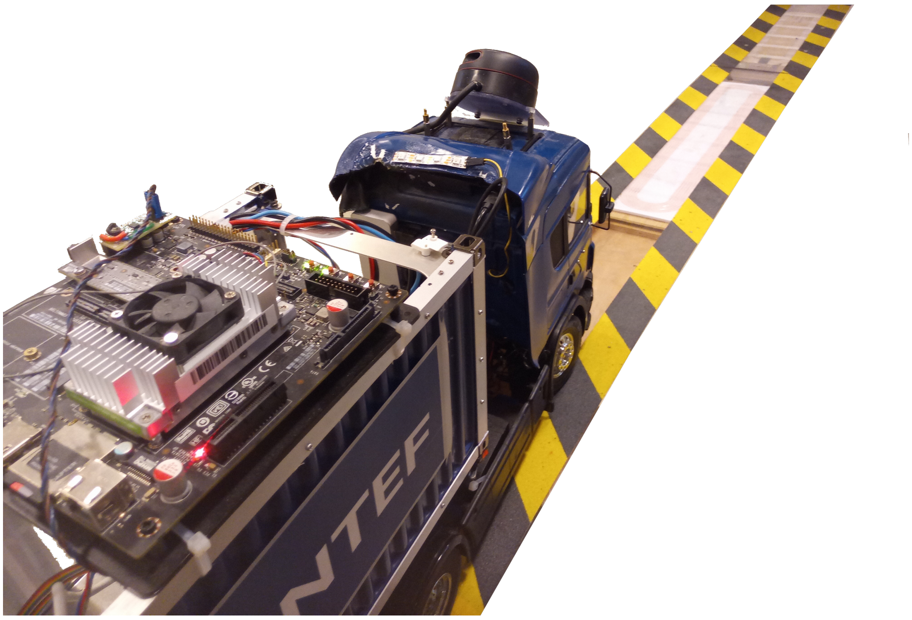
Master's thesis

Alaa Khalaf

Optimization of transfer efficiency of a dynamic wireless charging system for electric vehicles

Master's thesis in embedded computer systems
Supervisor: Jon Are Suul
July 2020

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics



NTNU
Norwegian University of
Science and Technology

Optimization of transfer efficiency of a dynamic wireless charging system for electric vehicles

Alaa Khalaf

Supervisor: Jon Are Suul

July 6, 2020

Preface

Autonomous vehicles with solutions for dynamic on-road wireless power transfer have the potential to be fully autonomous in terms of operation and energy supply. The purpose of this Master Thesis was to study self-driving strategies for controlling the position and to manage the interaction with the control of a wireless charging system when the vehicles is passing a road section with wireless power transfer capability. For this purpose, an existing small-scale autonomous truck model equipped with a wireless dynamic charging system was used as the test platform. The objective was to develop methods to maximize the energy transfer efficiency from the road-side infrastructure to the on- board batteries. Two complementing methodologies were pursued for obtaining this objective. The first was to fine-tune the driving controls of the existing self-driving system in order to control the alignment between the coils in the vehicle and in the road. The second approach was to introduce communication and coordination between the self- driving system and the power transfer system by remotely controlling the road-side infrastructure.

Abstract

A scaled down truck model equipped with a wireless dynamic charging system was the main platform for development, in this study. The goal was to enhance the efficiency of the power transfer from the road-way coil to the truck's on-board pick-up coil. This was done by following two methodologies; the first was to fine tune the driving controls through an autonomous driving system, based on classic computer vision algorithms, in order to maintain a close to perfect alignment between the two coils. The second methodology was to establish a real-time communication link between the truck and the roadway controller; in order to synchronize the triggering of the road-way coil to minimize the energy lost when the truck is not passing over the road-way charger. The truck is equipped by an Nvidia Jetson Tx2 board that handles the autonomous driving, and a Zynq board controls the charging of the truck's battery by rectifying and regulating the wirelessly received power.On the sending side, the roadway unit is controlled by another Zynq board, which was connected through bluetooth to the truck's Jetson board. The BT link enables the truck to trigger the charger on/off in real-time. Two different triggering scenarios were tested; to quantify the parasitic idling losses and to illustrate the utilization of the established real-time communication in minimizing such losses. The development platform for the truck's autopilot was ROS (robot operating system), which is a powerful middleware framework that runs on Linux based operating systems. As for the roadway unit, the Xilinx Vivado design suite was used for development.

