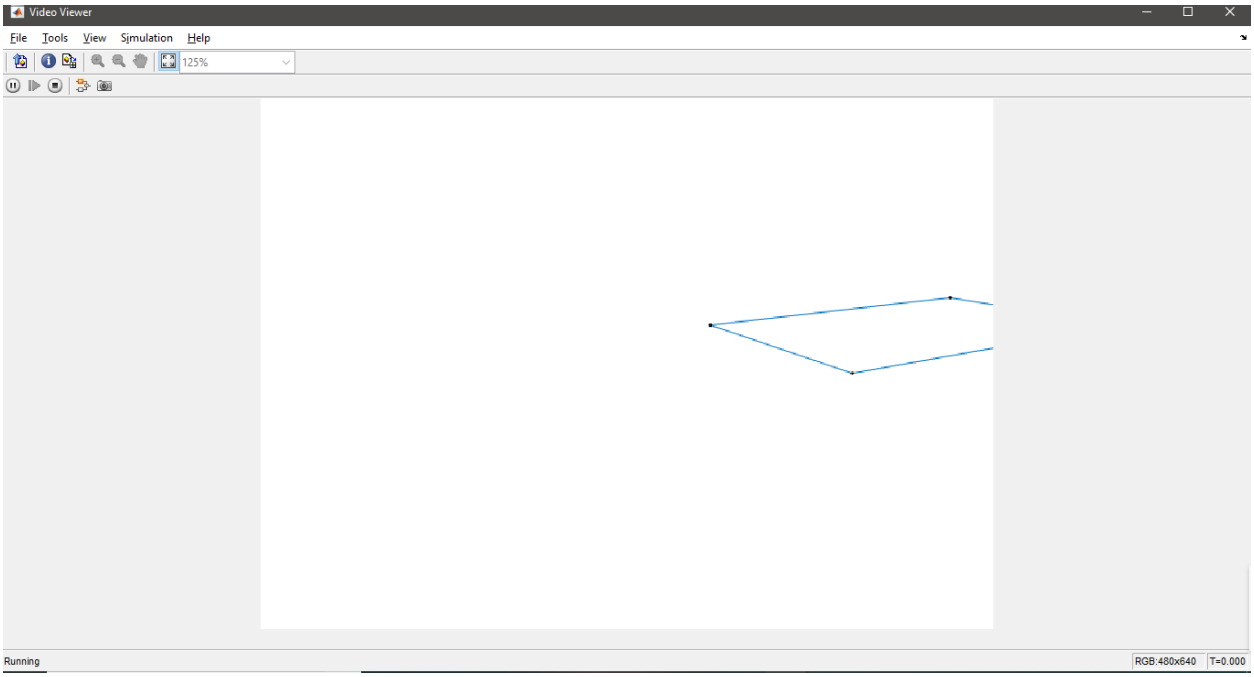

Figure 2. 10 Visualization of the parking spot mimicking imagery from a camera

2.3 Perception module

Now that we have an image that includes a representation of the parking spot, in order to estimate the position of the parking spot from imagery feed, first the detection of the corner points in the image is done, then the inverse perspective transformation is applied using the camera model equations, after that, an estimation algorithm is applied for missing corners, and finally the corner points are put together in order to finally estimate the final pose.

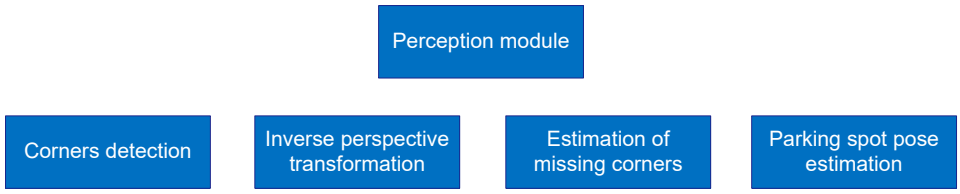

Figure 2. 11 Perception submodules

2.3a Corners detection module

As the visualization module provides a four corners image of the parking spot, this image is processed where corners are searched for. MATLAB is used to detect corner points within the

image, and it brings back not just 4 pixels but whole lot of them. To group the acquired pixels into 4 corners the following algorithm is used:

1. One corner is picked and is grouped with every other corner that is close to it in both x and y directions in the image plane.
2. Classified corners are removed from within the original set of pixels
3. A second corner is chosen and grouped with close-by corners and so the steps 1-3 are repeated till four groups of corners are acquired.
4. Finally, for each group of points, a mean value is taken such that only one cartesian position is taken for each corner within the image space.
5. In case of missing corners, the missing corner is estimated after performing inverse perspective transformations within the vehicle frame, for example: figure 2.12 shows 3 corners. The estimation is explained in section 2.3c.

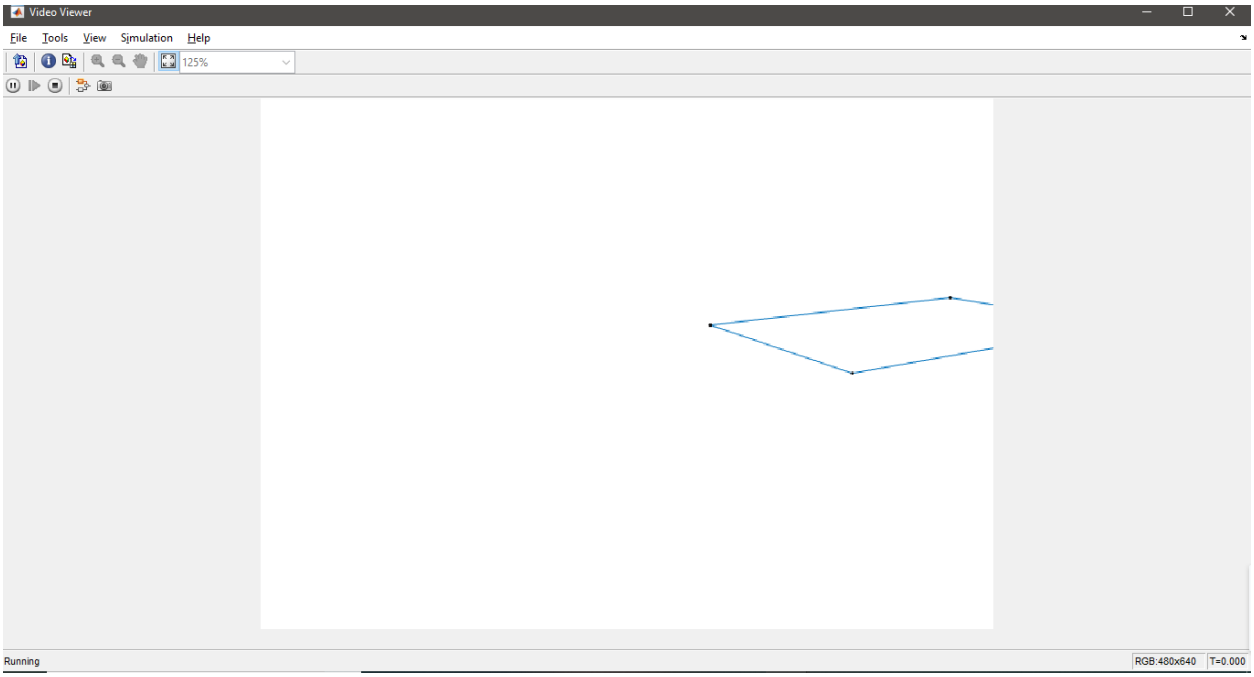

Figure 2. 12 Visualization of the parking spot mimicking imagery from a camera

2.3b Inverse perspective transformation

The modelling equations of the camera allows moving back and forth from pixel coordinates towards worlds coordinates. Once the detection module terminates its computations, acquired corners in pixel form are transformed into 3D locations in the vehicle frame using camera parameters.

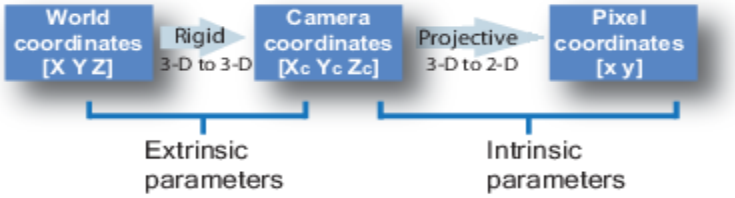

Figure 2. 13 Block diagram of Camera parameters and their roles in transformations from world-camera-image frames

While figure 2.13 shows transformation from world points to image plane the inverse can be done knowing the intrinsic and extrinsic parameters using the command:

```
>>vehiclePoints = imageToVehicle(sensor,ImagePoints1);
```

Image points are 2D pixels points while vehicle points are 3D points in the vehicle frame. Sensor is a monocular camera defined in the same way as explained when modeling the camera.

2.3c Estimation of missing corners

In some caes, only two corner points of the parking spot are visible. Knowing the length and width of the parking spot, and since two corners are already known, other missing corners can be estimated using basic geometry.

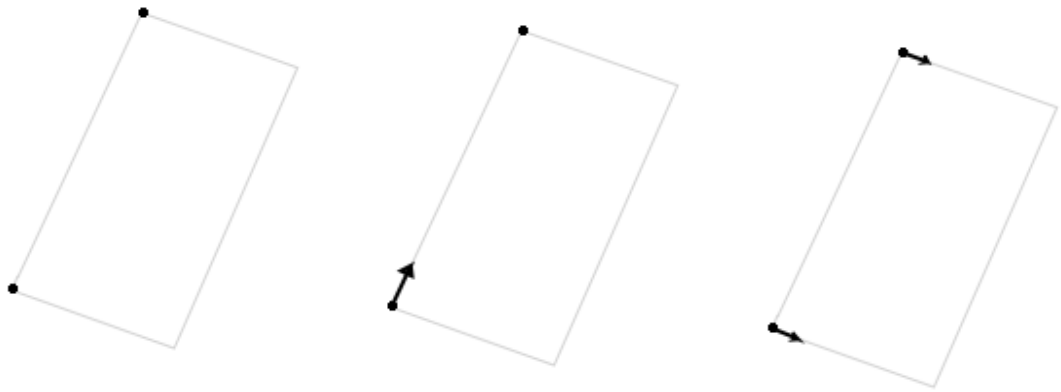

Figure 2. 14 Missing corners estimation process. Two available corners(left), a vector is built(middle) rotated vectors are built towards the missing corners (right)

Figure 2.14 demonstrate corners estimation concept, where a unit vector is constructed in between the captured two points, this unit vector indicates the orientation of the parking spot. Then other missing points are constructed by rotating the unit vector by 90° and adding the width of the parking spot to the unit vector. The rotation direction of 90° depends on which corner is available and which is being built.

The estimation method works by first checking which corners are available, then building accordingly the missing corners.

The algorithm is further designed such that if three corners are available, the fourth corner will be created based on two available corners.

2.3d Parking spot pose estimation from corner points locations

After corner points are detected and transformed into 3D, the four corners are grouped into one single pose, that is the parking spot pose. To estimate the position of the parking spot, the mean value of all four x-coordinates is calculated, as well as the mean value of the y-coordinates. This is done through the MATLAB function `mean` as follow:

```
>> xpsu = mean(cps(1:4,1));
>> ypsu = mean(cps(1:4,2));
```

Where $xpsu$ are the x coordinates of the four corner points in the vehicle frame, and $ypsu$ are the y coordinates of the four corner points in the vehicle frame. cps are the four corner points in the vehicle frame containing both x and y coordinates.

Now in order to know the orientation of the parking spot, two points are used to calculate the tangent and then the orientation is calculated using the tan inverse function from MATLAB. Where dyu is the difference in y, and dxu is the difference in x.

```
>> atan2d(dyu,dxu);
```

2.4 Single track kinematic model

Given the vehicle longitudinal velocity and steering angle of the front wheels, the single track kinematic model provides the cartesian velocities of the vehicle with respect to a fixed frame including: $\dot{x}, \dot{y}, \ \& \dot{\theta}$, see figure 2.15, the kinematics can be concluded as follows.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos\ \theta \\ sin\ \theta \\ tan\ \gamma / L_{wb} \end{bmatrix} v_L$$

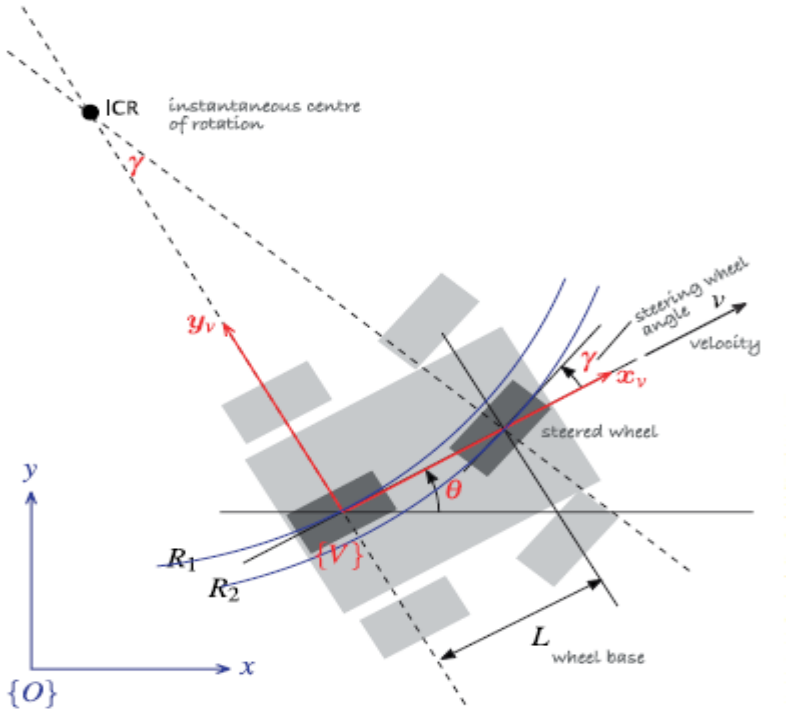

Figure 2. 15 Four wheeled vehicle reduced to two wheels single track model

v_L: longitudinal velocity

γ: steering angle

L_{wb}: wheelbase

θ: yaw angle

2.5 Controller design

2.5a Optimization based controller design

[19] suggests a minimization control law to solve the autonomous parking control problem. The desired features are the plucker coordinates of the lines of the parking spot. Figure 2.16 shows a generalized criterion of the suggested controller where six laser sensors are mounted on the vehicle. In our case, the six sensors are replaced by a monocular camera from which is perceived the four corners of the parking spot. While figure 2.16 shows a reverse parking case, in our case the vehicle is going forward towards the parking spot as shown in figure 2.17.