Contents

Figures

Tables

List of Acronyms

EV	electric vehicle
RPEV	road powered electric vehicle
OEM	original equipment manufacturer
WPT	wireless power transfer
IPTS	inductive power transfer system
PATH	"partners for advanced transit"
KAIST	Korean advanced institute of science and technology
OLEV	on-line electric vehicle
PWM	pulse width modulation
MOSFET	metal oxide silicon field effect transistor
SiC	silicon carbide
IGBT	insulated gate bi-polar transistor
FPGA	field programmable gate array
SAE	society of automotive engineers
ADC	analog to digital converter
OS	operating system
RFCOMM	radio frequency communication
SLAM	simultaneous localization and mapping
MPC	model predictive control

openCV	open computer vision
RGB	red-green-blue
ROS	robot operating system
Lidar	light detection and ranging
BT	Bluetooth
UART	universal asynchronous receiver transmitter
pu	per unit
FSM	Finite state machine

List of Symbols

V_x	DC voltage across battery/power-supply
v_x	normalized voltage
i_x	normalized current
k	normalized magnetic coupling coefficient
Q	electrical quality factor
R_x	resistance of coil x
r_x	normalized resistance
r_{xy}	equivalent normalized resistance
ω	LC resonance frequency
L_x	inductance of coil x
M	mutual coupling coefficient
C	capacitance
ℓ	vehicle wheel base
R	vehicle turning radius
δ	vehicle steering angle

Chapter 1

Introduction and motivation

1.1 Introduction

In the recent years, the electrification of road vehicles has witnessed important technological advancements, that significantly increased the market share for EVs (electric vehicles), and the number of large scale deployments of EV supporting infrastructure increased accordingly -in many countries around the world. This was made possible due to the extensive development in battery technologies that led to large scale production of relatively cheap batteries with reduced weight and extended capacities and lifetime. However, the cost of ownership of a fully electric vehicle or even a hybrid is still significantly higher than that of conventional ICE (internal combustion engines) vehicles, and the tax incentives offered by governments to buyers of electric vehicles are key factor in their decision to pay such high prices. These incentives -in some cases are even more important than the reduction in operating costs attributed to the fuel savings and reduction in maintenance costs of electric vehicles.

Relatively higher costs of ownership, however, is not the only challenge that faces the widespread of electric vehicles- especially fully electric. Long charging times and the risks that would accompany the use of fast chargers -relatively high DC voltages and currents that would require larger cables that are heavier and more difficult to handle manually- are still major setbacks of plug-in vehicles. In addition to this, the limited number of charging stations compared to gas/diesel stations further discourages consumers from considering a fully electric vehicle as an option. To face theses challenges OEMs and suppliers try to maximize the range of such vehicles and come up with advanced fast charging systems while avoiding increases in the battery pack size/weight, to curb the increase of their vehicles' prices.

A promising technology that would potentially solve the limited range and large battery size/cost for plug-in Evs, is RPEVs (road powered electric vehicles)- even though they still have their own technical and economic challenges. As the name implies, an RPEV gets its energy as it moves along the road either through conduction similar to a tram or through magnetic induction similar to high speed trains. This means that such vehicles could have infinite ranges with minimal onboard batteries, which would lead to significant reduction in ownership costs of such vehicles in the case of large scale adoption of road-way charging units.

This study focuses on an experimental wireless power transfer (WPT) system, designed to charge a scaled down model of an electric freight truck. In the case of a commercial electric truck, the costs of adding more batteries to extend the range is far more expensive than in the case of a passenger vehicle. This is due to the much higher energy demand for the truck, and also due to the direct limitations that would be imposed on the truck's useful loading capacity as the battery pack increases in size and weight.

1.2 Brief history of dynamic WPT systems for road vehicles

The concept of RPEVs stems from an old patent with the title :'Transformer system for electric railways' by M. Hustin and M. Leblan in France in 1894, where a large air gap transformer was placed under a train [1] for electric power transfer through magnetic induction, as shown in figure 1.1. [2]

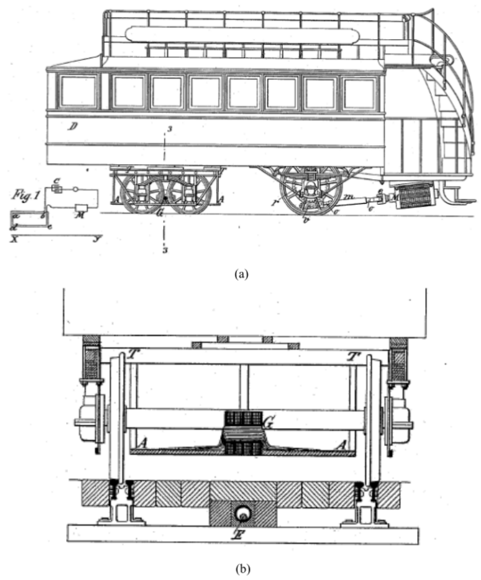


Figure 1.1: First patent of an RPEV in 1894. (a) Side view of the electric train. (b) Cross-sectional view cut on the line 3–3 of (a) [2].

In the patent, the authors claimed several design issues of inductive power transfer systems (IPTS) on the design of their power supply rails (coils), such as high power transfer to pick-up coils and reduction of conduction and eddy current losses; which are still important design issues [1].

The oil crisis in the early 1970s sparked shifting to high-way electrification [2] and the idea of wireless RPEVs resurfaced, specially with the advent in solid-state controllers for power electronics. Of the earliest well documented research projects of that time was Santa Barbra's electric bus project, which was undertaken by the Lawrence Berkeley laboratory and the Lawrence Livermore national laboratory under the sponsorship of the US department of energy between 1976 and 1982 [3]. According to [4], a successful prototype of a public bus was manufactured and statically tested for quasi dynamic operation, with a road-way coil operating at a frequency of 400 Hz. The road-way inductor was about 37 inch wide and 4 inch deep and buried immediately below road surface coating. The inductor's cables were rated at 1000 amps and the system was capable of transferring 67kW across an air gap of 3 inch at an efficiency of 95%. The cross sections of the road-way and pick-up inductors are shown in figure 1.2 [3].

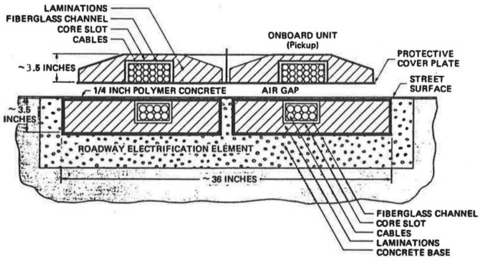


Figure 1.2: Schematic diagram of roadway and pick-up inductors developed for the Santa Barbra road powered bus project [3].

Later on, this work was extended by a group from UC Berkeley, California: partners for advanced transit (PATH) to further investigate issues of dynamic power coupling, range improvement, system efficiency, acoustic noise level, on-board weight and system costs[5]. The system was re-evaluated from a systems engineering perspective and optimal operating conditions were selected following involved computer simulations. A 700 foot test track, with two 200 foot electrified segments, was used for dynamic charging tests. An interesting contribution was that a steering assist system was incorporated into the test vehicle to assist the driver to minimize the center offset between the road-way and pick-up coils when passing over the electrified segments [5]. The reported overall transfer efficiency was 60% and range tests showed that the vehicle was capable of all-day operation as opposed to two hours of operation consuming a single discharge cycle of the battery pack, which is estimated as an increase in range from 15 to 200 miles [5].

The PATH program, however, had not been successfully commercialized due to high power rail construction cost, heavy on-board coils and high acoustic noise, in addition to relatively low transfer efficiency and large primary current which reached multiple thousands amps owing to the low operating frequency of 400 Hz. Furthermore, the maximum ground clearance for efficient operation was too small to meet road safety regulations and the maximum allowable lateral displacement was limited to 10cm which is very difficult for a human driver to maintain [2].

In 2009 a group of researchers at Korea Advanced Institute of Science and Technology (KAIST) undertook a project under the title “On-line electric vehicle” (OLEV), which aimed at reducing the costs of IPT systems for RPEVs. After several generations of their system, they managed to achieve promising results by operating at a much higher frequency compared to the aforementioned systems. The operating frequency for the OLEV systems reached 20 kHz and that allowed for a 20 cm air-gap and a fairly good lateral tolerance of 24 cm [2]. More details on the OLEV program and the evolution of its generations can be found in [2] and [6].

1.3 Challenges in power transfer control and regulation for IPT systems for RPEVs

IPT technology for RPEVs have dramatically improved over the past 20 years, however, it still faces some challenges that prevent it from becoming mainstream. Perhaps, the biggest of these challenges is the economic feasibility of large deployment of the roadway chargers in comparison to the more conventional transportation technologies. This problem in itself is beyond the scope of this study, but there are some other technical challenges which strongly impact the power transfer efficiency and the power ratings of the electrical components of dynamic IPT charging systems; such as the:

- relatively large ground clearance,
- tolerance to lateral offset,
- dynamic nature imposed by the relative motion between the two coils,
- conduction losses in the roadway coils when idling (no vehicles are passing over them),
- regulation of the pick-up power to match the varying load requirements (e.g: battery’s state of charge) while keeping the footprint of the on-board electronics cheap and reliable enough to match the feasibility requirements of automotive OEMs.

Implementing good engineering practices to face these challenges would lead to more efficient power transfer, hence less losses which translate into reduction in power ratings and ultimately cost savings. In the the next section,simplified modeling of the operational basics of resonating IPT equivalent circuits is introduced to further elaborate on the aforementioned challenges and point out to how autonomous driving technologies could aid in resolving some of them.

1.4 Characteristics of dynamic charging IPT circuits

As illustrated in figure 1.3 [8], in a typical IPT series-series compensated system; the two coils can be modeled as two coupled voltage sources where the voltage in each of the coils is proportional to the mutual inductance M and the rate of change of current in the other coil.