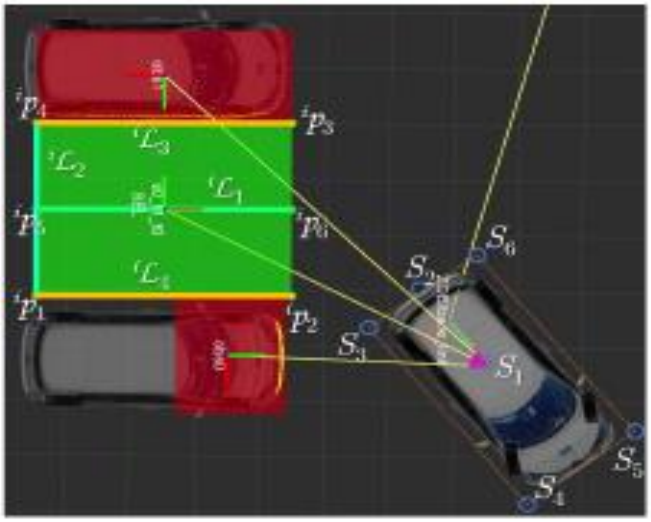

Figure 2. 16 Vehicle equipped with laser sensors in a reverse parking situation

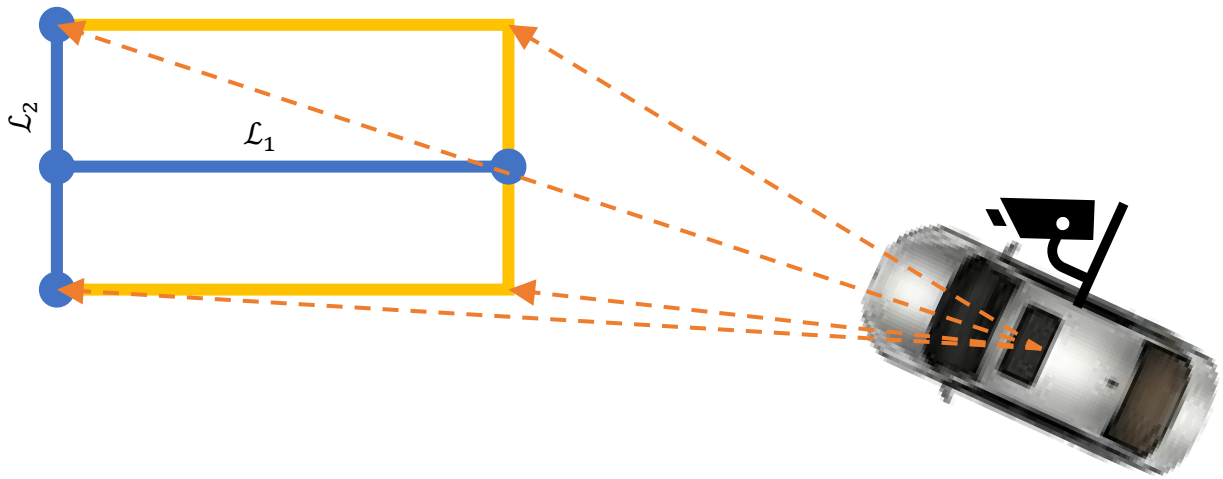

Figure 2. 17 Vehicle equiped with camera perceving corner points of the parking spot overlain with lines of the task features

Figure 2.17 shows the vehicle with a mounted camera from which each corner point is perceived. The desired features are the plucker coordinates of $\mathcal{L}_1$, and $\mathcal{L}_2$.

Plucker coordinates (u_2, w_2) of line $\mathcal{L}_2$ can be measured as the subtraction and cross product of its end points $\overrightarrow{p_4}$, and $\overrightarrow{p_1}$ shown in figure 2.18, the plucker calculations are done as follows

$$(u_2, w_2) = (\overrightarrow{p_1} - \overrightarrow{p_4}, \overrightarrow{p_1} x \overrightarrow{p_4})$$

The above calculations result in six variables from which in our case only three variables represent $\mathcal{L}_2$ because it is a planar case and the parking lines lie are on the ground plane, other three variables result equal to zero.

The normalized plucker coordinates of line $\mathcal{L}_2$ are called $s_{\mathcal{L}_2}$ that is calculated as follow:

$$s_{\mathcal{L}_2} = \left[\frac{u_2(1)}{|u_2|} \quad \frac{u_2(2)}{|u_2|} \quad \frac{w_2(3)}{|u_2|}\right]$$

$u_2(1)$ is the first element of the subtraction part of plucker coordinates, while $u_2(2)$ is the second element of that part. $w_2(3)$ is the third element of the cross-product part of the plucker coordinates calculations and finally $|u_2|$ is the magnitude of the subtraction part of the plucker coordinates.

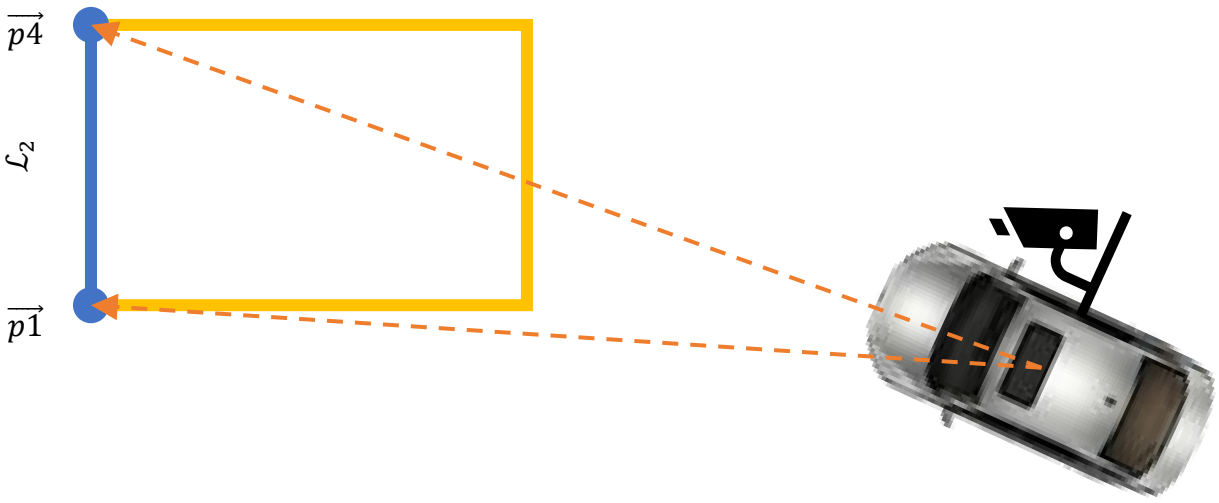

Figure 2. 18 Detected corner points used to calculate the plucker coordinates of $\mathcal{L}_2$

To come to a generalized form, normalized features $s_{\mathcal{L}_j}$ of line $\mathcal{L}_j$ can be expressed as

$$s_{\mathcal{L}_j} = \left[\underline{u_j}(1)\ \underline{u_j}(2)\ \underline{h_j}(3)\right]$$

In the above equation, $\underline{u_j}(1) = u_j(1)/|u_j|$, where $u_j(1)$ is the first element of the subtraction part of the plucker coordinates of line $\mathcal{L}_j$, and $|u_j|$ is the magnitude of the subtraction part of plucker coordinates for line $\mathcal{L}_j$. Following further on the above equation, $\underline{u_j}(2) = u_j(2)/|u_j|$, where $u_j(2)$ is the second element of the subtraction part of the plucker coordinates for line $\mathcal{L}_j$. Finally, element $\underline{h_j}(3) = h_j(3)/|u_j|$, where $h_j(3)$ is the third element of the cross-product part of the plucker coordinates of line $\mathcal{L}_j$

With this approach, task features are the normalized plucker coordinates of lines $\mathcal{L}_1$, and $\mathcal{L}_2$, and the set of task features s^t which is a 6x1 vector can be defined

$$s^t = [s_{\mathcal{L}_1}\ s_{\mathcal{L}_2}]^T$$

To eventually develop the control system there must be a way to map back and forth between task features velocity, and the vehicles velocity. This is done through the interaction matrix which is defined as follows

$$L_{\mathcal{L}_j} = \begin{bmatrix} 0 & 0 & \underline{u_j}(2) \\ 0 & 0 & -\underline{u_j}(1) \\ -\underline{u_j}(2) & \underline{u_j}(1) & 0 \end{bmatrix}$$

$L_{\mathcal{L}_j}$ essentially can be used to calculate the features velocity $\dot{s}^t$ having given the velocity of the vehicles geometric center v_m

$$\dot{s}^t = [L_{\mathcal{L}_1}, L_{\mathcal{L}_2}]^T \, v_m$$

$\dot{s}^t$ is weighted for further use in the optimization control law as $\dot{s}_H^t = H \, \dot{s}^t$ Where H is a diagonal weighting matrix.

Control law is developed such that, first objective of the control law is to close the gap in between current measurements of task features s^t and desired features s^{t*}i.e. to make the weighted task feature error $e_H^t = He^t$ as close to zero as possible. Where H is the same diagonal weighting matrix used to calculated weighted features velocity, and e^t is calculated:

$$e^t = s^{t*} - s^t$$

Another objective of the control law is to find out a solution that requires as minimum as possible the control inputs, those are the longitudinal velocity and the steering angle. Those objectives are formulated into a minimization problem which solves for v that is a 2-element scalar composed of vehicle longitudinal velocity and steering angle

$$v = argmin\|\dot{s}_H^t + \lambda \, e_H^t\|^2$$

The solution of the above minimization function is the vehicles longitudinal velocity and steering angle. Those are also implicit arguments of the minimization function through $\dot{s}^t$ the task features velocity.

The weighting function relevant to $\underline{u_j}(1)$ and $\underline{u_j}(2)$ is synthetically constructed as shown in figure 2.19. While weights relevant to $\underline{h_j}(3)$ are simply constant weighting parameters.

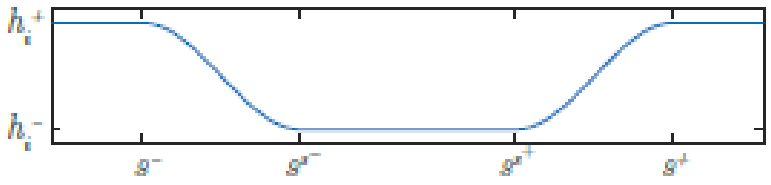

Figure 2. 19 Smooth weighting function synthetics

The idea behind the weighting function shown in figure 2.19 is to apply a higher weight for the specific feature once the vehicle is on a close distance towards the parking spot. When the

measured task feature is more than s^+ then the weight of that feature is equal to h_i^+, otherwise when the measured task feature is less than s^{s^+} then the weight of that feature is equal to h_i^-, when the measure task feature is in between s^+ and s^{s^+} then the weight is decided by a smooth polynomial function.

While solving the optimization problem, it is possible to make limitations over the acquired solution. Those limitations come in the following form:

$$Av \leq b$$

$$A = [L^c, -L^c]^T$$

$$b = [\alpha(s^{c^+} - s^c), -\alpha(s^{c^-} - s^c)]^T$$

s^c are the constrainting features being continuously estimated,while s^{c^+}, and s^{c^-}represent the interval in which s^c should remain. L^c is the interaction matrix of the chosen features and α is a gain constant.

The minimization problem within MATLAB is solved using fmincon MATLAB function, the minimization technique is discussed in [20]. That is a nonlinear solver to find the minimum of the problem specified by:

$$\min_x f(x) \text{ such that } \begin{cases} c(x) \leq 0 \\ ceq(x) = 0 \\ A \cdot x \leq b \\ Aeq \cdot x = beq \\ lb \leq x \leq ub, \end{cases}$$

Whereas provided by MATLAB *b* and *beq* are vectors, *A* and *Aeq* are matrices, *c*(*x*) and *ceq*(*x*) are functions that return vectors, and *f*(*x*) is a function that returns a scalar. *f*(*x*), *c*(*x*), and *ceq*(*x*) can be nonlinear functions. *x*, *lb*, and *ub* can be passed as vectors or matrices.

2.5b Proportional controller design in the polar coordinates

Cartesian to polar transformation:

Using simple geometries, cartesian coordinates x,y,θ can be tranformed into their alternative form of ρ, α, β in the polar space. This is done through the following equations while relating to figure 2.20 which shows the vehicle in its initial pose in relation to the goal pose.

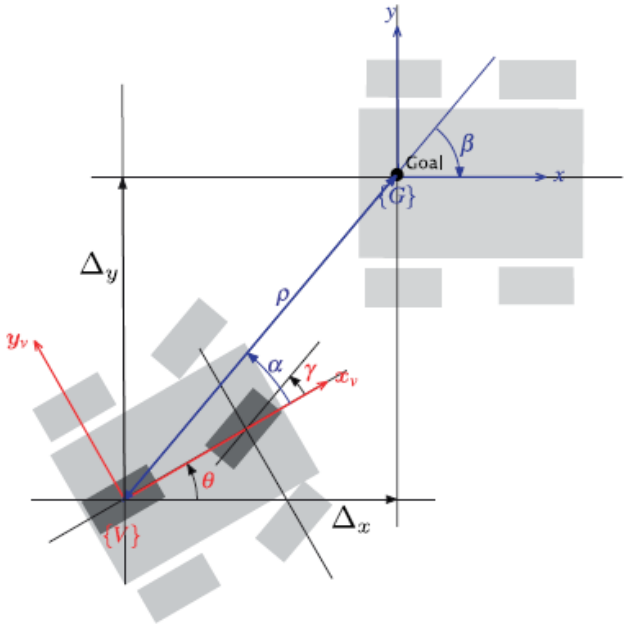

Figure 2. 20 Definition of relative polar coordinates in between current and desired pose

ρ is the distance shortest in between vehicle frame and the goal pose, it is simply calculated using Pythagorean theorem

$$\rho = \sqrt{\Delta_x^2 + \Delta_y^2}$$

α is the angle between ρ and x_v that is the longitudinal heading line of the vehicle, α is calculated over two steps by first calculating the angle between ρ and Δ_x then subtracting the yaw angle of the vehicle θ as follows.

$$\alpha = tan^{-1}\frac{\Delta_y}{\Delta_x} - \theta$$

β is the angle between ρ and the desired longitudinal heading of the vehicle, calculated as follows

$$\beta = -\theta - \alpha$$

Kinematics in the polar domain

To map from longitudinal velocity and steering angle towards the polar coordinates velocities the kinematics are formulated as follows

$$\begin{pmatrix}\dot{\rho}\\ \dot{\alpha}\\ \dot{\beta}\end{pmatrix}=\begin{pmatrix}-cos\alpha & 0\\ sin\alpha/\rho & -1\\ sin\alpha/\rho & 0\end{pmatrix}\begin{pmatrix}v_L\\ \gamma\end{pmatrix}, if\ \alpha\in\left(-\frac{\pi}{2},\frac{\pi}{2}\right)$$

$\dot{\rho}$ can be calculated given the longitudinal velocity and α using basic geometries. The angular velocity of the vehicle $\dot{\beta}$ can be calculated as the velocity vector perpendicular to ρ divided by ρ and finally once subtracting the wheel steering angle $\dot{\alpha}$ is obtained.

Control law:

To influence ρ towards reaching zero, the vehicle longitudinal velocity v_L is commanded where k_ρ is a control gain as follows

$$v_L = k_\rho \rho$$

Then, to influence the yaw angle to reaching its final orientation, the following steering angle is commanded where $k_\alpha,\ and\ k_\beta$ are control gains.

$$\gamma = k_\alpha \alpha + k_\beta \beta$$

Now that we have formulated the control law, we can substitute γ and v_L into the kinematics equation as follows

$$\begin{pmatrix}\dot{\rho}\\ \dot{\alpha}\\ \dot{\beta}\end{pmatrix}=\begin{pmatrix}-k_\rho\ cos\alpha\\ k_\rho sin\alpha - k_\alpha\alpha - k_\beta\beta\\ -k_\alpha\ sin\alpha\end{pmatrix}$$

k_ρ must be greater than zero, this condition assures that the vehicle is accelerating towards the target, not backwards from the target. The following condition must be also met to keep the system stable [25]:

$$k_\beta < 0, k_\alpha - k_\rho > 0$$

Path evolution

Figure 2.21 shows how the path of a vehicle is evolving while using the cascade proportional controller to reach the goal.

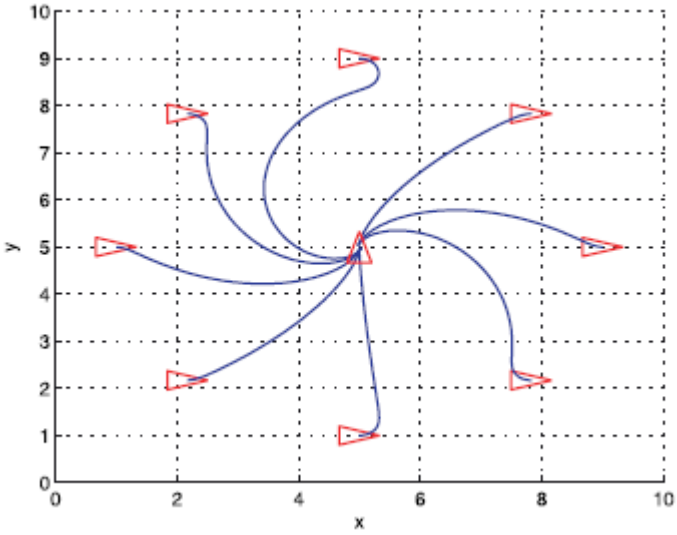

Figure 2. 21 Path evolution resulting in multiple final poses denoted in red

2.6 CARSIM model and implementation

In this section, the configurations for the Carsim model will be shown starting with figure 2.22. Vehicle configuration were left to the default one, while the procedure is set to constant speed for Ext driver control, which allows simulating the situation in which the speed and steering are externally provided.

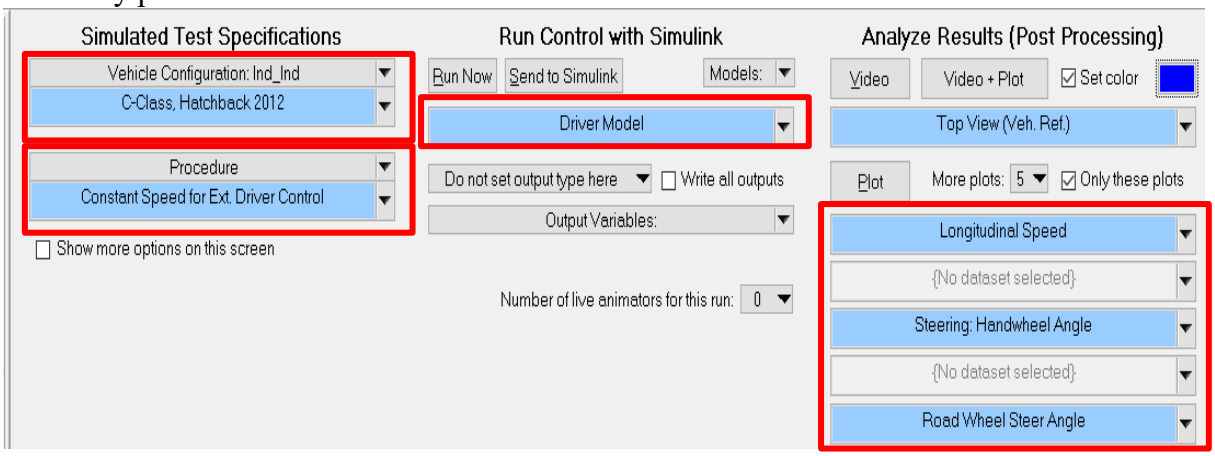

Figure 2. 22 Carsim model configuration

Driver model is selected from the middle pane from which model imports and exports can be specified as shown in figure 2.23. Analyze Results pane from figure 2.22 allows to choose which variables are to be plotted with respect to time once simulation is performed.

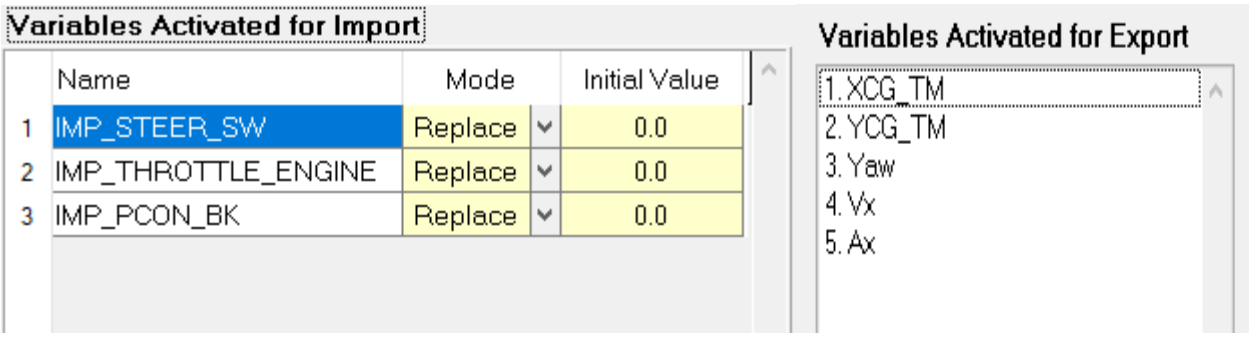

Figure 2. 23 Carsim model imports and exports

For further specification over procedure, see figure2.24. Those configurations allow for letting control signals of brake, throttle, and steering into the Carsim model. Figure 2.24 shows Path station is set to 1.105 m; this is done in order to start from the zero cartesian point within the simulation plane.

For the visual environment, the 1200 m one lane w/ tree selection is further modified as a 1km Square 10 m grid.

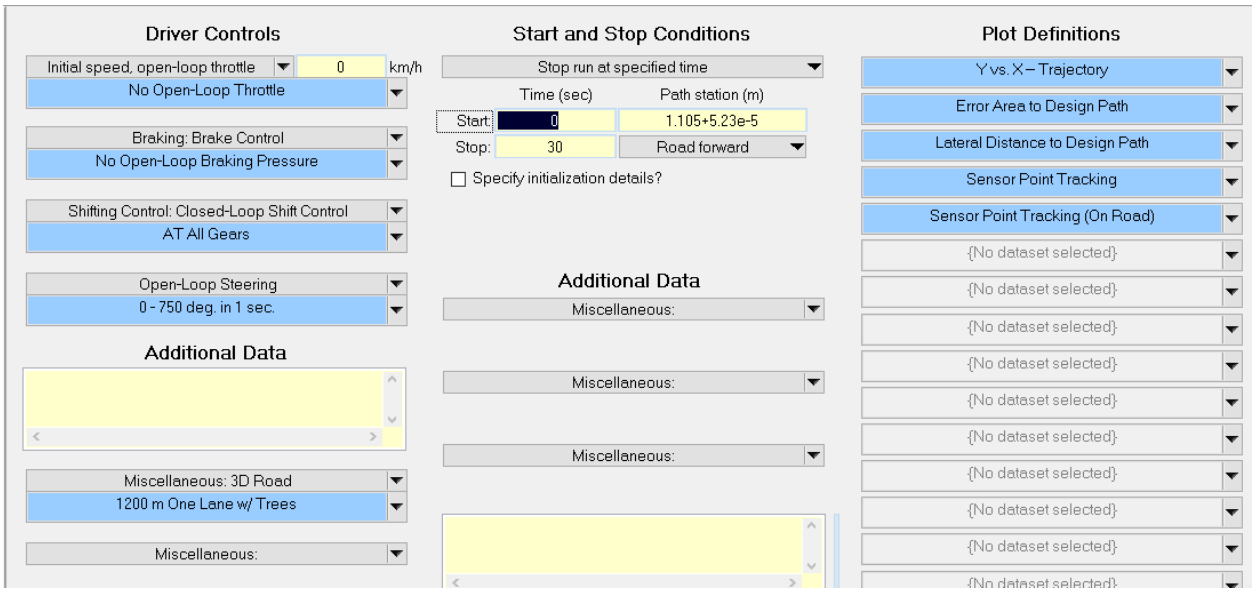

Figure 2. 24 Carsim model open loop model configuration so inputs can be set externally

2.7 Brake and gas control

Carsim model takes as an input the driver steering wheel angle, as well as the brake and gas pedal inputs. Instead, in the simulation environment, inputs are the steering angle of the wheels, and the longitudinal velocity. In order to further verify the controller performance that was simulated with the single track kinematic model in hand, a connecting block must be built which takes the velocity and steering wheels angle as an input and modify this signal to brake and gas commands as well as the driver's wheel steering angle, see figure 2.25.