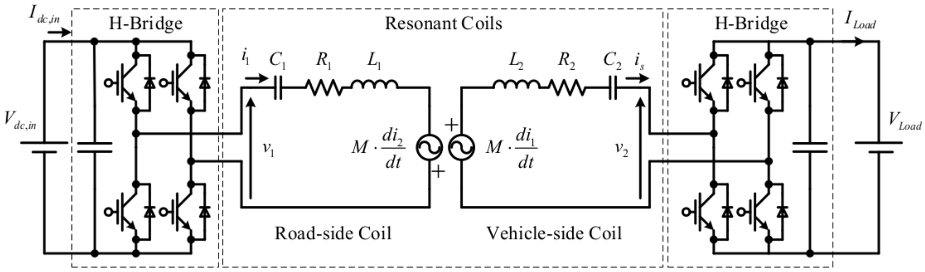


Figure 1.3: Equivalent circuit for a series-series compensated IPTS for charging vehicles [8].

In order to maximize the power transfer capabilities and minimize the VA requirements of the sending-end converter, the resonant circuits at either side of the link are usually tuned to the same resonant frequency [12]. By normalizing the system of equations representing the circuit in figure 1.3 and substituting M with k (non-dimensional magnetic coupling coefficient) the following is obtained [8]:

$$i_1 = \frac{\frac{v_1}{r_1} + \frac{v_2}{r_{12}}.kQ}{1 + (KQ)^2} \tag{1.1}$$

$$i_2 = \frac{\frac{v_1}{r_{12}}.kQ - \frac{v_2}{r_2}}{1 + (KQ)^2} \tag{1.2}$$

where:

$$r_{12} = \sqrt{\frac{R_1 R_2}{Z_{N,1} Z_{N,2}}}\ , Q = \sqrt{\frac{\omega L_1 \omega L_2}{R_1 R_2}} \text{ and } k = \frac{M}{\sqrt{L_1 L_2}}.$$

Equations 1.1 and 1.2 highlight the effect of varying the magnetic coupling coefficient k on the current levels i_1 and i_2 in both coils normalized by their rated values -at maximum overlap between the coils- in the primary and secondary coils respectively. Figure 1.4 [9] illustrates this sensitivity to the coupling coefficient (k) for both the ideally tuned case (blue and red curves) and also for the case of slight detuning of both coils (green and purple curves).

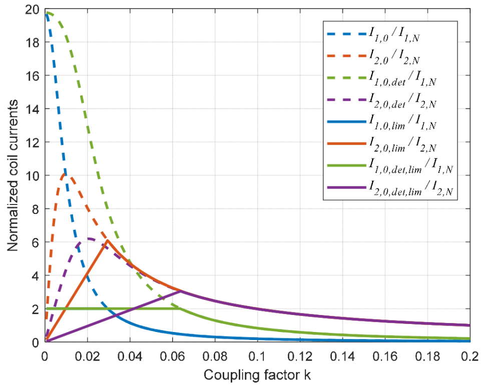


Figure 1.4: Sensitivity of the current in the primary coil to changes in the magnetic coupling coefficient k [9].

The trends shown in figure 1.4 are crucial when designing a control system for an RPEV dynamic charging system; as it is clear that for ideally coupled systems, the currents in both coils reach significantly higher values than their ratings when the coupling factor approaches relatively low values, and such situation is inevitable in the case of dynamic charging as the vehicle is continuously moving over the roadway coil and the coupling factor starts off virtually at zero and it increases until the two coils overlap, then this trend is reversed when the vehicle approaches the end of the roadway unit, until there is no overlap between the two coils and k drops back to zero. Such strong variation in the coupling coefficient means that the current ratings in the electric components on both sides should be much higher than their nominal operating currents unless a robust control system is introduced in order to limit the large increases in the currents. Successfully limiting the currents during the transient variation in k means that the ratings of the electrical components of the system could be dropped to values that make the system more feasible; without compromising the efficiency of power transfer. However, there would be some parasitic losses attributed to the idle operation of the roadway unit when no vehicles are passing over it. To alleviate such losses, a wireless connection between the vehicle and roadway unit needs to be established in order to have the vehicle trigger the charging unit only when the vehicle approaches it, as will be shown in this study.

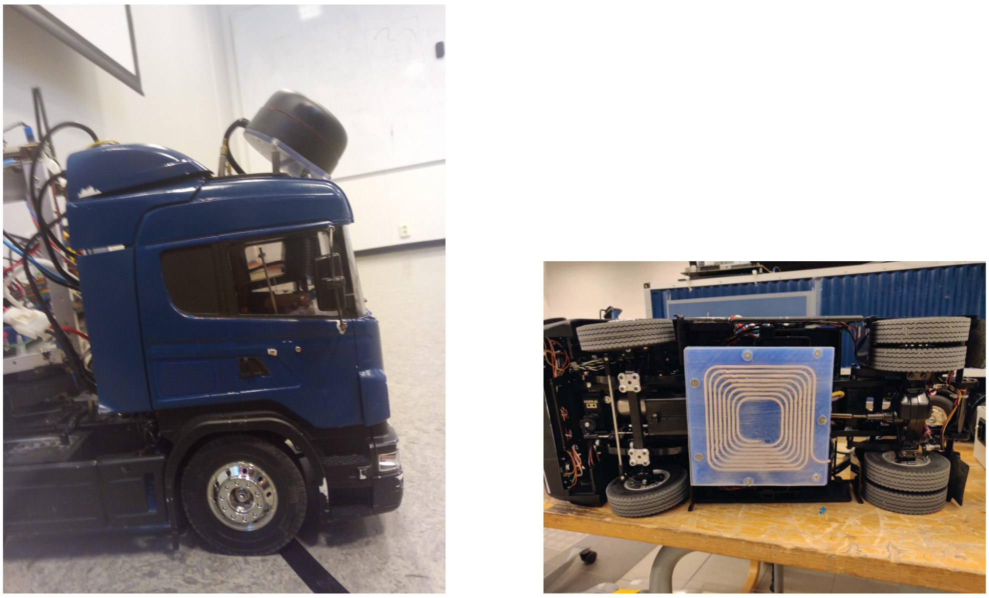
1.5 How could autonomous driving enhance IPT systems for RPEVs?

Considering the strong impact of k on the performance of dynamic IPT systems and taking into account that k is strongly dependent on the lateral offset between the primary and secondary coils; it becomes clear that accurate alignment between the pickup coil on board the vehicle and the roadway coil is crucial for such systems to operate efficiently and effectively. Such accurate alignment -within centimeters- could be difficult for a human driver to attain specially with large trucks that are difficult to maneuver. AI based autonomous driving could provide a good solution for this problem. In addition to that, making use of accurate mapping and localization techniques that are implemented in autonomous driving systems could allow for reductions in the parasitic conduction losses in the primary coil- mentioned earlier; by providing an accurate triggering signal to energize the roadway unit only when the vehicle is approaching instead of keeping the primary coil energized when idling. It is to be noted here, that in a real system the coil would probably only get energized when the truck is already moving over it; and as soon as the truck leaves the coil the energy to the coil would be cut-off. This is because the strong magnetic field generated by a full-scale idling energized coil would be accompanied with hazardous radiation.

1.6 Description of the experimental system investigated in this study

To evaluate the potential of utilization of autonomous driving in enhancing the performance of dynamic IPT chargers for RPEVs, a scaled down model of a freight truck equipped with wireless charging and autonomous driving capabilities is used. Figure 1.5 shows the truck with a lidar (light detection and ranging sensor) and a camera attached to the cabin and an on-board coil attached to it's bottom to receive the charging power.

Figure 1.6 shows the test track with two roadway charging coils that are connected to a rectifier/inverter unit which is responsible for energizing the coils. Technical descriptions of the truck and the charging system are given in [9] and [8] respectively, it is worth mentioning here that the computations for the autonomous driving part takes place on an on board Nvidia Jetson Tx2 development board, which will be referred to hereafter as the Jetson board. The operation of the charging system and the truck's autopilot are thoroughly discussed in the next chapters.



(a) Truck cabin with the Lidar mounted on top and the webcam mounted inside.

(b) The pick-up coil mounted underneath the truck [17].

Figure 1.5: Truck model used for testing.

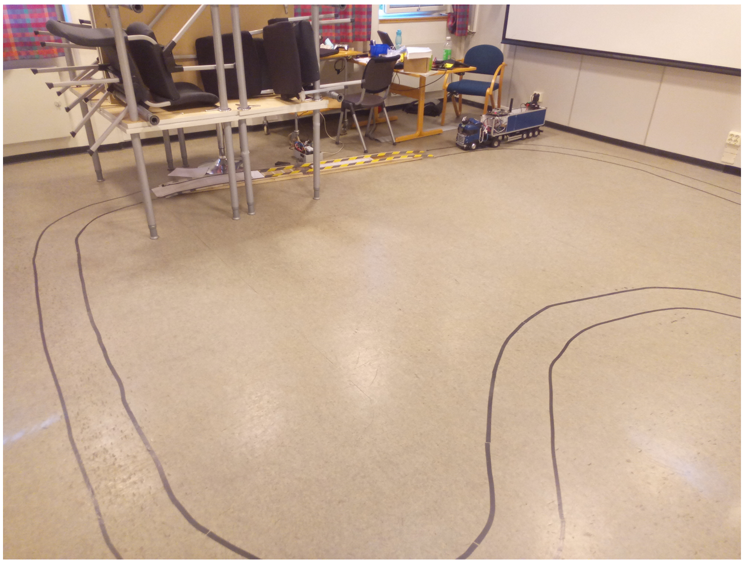


Figure 1.6: Test track with the two roadway coils placed within a marked charging ramp.