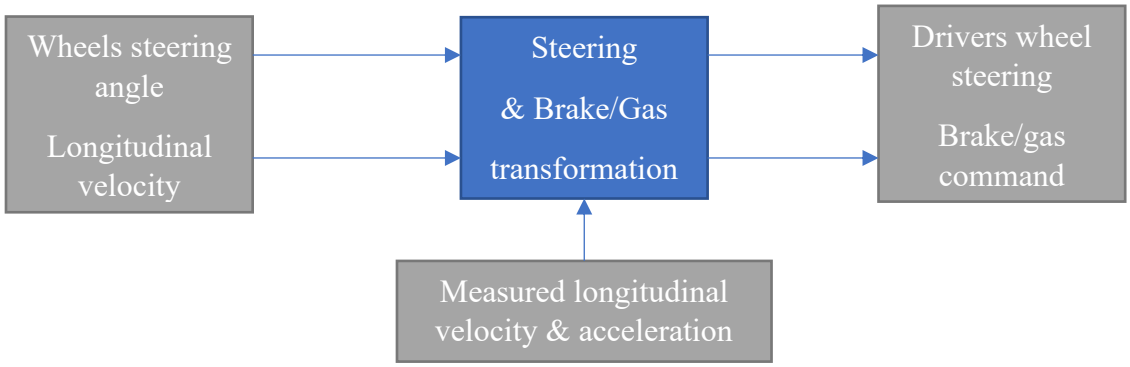

Figure 2. 25 Inputs and outputs of a steering & Brake/Gas transformation model.

2.7a Steering angle transformation

For the single-track kinematic model, steering angles of the front wheels is taken as an input, while Carsim model requires as an input the angle of the drivers steering wheel. A simple gain ratio is applied in order to make the transformation. In our case, a ratio of 17 has been applied. This means for example, if we would like to turn the front wheels by 10 degrees, driver would need to apply 170 degrees turn of the steering wheel.

2.7b Brake and gas transformation

The brake and gas model is taken from [21], the model is essentially a feedback compensator which takes as an input longitudinal velocity and acceleration as well as the reference velocity to be tracked. The feedback controller works in the following criteria:

1. It calculates an error as the subtraction between desired and current velocity, it multiplies the error with a gain resulting in a form of an artificial acceleration.
2. Current acceleration measurement is subtracted from the artificial acceleration calculated in step one resulting in a relative error in acceleration.
3. If the resulting error is more than zero, a proportional acceleration command is applied. While if the resulting error is less than zero, a proportional brake command is applied.
4. A safe condition is that if either the brake or gas command is applied the other is set to zero.

2.8 Parking spot detection

Part of performing visual servoing control is the detection of the desired object in the image. The parking spot pose must be detected so it can be used within the closed loop control scheme in a visual servoing framework. To detect the parking spot, two approaches were investigated: a lane detection approach, and a learning-based approach.

2.8a Lane detection approach

Lane detection can be easily done through well-defined steps from literature starting with image un-distorting, following with cropping and resizing, transforming to grey scale, edge detection, Hough transformation, and finally filtering the parking spot lines. Those techniques will be used for parking spot detection and each step will be thoroughly explained.

The following image of figure 2.26 was taken using a webcam at the backyard parking lot at Marelli. And the detection algorithm will be demonstrated over this RGP image.

Figure 2. 26 Original RGB image of backyard at Marelli's parking yard

Image undistorting:

In this step, the distortion effects caused by the nature of the camera lenses are compensated for. This can be done through the function `undistortImage` from MATLAB which takes as input the distorted image along with camera parameters that include distortion parameters. Distortion parameters are acquired through the calibration process as well as other camera parameters. In our case, the acquired radial and tangential distortion parameters are shown in figure 2.27.

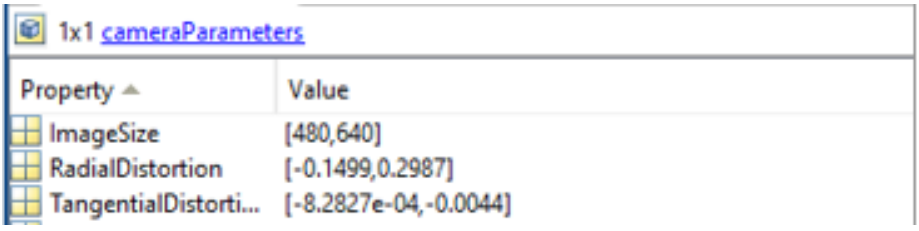

Figure 2. 27 Distortion parameters taken from MATLAB calibration app.

Cropping and resizing:

Image is cropped using the command `imcrop` which requires the original image as an input and the corners to be cropped at as attributes.

Grey scale transformation:

This is simply done using the command `rgb2gray` which turns the RBG image into a greyscale image that is equivalent to a 2-dimensional matrix with values in the range of 0-255. The function rgb2gray converts RGB values to grayscale values by forming a weighted sum of the *R*, *G*, and *B* components:

$$0.2989 * R + 0.5870 * G + 0.1140 * B$$

Edge detection:

To detect the edges within the image, the greyscale image is processed using Canny method. It works in a multi-stage process [22].

- Smoothing the image using the Gaussian convolution
- 2-D first derivative operator e.g. Robert Cross
- Edges give rise to ridges in the gradient magnitude image
- Algorithm tracks along the top of these ridges and sets to zeros all ridges that are not on the ridge top to give a thin line in the output
- The tracking algorithm is the non-maximal suppression
- Finally, the tracking process exhibits hysteresis over two thresholds to avoid breaking down "noisy" edges.

Hough transform:

Hough transform is a feature extraction technique based on the standard Hough transform to detect objects within a certain class of shape (lines in our case) by a voting procedure. The Standard Hough Transform (SHT) uses the parametric representation of a line:

$$\text{rho} = x*\cos(\text{theta}) + y*\sin(\text{theta})$$

rho is the distance from the origin to the line along a vector perpendicular to the line. *theta* is the angle of the perpendicular projection from the origin to the line measured in degrees clockwise

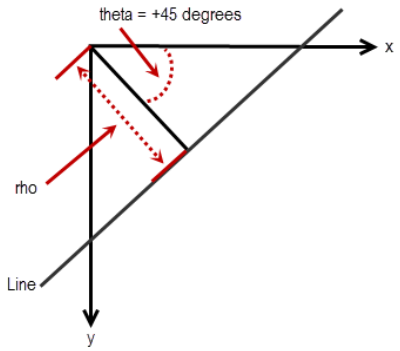

Figure 2. 28 Standard Hough transform line parameterization

from the positive x-axis. The range of *theta* is $-90° \leq theta < 90°$. The angle of the line itself is $\theta + 90°$, also measured clockwise with respect to the positive x-axis. The parameters of a line are demonstrated in figure2.29

Line filtering

After acquiring lines through the Hough transform method, lines are filtered such that

- Only lines oriented towards the center are considered
- Lines in the middle of the image within a defined threshold are excluded
- Lines with slopes in range of $[-30°, 30°]$ are disregarded
- The closest lines to the center from both left and right sides of the image are chosen.
- The starting point of the lines must be in the lower quarter of the cropped image
- Finally, the filtering process leaves out two lines as the side lines of the parking spot as shown in figure 2.29.

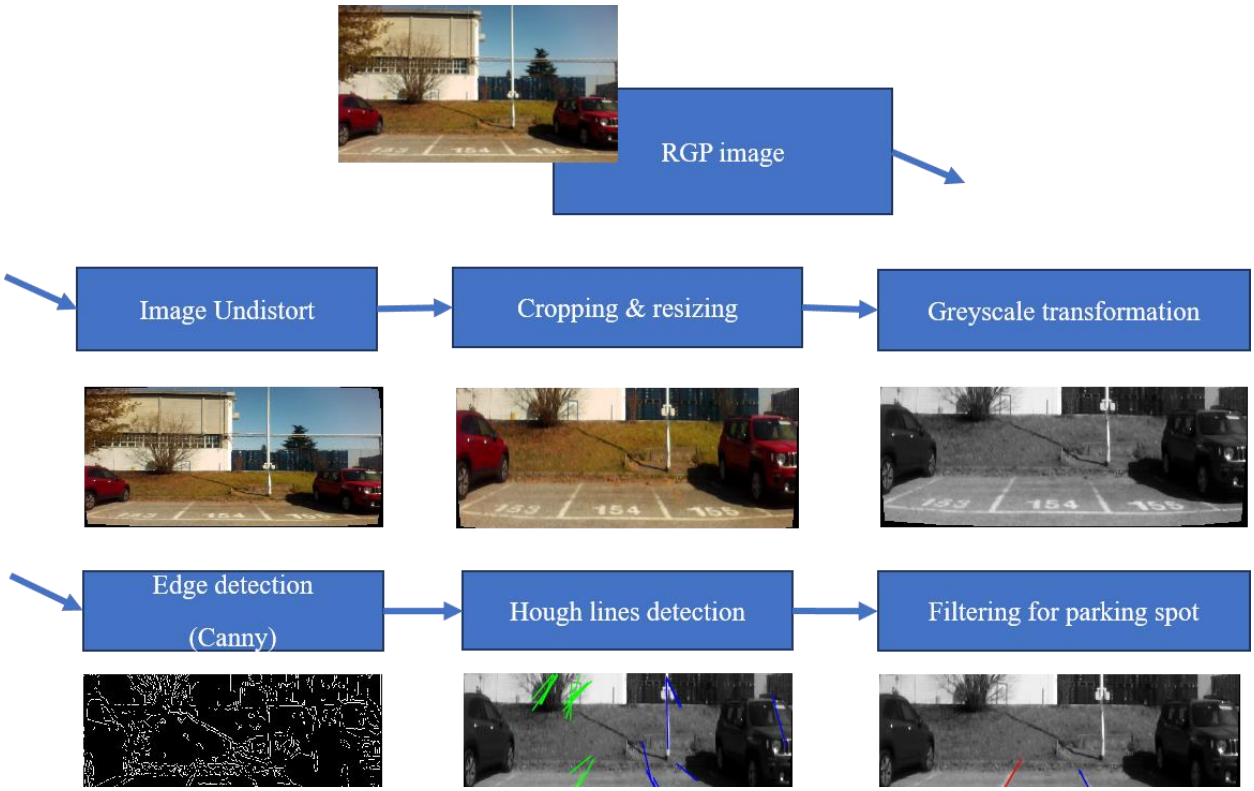

Figure 2. 29 Parking spot detection overall diagram

Center line construction:

After having detected the parking spot side lines, constructing the bisecting line would help knowing how close the vehicle is with respect to the center line plotted in yellow see figure 2.30.

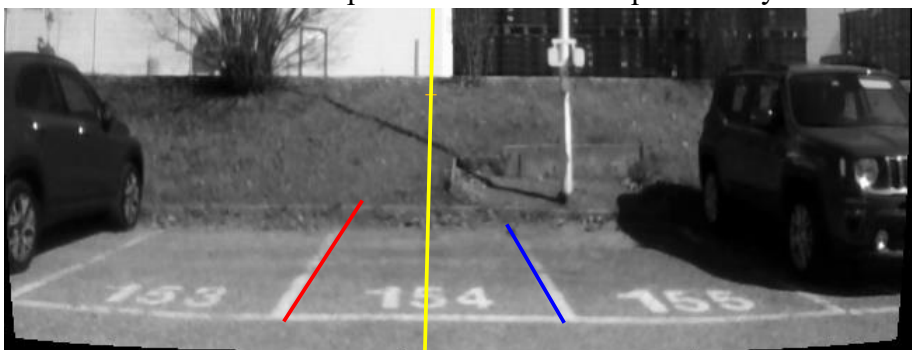

Figure 2. 30 Center line overlain over the undistorted image along with the parking spot sidelines

2.8b Parking spot detection: Learning based approach

In this approach [2] 8600 images were acquired as dataset in which entry points of the parking spots were manually labelled. Researchers also provide a tool with a GUI to label new training data. For this approach, a top view of the image of the parking spot has to be available, while physically it is not possible to mount a camera that is directly pointed towards the ground, the equivalent top view image can be acquired with some image processing over the original image known as the bird's eye view image. Figure 2.31 shows both the offline learning classifier among with the procedure for online detection.