Chapter 2

Architecture of the experimental charging system

2.1 Introduction

In this chapter, the experimental charging system used in this study is explored. The system's operation and it's hardware/software architectures are briefly discussed. The need of establishing a communication between the wireless charger and the truck passing over it becomes clearer, as the overall energy transfer efficiency of the system can be significantly improved if the primary coil is to be energized only when a strong magnetic coupling between it and the secondary coil exists. An example to this enhancement in system efficiency is introduced at the end of this chapter as this communication link was tested under well controlled conditions in a laboratory as a prelude to establishing such a communication with the autonomous truck.

2.2 Hardware architecture

Figure 1.1 that shows the layout of a typical inductive wireless power transfer system that could be used for charging batteries, the topology is shown here again in figure 2.1 for convenience. The core controlling elements in such a system are the high frequency DC/AC inverter and the AC/DC rectifier on the primary coil -sending side- and the secondary coil -receiving side-, respectively.

For the system investigated in this study, both the inverter and the rectifier are equipped with H-bridges that allow for better controllability; by introducing a phase shift between the PWM (pulse width modulation) signals that control the operation of the gates of the leading leg and the signals that control the operation of the gates of the lagging leg, as illustrated in figure 2.2 [16]. Such a control technique allows for setting of the voltage across the primary coil and also that of the secondary coil independently, which is an effective method to limit the system currents when the magnetic coupling between the two coils changes or to

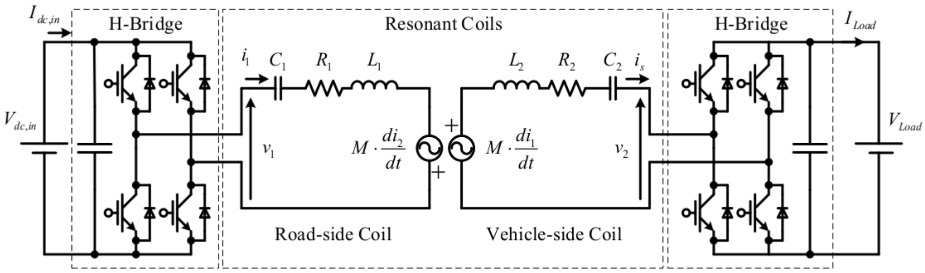


Figure 2.1: Equivalent circuit for a series-series compensated IPTS for charging vehicles [8].

control the amount of power transferred through the two coils – depending on the application. In [11] this additional degree of freedom introduced by the dual-side control was used to adjust the operating point by optimally distributing the losses between the primary and secondary coils to maximize the overall efficiency[11].

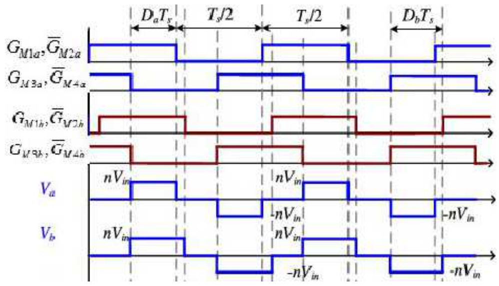


Figure 2.2: An example of the effect of changing the phase shift between the PWM signals controling an H-bridge in a typical inverter application, on the output voltage. The gate signals for two different cases are shown in the upper and middle pair of plots, respectively. The resulting output voltage for the two cases are shown in the last two plots to the buttom. [16].

The main hardware components that make up the road-way unit are:

- Coils and compensating capacitors -which are required to tune the circuit frequency to the operating frequency. An induction coil is the main component in the system. Driving the coil with an alternating voltage generates the magnetic field which when transferred to the pickup coil induces the driving voltage in the receiving circuit. The coils are made of Litz wire to minimize parasitic losses and sits on a ferrite back-plate to reduce the leakage flux and enhance the magnetic coupling.

- MOSFETs[1] (metal oxide silicon field effect transistors), which act as the electronic interface between the digital controller and the high voltage electric circuit. The fast switching nature and reliability of such devices make them

[1] in a full scale system the choice of devices might be a bit different (i.e. could be either IGBTs, or more likely SiC (silicon carbide) MOSFETs instead of Si MOSFETs as in the model).

a good option in high frequency switching applications, where their gates are controlled from a digital PWM controller signal generated by a digital computing device (e.g.: micro-controller or FPGA). Each full bridge converter/rectifier is made up of four MOSFETs and driven by PWM signals at the LC circuit's resonance frequency. The operating frequency for the system is around 75kHz, and was selected to be close to the standard frequency for wireless charging of vehicles as standardized by SAE (society of automotive engineering) at 85kHz by taking the dimensional scaling factors of the model into account [14].

- Computational devices: Two Xilinx zynq development boards are dedicated to control the charging system- one at each side. The board is equipped with an FPGA (field programmable gate array) chip together with two ARM processors. The FPGA is utilized to generate the HF PWM signals that control the switching of the MOSFETs and the phase shift between the two legs of the bridge is calculated in the microprocessors through software implemented PI controllers and the output of the controller is written to a register in the FPGA which in turn, generates the physical control signal. The board is also equipped with an ADC (analog to digital converter) and a bunch of IO peripherals.
  The FPGA is programmed to operate at variable frequencies, however, this is not the utilized in the control software for this study, as the frequency is fixed to replicate the state of the art standard for wireless charging systems for vehicles.

- Sensors: the primary coil circuit is equipped with a voltage sensor that measures the input DC voltage to the converter, together with two current sensors; the first measures the DC current through the converter which combined with the measured voltage comprise the input power. The second current sensor measures the maximum value for the current through the coil, which is needed when limiting the coil current so that it doesn't exceed the rated value of the design.

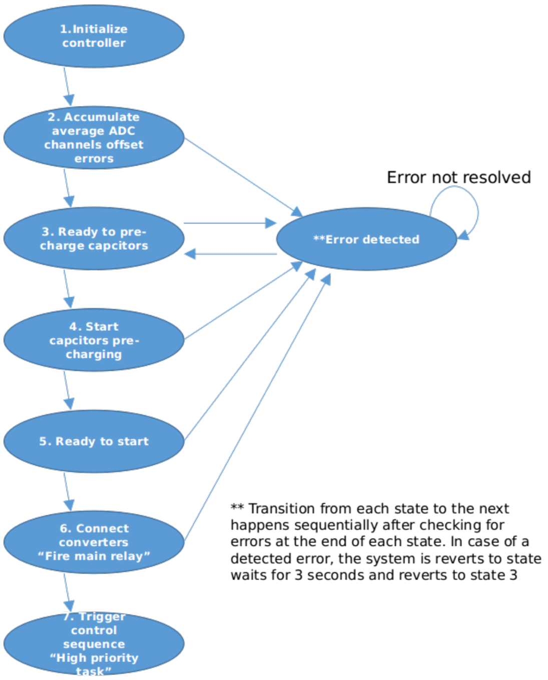
2.3 Software architecture

The software platform is based on freeRTOS [15], which is a lightweight operating systems that supports real-time performance by providing different levels of task priorities, together with other common OS (operating system) services. The SW operation of the road-way charging controller is divided into two main tasks; a high-priority task and a low-priority task. As the names imply, the high-priority task controls the switching of the converter bridge by setting a reference voltage based on the amount of power flowing through the coil. To further guarantee a better hard real-time performance of the controller, the high-priority task is triggered by a hardware timer interrupt that periodically preempts any running tasks at a fixed rate (50μ sec), and switches the context back to the high-priority task.

The low-priority task is mainly there to control any communication between the system and external devices, that could be used for debugging or logging of the system behavior. In this study, the low-priority task is utilized to listen to a start/stop signal that is received from the truck's autopilot.

The main software behavior of the controller is shown in figures 2.3 and 2.4 which represent two finite state machines representing the control flow of the software controller as follows:

1. High level system startup.


1.Initialize controller
2. Accumulate average ADC channels offset errors
3. Ready to pre-charge capcitors
4. Start capcitors pre-charging
5. Ready to start
6. Connect converters "Fire main relay"
7. Trigger control sequence "High priority task"
**Error detected
Error not resolved

** Transition from each state to the next happens sequentially after checking for errors at the end of each state. In case of a detected error, the system is reverts to state waits for 3 seconds and reverts to state 3

1)
- Both the precharge switch and the main switch feeding the coils are turned off
- ADC offset error reset to zero

2)
- Accumulate offset errors the ADC channels utilized
- Average of 1000 samples is calculated and stored for each channel

3)
- A soft reset state to revert to in case errors occur in the next steps
- Errors are abnormal values read from any of the voltage or current sensors

4)
- Turn the precharge switch on

5)
- Intermediate state

6)
- Turn the main switch on

7)
- Set an enable flag to trigger the feedback controller software

Figure 2.3: High level startup sequence for the controller of the roadway unit.