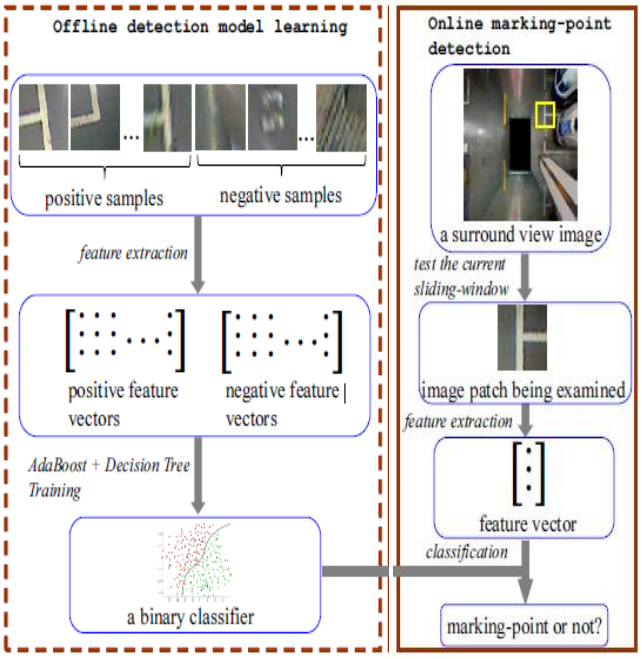

Figure 2. 31 A learning based approach for parking spot corners detection

The image which includes the parking spot is essentially processed and the following features are looked for in order to classify the parking spot, those features are:

- Normalized intensity
- Gradient magnitude (Sobel)
- Oriented gradient magnitude

The criteria basically rely on the top view image acquired from around vehicle view camera system. Four images taken from fisheye cameras are finally composed into a top view image.

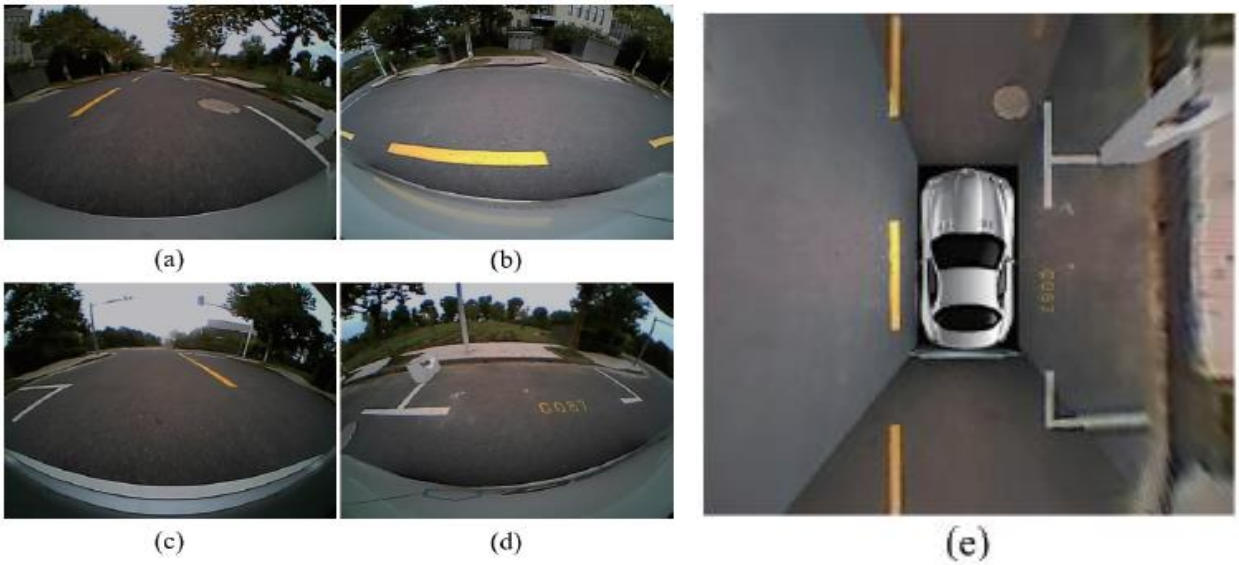

Figure 2. 32 Around the vehicle images a, b, c & d are grouped into a single top view in image e

Finally, this approach can deal with variety of situations shown in figure2.33.

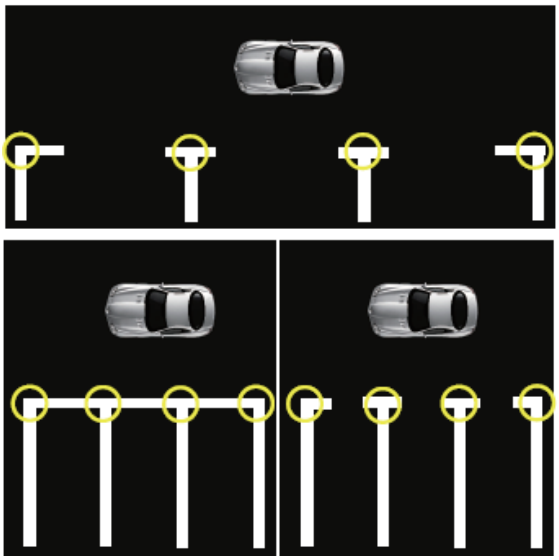

Figure 2. 33 Parking spot shapes that can be perceived through the learning-based approach

2.9 Chapter summary

Throughout this chapter, a complete description of the simulation environment setup was discussed including those parts of perception, camera modeling, controller design and plant model. Both proportional controller and the optimization control law were presented. The detection techniques of the parking spot were suggested as detecting the parking spot is an essential part during a real-life scenario, those techniques have not been applied within simulation to avoid unnecessary complexity.

Chapter Three: Results and Discussion

1. **Final parking maneuver: Collection of simulation results**
2. **Perpendicular parking with side and angle misalignment**
3. **Permissible parking area analysis**
4. **Permutation in camera model effects**
5. **Calibration error analysis**
6. **Perpendicular parking: L curve**
7. **Perpendicular parking from a wide angle**
8. **Carsim simulation results**
9. **Parking spot detection results**

3.1 Final parking maneuver: acquiring simulation results

In this chapter, the results of several simulation experiments are presented to demonstrate the performance of the autonomous final parking maneuver. The simulations are done for both control techniques: cascade proportional and optimization-based control law. For all the scenarios, the evolving motion of the parking situation is presented, along with velocity and steering angle curves. The final relative errors in the cartesian space are calculated, as well as the weighting functions and plucker features errors for the case of optimization-based control. Similarly, evolving motion, velocity and steering angle profiles are presented for the model validation case using a Carsim model. Finally, successful detections of real parking spots are demonstrated.

3.2 Perpendicular parking with side and angle misalignment

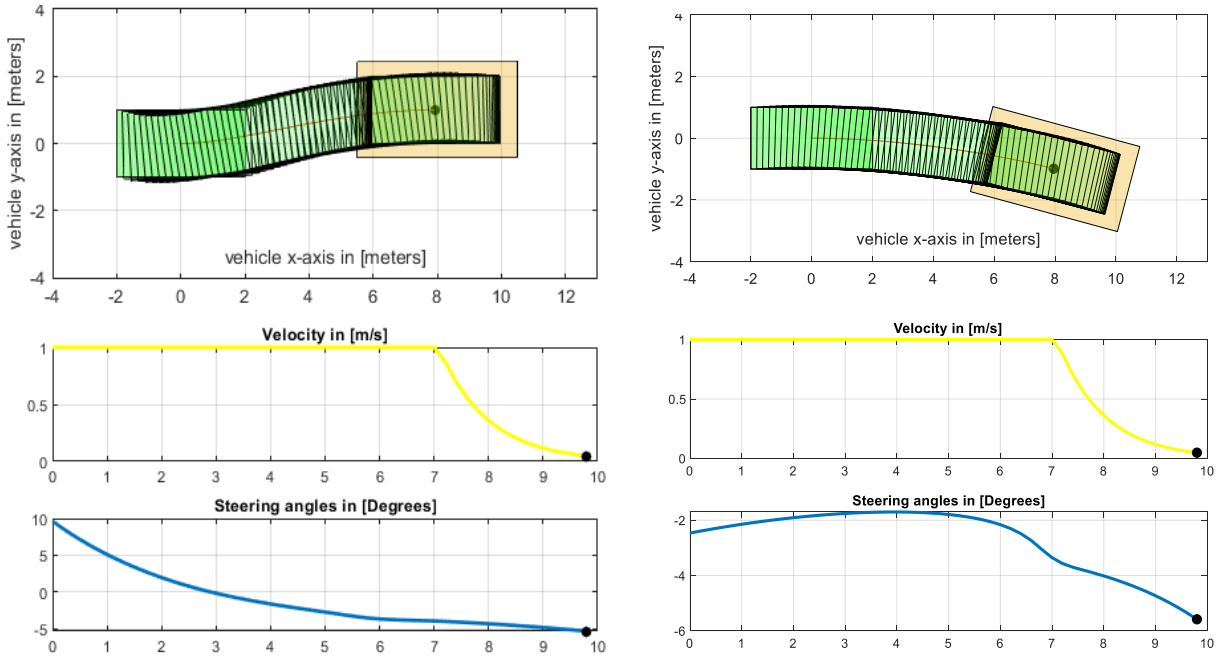

Figure 3. 1 Evolution of autonomous parking and relevant velocity and steering angle profiles with horizontal misalignment (left), and an angle misalignment(right)

Figure 3.1 shows two cases of autonomous parking where in both cases the proportional controller is used:

1. Parking spot is on a side misalignment with a relative cartesian pose of $[8, 1, 0°]$ with a final relative error in the planar cartesian space of $[0.0480 \quad 0.0003 \quad -0.6401°]$
2. Parking spot is on an angle misalignment with a relative cartesian pose of $[8, -1, -15°]$ and a final relative error in the planar cartesian space of $[0.0468 \quad -0.0124 \quad -0.3048°]$

In both cases, the velocity profiles start at a constant value and as the vehicle comes close to the parking spot, the velocity declines to zero. This is because the velocity command is proportional to the distance between the vehicles pose and the parking spot pose. While figure 3.1 presents the results of only two parking cases that use the cascade controller, an extensive experimental

procedure was done to find out which is the area that we can start parking from without hitting the limits of the parking spot shown in the following section.

3.3 Permissible parking area analysis

In order to know which is the area that it is possible to start parking from without hitting the side limits , simulation experiments were done by incrementally making side offsets of the vehicles initial position and observing when the vehicle would surpass the side limits, those points are all concluded within one area as shown in figure 3.2. This area was acquired while applying the proportional controller.

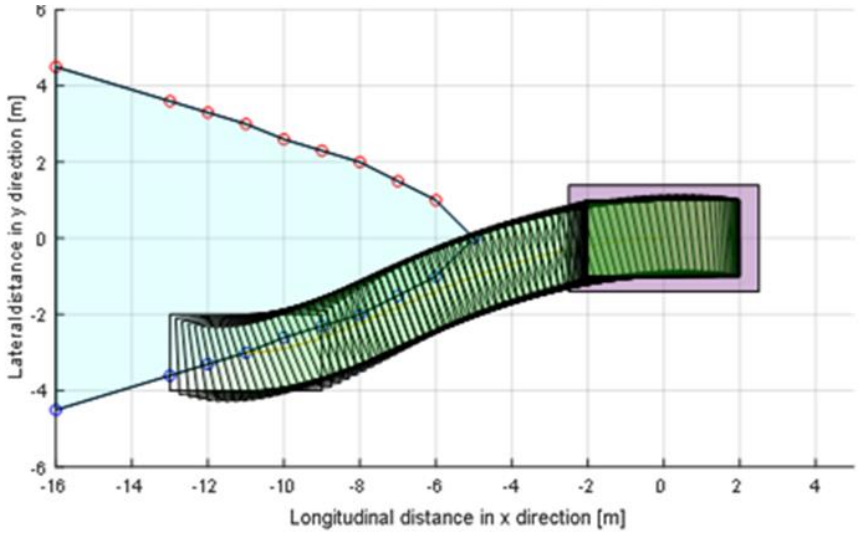

Figure 3. 2 Permissible area to initiate parking from on a horizontal misalignment

Figure 3.3 shows that the permissible parking area narrows down when the parking spot and the vehicle are misaligned by 10°. The area is considered permissible such that allows performing the parking maneuver without surpassing the parking spot limits.

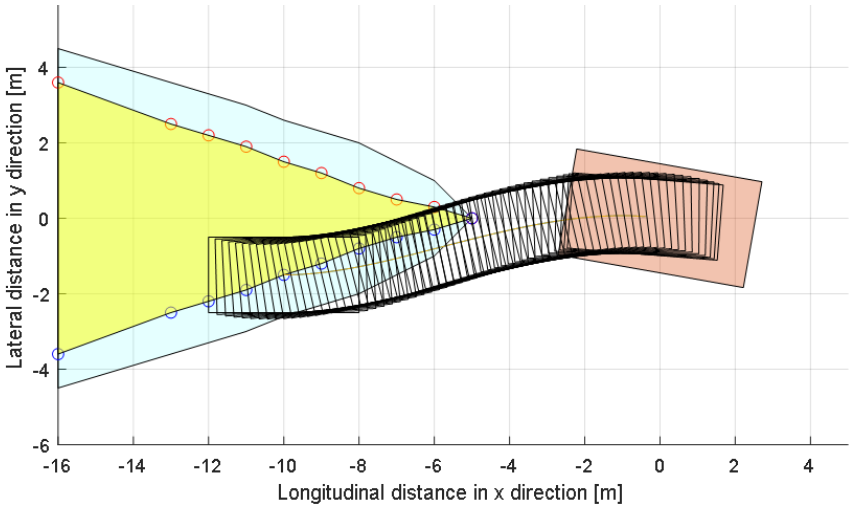

Figure 3. 3 Permissible area to initiate parking from when an angular misalignment is present

3.4 Permutation in camera model effects

To analyze the effect of changing in the camera parameters, the model of the camera within the estimation module is modified such that camera is modeled with extrinsic angle of 17° instead of 14°. Figure 3.4 shows cartesian measurements of the vehicle motion both ideally and estimated. To start with, there shows a clear offset of 4 m in the lateral direction at the beginning. While ideally the distance is 8 m, the distance is estimated to be 12 m.

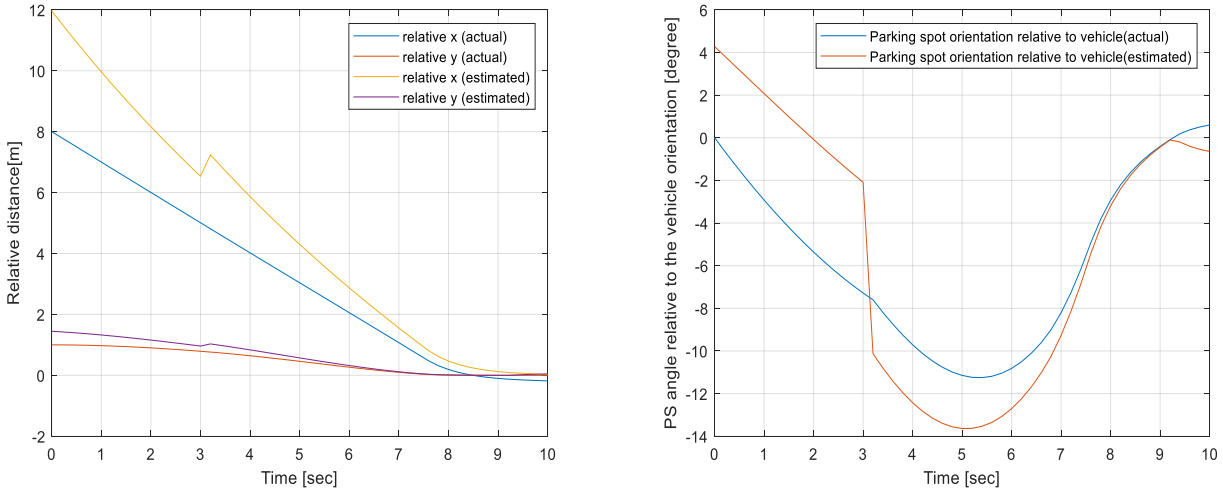

Figure 3. 4 Variating camera model parameters effect over position measurements(left), and yaw angle measurements(right)

For the angle measurements, there shows a clear offset of around 4° in the first 3 seconds, and the error diminishes as the vehicle moves closer to the target spot.

To visualize the effect of this modification in camera model, the evolution of motion is recorded in figure 3.5. The ideal case shows no surpassing the limits of the parking spot, while in the case of modified parameters there is a slight hitting of the parking limits (see inside figure 3.5 right).

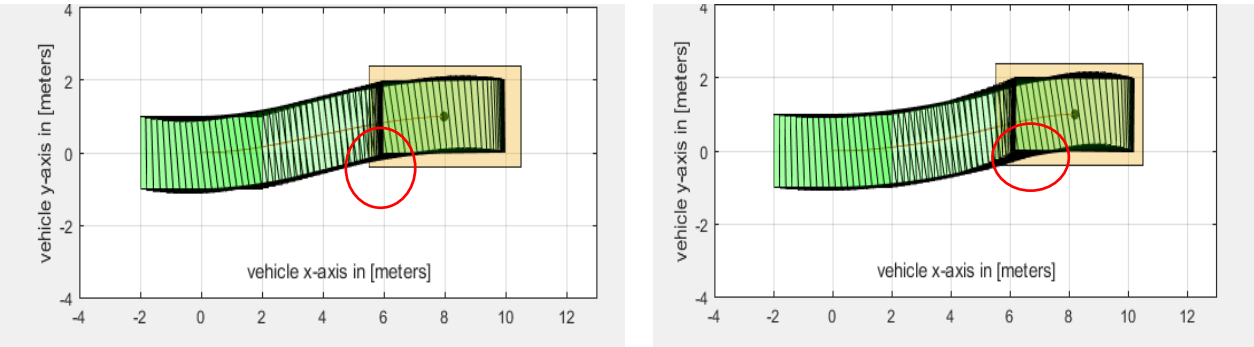

Figure 3. 5 Variating camera parameters effect over vehicle evolution towards the parking spot, ideal case (left) variating case(right)

3.5 Calibration errors analysis

To estimate camera parameters, 20 pictures of the checkerboard were taken from a webcam, 3 of them were excluded leaving 17 pictures to perform the calibration process. For each picture, the detection of checkerboard corners was performed with a relative accuracy of $0.2 - 0.7$ pixels. The overall detection error is roughly 0.39 pixels presented in figure 3.6. This means the calibration parameters are estimated such that if the pixels are moved 0.39 pixel from the original location. The reflection of those results over our criteria is that, the pose of the parking spot is estimated

with an average error of 0.39 pixels. This translates to a negligible error in meters specially when very close to the parking spot.

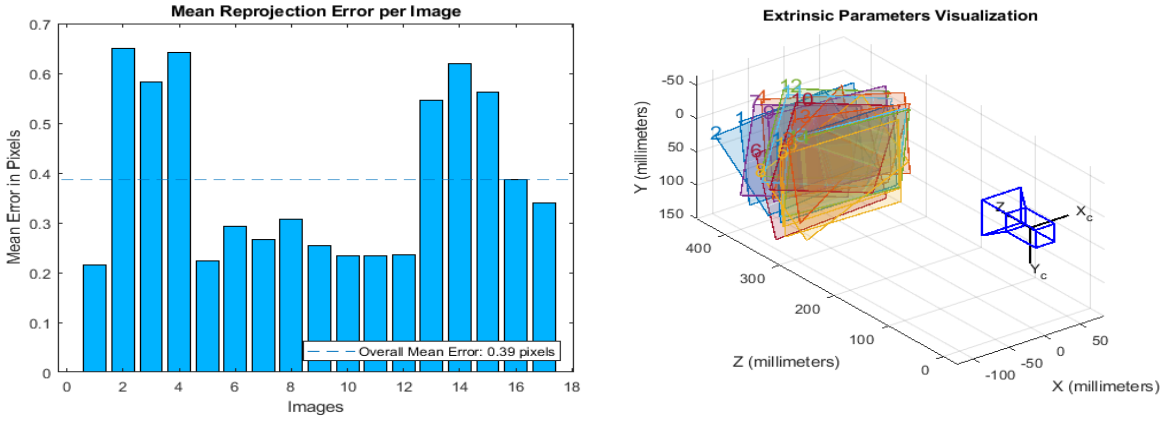

Figure 3. 6 Mean reprojection error of each calibration image (left), pose of each image used for calibration(right)

3.6 Perpendicular parking: L curve

Figure 3.7 shows the parking case with an initial relative cartesian pose of $[4 \ -6 \ -90°]$. The evolving motion and velocity profiles are demonstrated for both controllers with a final relative error as follows:

1. Optimal control law error in the planar cartesian space: $[0.0804 \quad -0.0416 \quad -0.2200°]$
2. Cascade controller error in the planar cartesian space: $[0.0010 \quad -0.0495 \quad -1.8369°]$

In the case of optimal control, the controller is tuned such that the vehicle does not surpass the parking spot limit, while in the proportional control, the vehicle surpasses the limits of the parking spot.

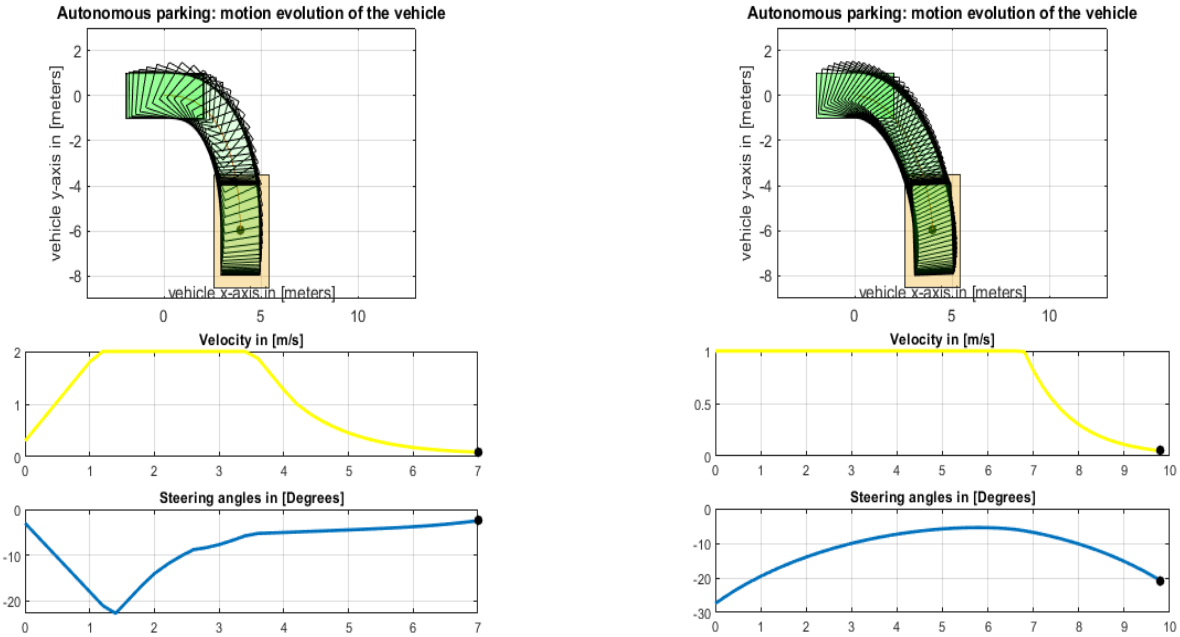

Figure 3. 7 Motion evolution, velocity and steering angle, optimization-based controller (left), cascade controller (right).

For the proportional control case, the steering angle starts from around $-30°$ which means the vehicle must steer before starting to move, this can affect the steering column and tear the wheels quickly. For the optimal control case, the steering evolves from around zero degrees.

For the optimal control case, the velocity evolves from zero, then stays on a constant level, and finally declines to zero. Consequently, it is possible to follow such a curve in the real case scenario as it does not demand a fast response. Using the proportional controller, the velocity command starts at $1m/s$ and then once close enough to the parking spot it starts declining to zero. In a real case scenario, there will be a delay in which the longitudinal velocity responds to such a command as a result to vehicle dynamics. Such a delay will also be observed over the Carsim model. When it comes to the cascade controller, the velocity command starts at a saturated value because it is proportional to the distance relative to the parking spot, so it is at its maximum when the vehicle starts to move towards the spot.

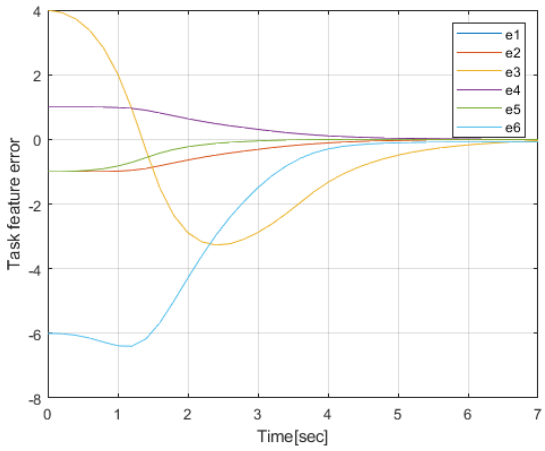

Figure 3. 8 Error of the plucker coordinates task features for the L curve perpendicular parking

Figure 3.8 shows the task feature errors, those are the difference between desired normalized plucker coordinates and estimated plucker coordinates of the two task feature lines as explained from the control criteria in chapter two. Essentially, we have six task features of the two lines and

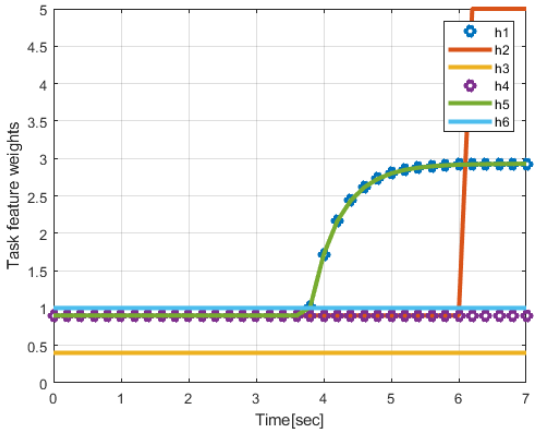

Figure 3. 9 Weighting variables for the optimization-based control

respectively, errors $e1, e2, e3$ represent the error of line $\mathcal{L}_1$, while $e4, e5, e6$ represent the errors of line $\mathcal{L}_2$ features. $\mathcal{L}_1, and\ \mathcal{L}_2$ are shown in figure 2.17.

Figure 3.9 shows how the weighting variables evolve when the vehicle is moving toward the parking area. Weights h3 and h6 are constants in all cases, while other weights are designed using smooth weighting function explained in the control method in chapter two.

3.7 Perpendicular parking from a wide angle

As the previous case of an L curve, results lack the use of the camera model because the spot cannot be seen by the camera from that distance, another case is considered where the parking spot is closer to the vehicle and two corner points can be seen by camera at all times. Results of both proportional and optimization-based control are shown. Weighting variables and task error profiles are shown for the optimization-based controller.

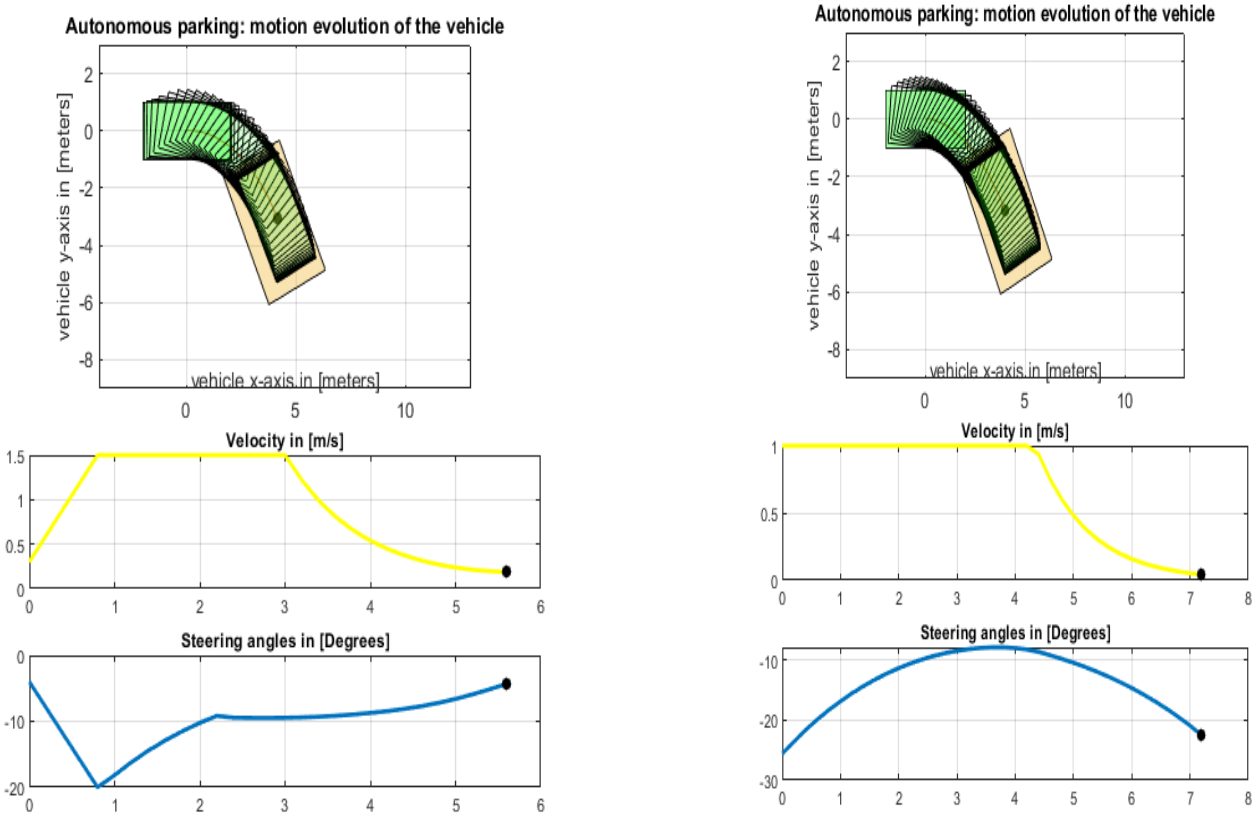

Figure 3. 10 Motion evolution, velocity and steering angle, optimization-based controller(left), cascade controller(right).

Figure 3.10 shows the parking case of an initial relative cartesian pose of $[4\ -3.2\ -65°]$. The evolving motion and velocity profiles are demonstrated and eventually, the final error for both cases is recorded:

1. Optimal control final error in the planar cartesian space: $[-0.1473\ -0.1283\ -0.8113°]$
2. Cascade controller final error in the planar cartesian space: $[0.0180\ -0.0371\ -1.4404°]$

Figure 3.11 shows the task feature errors, those are the difference between desired normalized plucker coordinates and estimated plucker coordinates of the two task lines. As explained from the control criteria, we have six task features which leads to errors shown in figure 3.11 $e1, e2, e3$ represents the errors of $\mathcal{L}_1$ features, while $e4, e5, e6$ represent the errors of $\mathcal{L}_2$ features.

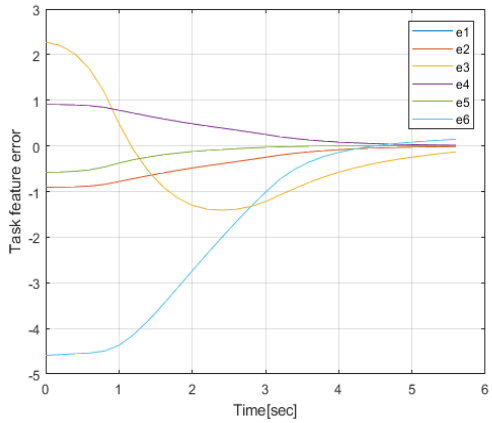

Figure 3. 11 . Error of the plucker coordinates task features for wide angle perpendicular parking

Figure 3.12 shows how the weights evolve when the vehicle is moving toward the parking area. Weights h3, and h6 are constants in all cases, while other weights are made using smooth weighting function explained in the control method.

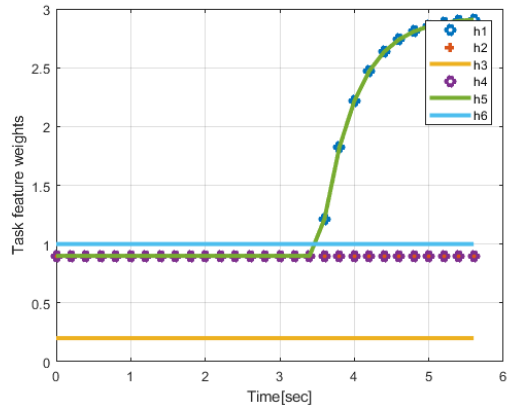

Figure 3. 12 Weighting variables for the optimization-based control

3.8 CARSIM simulation results

Cascade controller is applied over Carsim model to verify the feasibility of using the single-track kinematic model when designing the controller. The following two cases are simulated.

3.8a Case one: Horizontal misalignment

With an initial relative cartesian pose of $[8, 2, 0°]$ as shown in figure 3.13, a final error in the planar cartesian space is recorded: $[0.0499 \quad 0.0019 \quad -5.4406°]$. Results are taken by applying the proportional controller over the CARSIM model. Figure 3.13 shows a snapshot taken from Carsim that includes the outputs profiles of velocity and steering angles both of driver and front wheels.

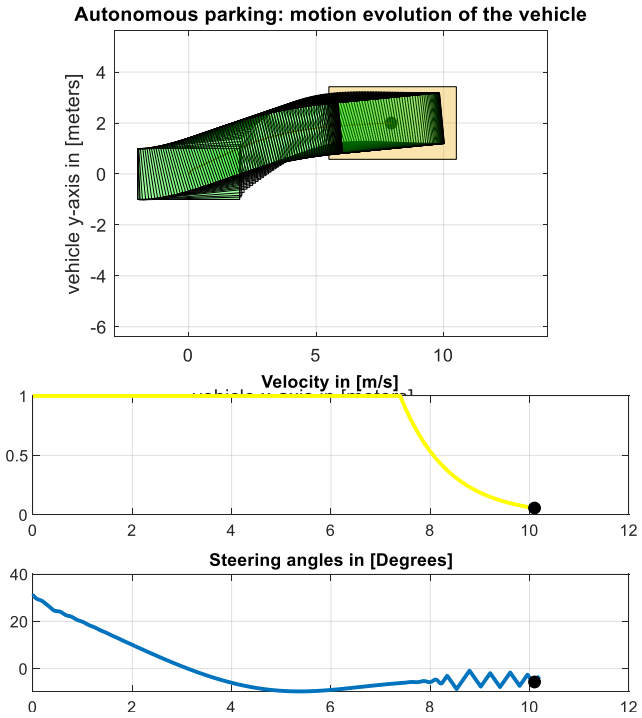

Figure 3. 13 Evolution of motion for the Carsim model parking case, as well as the velocity and steering wheels angle commands.

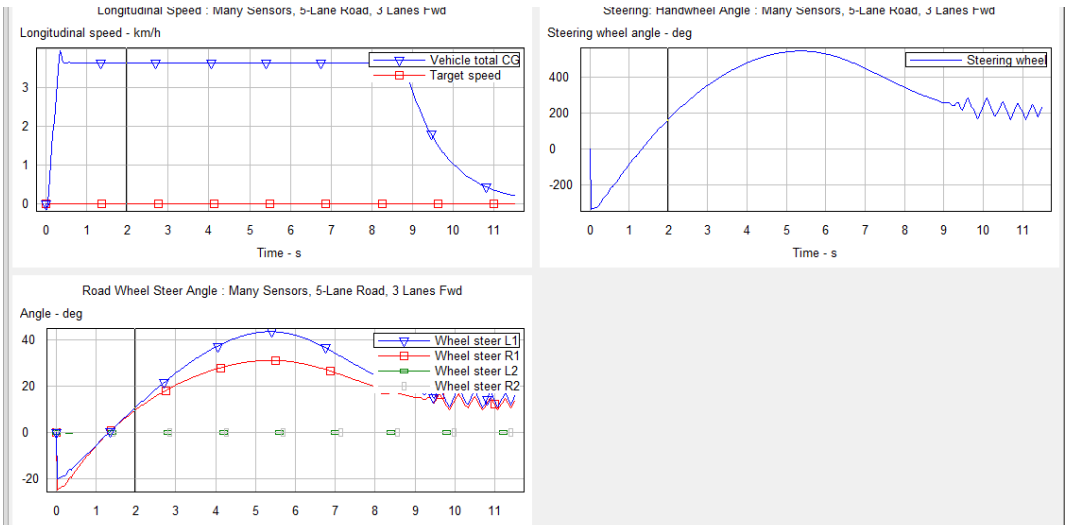

Figure 3. 14 Carsim plots: Measured longitudinal velocity (upper left), drivers steering wheel angle(right), front wheels steering angle (lower left)

The measured longitudinal velocity follows the step command and reaches a steady state value at 0.5 sec. Such a behavior shows the dynamics effect of the vehicle model.

3.8b Case two: L turn parking

With an initial relative cartesian pose of $[6, 6, 90°]$ as shown in figure 3.15, a final error is recorded: $[0.0052 \quad 0.0497 \quad 14.7667°]$. Results are taken by applying the proportional controller over the CARSIM model. Figure shows a snapshot taken from Carsim that includes the outputs profiles of velocity and steering angles both of driver and front wheels. The measured longitudinal velocity follows the step command and reaches a steady state value at 0.5 sec. Such a behavior shows the dynamics effect of the vehicle model, which eventually translates into a final heading error of 14°

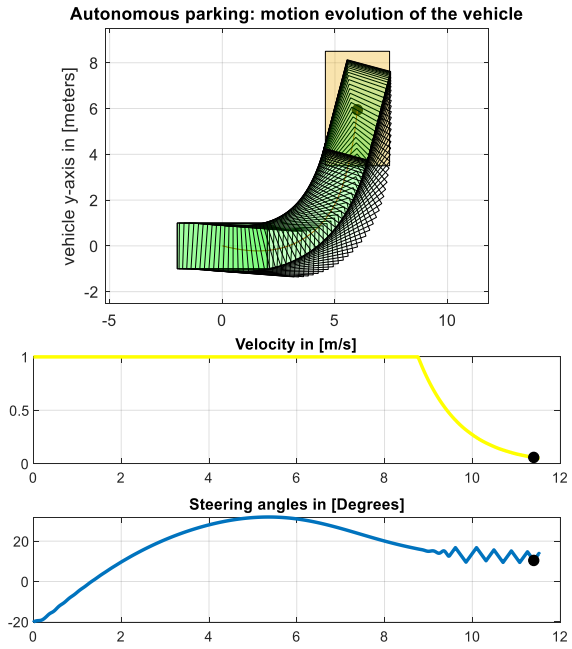

Figure 3. 15 Evolution of motion for the Carsim model parking case, as well as the velocity and steering wheels angle commands.

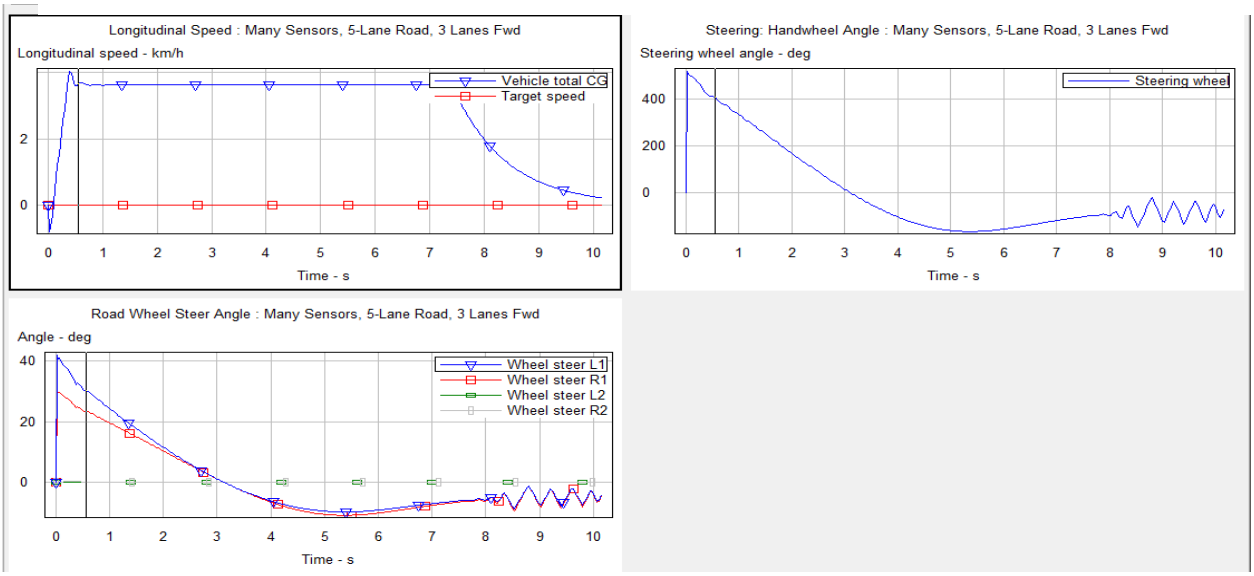

Figure 3. 16 Carsim plots: Measured longitudinal velocity (upper left), drivers steering wheel angle(right), front wheels steering angle (lower left)

3.9 Parking spot detection:

3.9a Lane detection approach:

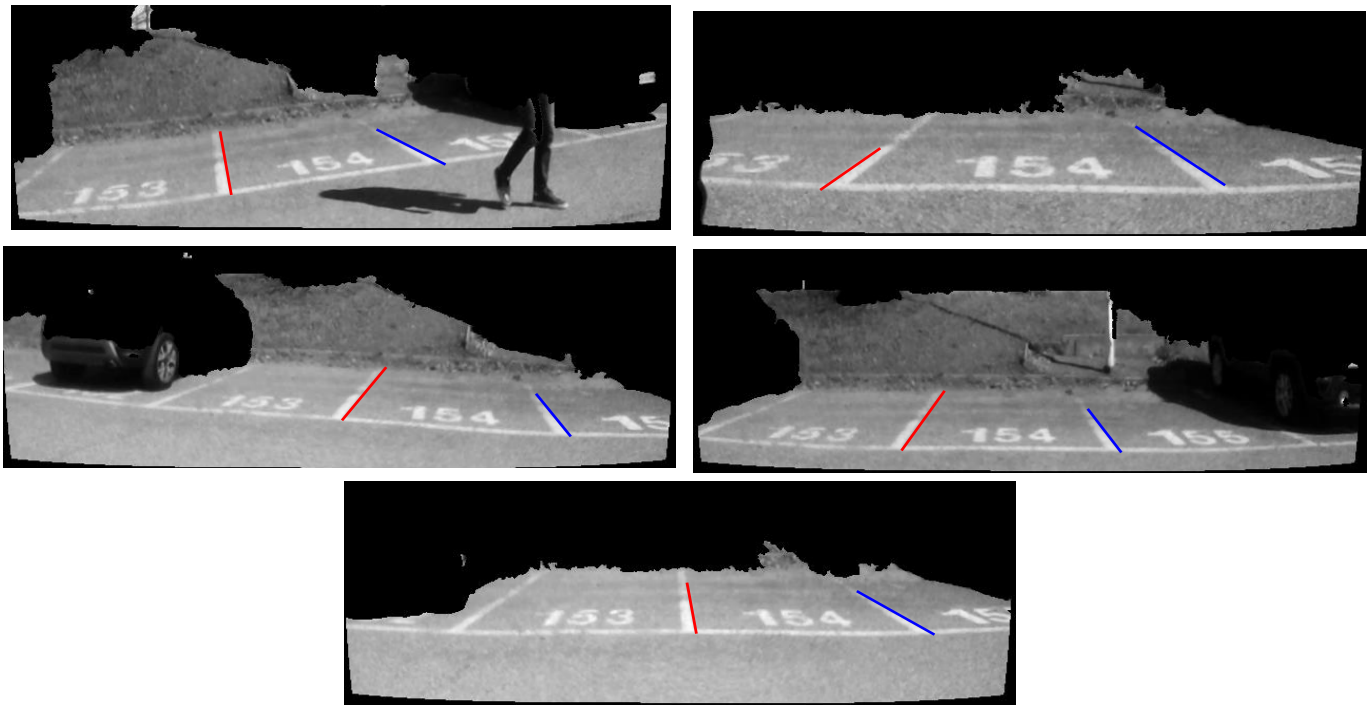

Figure 3. 17 Successful parking lines detection cases within the lane detection approach framework

The lane detection approach is tested with parking images from the backyard parking lot at Marelli see figure 3.17. While good detection results for most of the cases are recorded, the detection failed when the vehicle is too close to the parking spot or largely misaligned which suggests further development for this approach.

3.9b Learning based approach:

Figure 3.18 shows the result of detecting a parking spot by its entry points and then overlay it with markers. For this appraoch, the learning algorithm was taken without adding more pictures of training sets than originally provided.

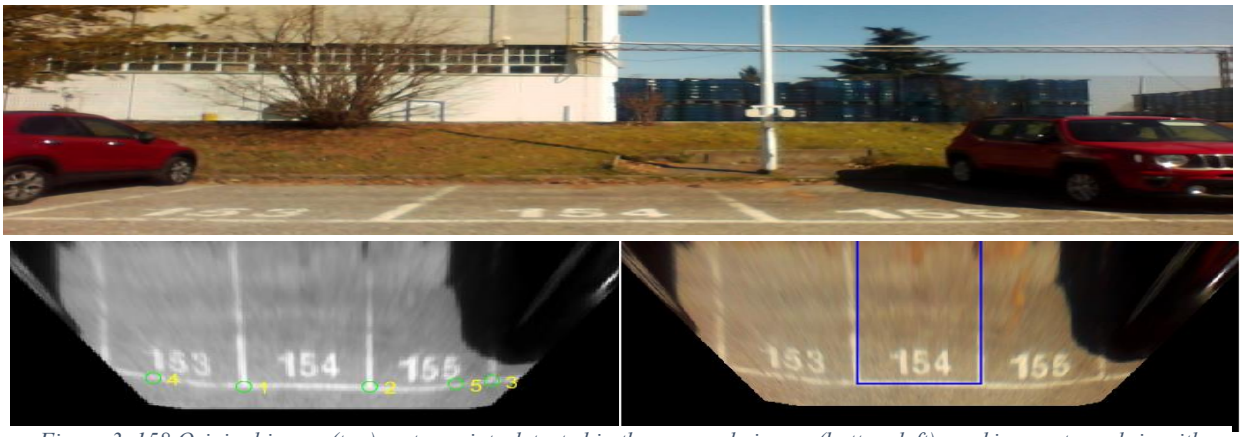

Figure 3. 158 Original image (top), entry points detected in the greyscale image (bottom left), parking spot overlain with markers (bottom right)

Conclusions

After doing several experiments over both the cascade and optimization control laws, it is concluded that in a wide angle scenario such as the L turn, the optimization control law performs better than the cascade; control and weighting parameters be tuned so it does not surpass the parking spot limits. There is also the feature of adding constraints to the control model over the input control quantities and the physical limits of the parking spot.

Both control techniques that were applied are positional based visual servo control techniques (PBVS). No image-based visual servoing (IBVS) control techniques were applied for the reasons discussed in chapter one as the vehicle is an underactuated plant, and that IBVS requires further pose estimation.

For the optimization-based control technique, it was only possible to acquire results for the L-curve parking and the wide-angle case. Cases in which the parking spot was on a mere misalignment did not show good results. The original research suggests that for such cases, the task features lines must be chosen differently, and the approach must be re-designed.

Cascade proportional control has shown a strong robustness to change in starting area towards the parking spot and did not require any further tuning over control gains. While the cascade approach requires tuning 3 control parameters, the optimization-based law requires tuning 6 control parameters, and 10 other weighting parameters, those parameters must be tuned again if the initial pose of the vehicle is changed. The optimal control algorithm requires 0.02 sec to be calculated, while the cascade computation time is negligible.

The cascade control technique was further tested in Carsim, and it performed similarly to ones applied over the single-track kinematic model, while the controller had required further tuning. The steering model of carsim is not a single wheel model as in the single-track kinematic model, the front wheel has two slightly different steering angles. Simulations over carsim showed an increase in the heading error of around 4° over the case of side misalignment and 14° heading error over the L turn case.

In conclusion, the positional based visual servoing approach is used and further two control techniques are applied from which the proportional control is preferred over the optimization based one for our typical parking scenario; as cascade controller shows robustness towards changing the initial position and more importantly, works well in all typical cases unlike the optimization based control which requires a total re-design and different choice of task features when changing the parking scenario to other than the L turn case.

References

[1] P. Muehlfellner, P. Furgale, W. Derendarz and R. Philippsen, "Evaluation of Fisheye-Camera Based Visual Multi-Session Localization in a Real-World Scenario.," in *2013 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*, 2013.

[2] L. Li, L. Zhang, X. Li, X. Liu, Y. Shen and L. Xiong, "Vision-Based Parking-Slot Detection: A Benchmark and A Learning-Based Approach," in *2017 IEEE International Conference on Multimedia and Expo (ICME)*, 2017.

[3] L. Qian, L. Chunyu and Z. Yao, "Geometric Features-Based Parking Slot Detection," *Sensors,* 2018.

[4] N. Dario, P. Giulio, Z. Pierluigi and S. M.Savaresi, "A two-wheeled vehicle oriented lane detection algorithm," in *2018 IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[5] L. Linshen, Z. Lin, L. Xiyuan, L. Xiao, S. Ying and X. Lu, "Vision-Based Parking-Slot Detection:A DCNN-Based Approach and a Large-Scale Benchmark Dataset," *IEEE Transactions on Image Processing,* 2018.

[6] A. Hillel, R. Lerner, D. Levi and G. Raz, "Recent progress in road and lane detection: a survey," *Machine Vision and Applications,* 2011.

[7] "http://ksimek.github.io/2012/08/22/extrinsic/," [Online].

[8] C. Francois, "Visual Servoing," *Computer Vision: A Reference Guide,* 2014.

[9] Y. Masutani, M. Mikawa, N. Maru and F. Miyazaki, "Visual Servoing for Non-Holonomic Mobile Robots," in *2001 European Control Conference (ECC)*, 2001.

[10] A. Cherubini, F. Chaumette and G. Oriolo, "An Image-based Visual Servoing Scheme for Following Paths with Nonholonomic Mobile Robots," in *10th International Conference on Control, Automation, Robotics and Vision*, 2008.

[11] J.-S. Farrokh, "Visual Servoing: Theory and Applications," in *Opto-Mechatronic Systems Handbook: Techniques and Applications*, 2002.

[12] S. Bruno and K. Oussama, Handbook of Robotics, Springer, 2008.

[13] H. Lang, M. Khan, K.-K. Tan and C. d. Silva, "Application of Visual Servo Control in Autonomous Mobile Rescue Robots," *International Journal Of Computers Communication & Controler,* 2016.

[14] W. Wang, Y. SONG, J. ZHANG and H. DENG, "Automatic Parking of Vehicles: A Review of Literature," *International Journal of Automotive Technology,* 2014.

[15] S. Yuyu and L. Chenglin, "Analysis and Review of State-of-the-Art Automatic Parking Assist System," in *2016 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, 2016.

[16] "https://cvg.ethz.ch/research/v-charge/," [Online].

[17] "https://venturebeat.com/2019/07/23/bosch-and-daimler-obtain-regulatory-approval-for-level-4-self-parking-system/," [Online].

[18] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 2000.

[19] D. Pérez-Morales, O. Kermorgant, S. Domínguez-Quijada and P. Martinet, "Laser-Based Control Law For Autonomous Parallel And Perpendicular Parking," in *International Conference on Robotic Computing,*, Laguna Hills, United States., 2018.

[20] R. H. Byrd, J. C. Gilbert and J. Nocedal, "A Trust Region Method Based on Interior Point Techniques for Nonlinear Programming," INRIA, 1996.

[21] A. Trotta, "Thesis: Trajectory planner for autonomous vehicles," Politecnico di Torino, Turin, 2016.

[22] "http://homepages.inf.ed.ac.uk/rbf/HIPR2/canny.htm," [Online].

[23] U. Ozgunalp, R. Fan, X. Ai and N. Dahnoun, "Multiple Lane Detection Algorithm Based on Novel Dense Vanishing Point Estimation," *IEEE Transactions on Intelligent Transportation Systems,* 2017.