2. Voltage controller: PI controller for PWM phase calculation and setting.

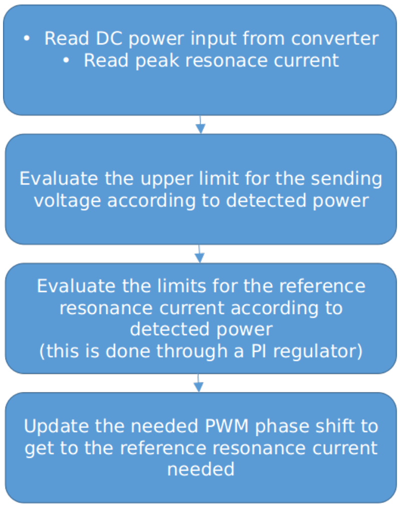

• Read DC power input from converter
• Read peak resonace current
Evaluate the upper limit for the sending voltage according to detected power
Evaluate the limits for the reference resonance current according to detected power
(this is done through a PI regulator)
Update the needed PWM phase shift to get to the reference resonance current needed

Figure 2.4: Abstract view of the PWM voltage controller for the roadway unit.

2.4 System behavior under well controlled operating conditions[14]

This section discusses the results of the experimental work done by the author of [14] to evaluate the performance of the scaled down model of the WPT system; that was designed to operate with the truck model.

The performance was experimentally tested under well controlled conditions by placing the pick-up coil on a servo controlled rail. The primary coil was placed underneath the rail and the two coils were perfectly aligned in the lateral direction, while the relative longitudinal positioning was precisely controlled by the servo rail as shown in figure 2.5. [14]

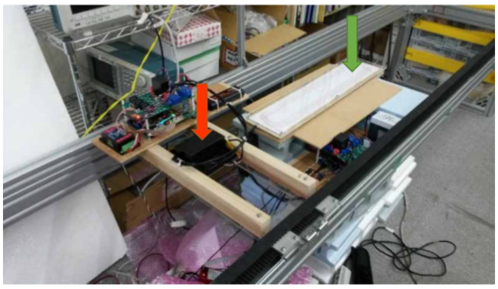


Figure 2.5: Experimental setup for the well controlled testing of the IPT system used in this study. Green and red arrows point to the primary and pick-up coils respectively[14].

Results from the testing are shown in table 2.1 [14] . The table summarizes the results from six different test cases with the parameters changing being the speed of the pick-up coil and the relative distance between the two coils at which the primary coil gets energized. The triggering was accurately controlled by estimating the distance from an encoder signal that is used to control the servo that moves the rail [14].

$0.845m/s$		
$E2_{-100} = 20.3J$	$\eta = 0.8$	$P_{av} = 25.5W$
$E2_{-100} = 20.4J$	$\eta = 0.82$	$P_{av} = 25.7$
$E2_{-100} = 17.3J$	$\eta = 0.83$	$P_{av} = 21.8$
$1.7m/s$		
$E2_{-100} = 9.9J$	$\eta = 0.8$	$P_{av} = 25.2$
$E2_{-100} = 9.7J$	$\eta = 0.82$	$P_{av} = 24.7$
$E2_{-100} = 7.7J$	$\eta = 0.82$	$P_{av} = 19.6$

Table 2.1: Pick-up power and efficiency at two operating speeds for three different triggering distances[14].

Figure 2.6 [14] shows the transient measurements of the load power and input power at the speed of 1.7 m/s and for the three different relative distance to trigger, namely, 100mm before the start of the overlap, right at the start of the overlap and 100mm after the start of the overlap between the two coils.

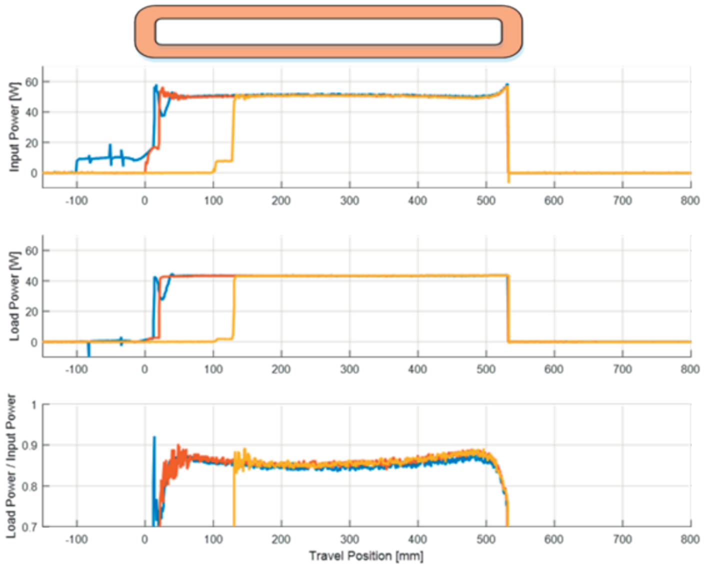


Figure 2.6: Transient input/pick-up power traces at 1.7m/s. A schematic roadway coil is shown at the top of the figure, as a reference for the relative postion of the pick-up coil. [14].

The results show that at the lower speed, triggering the primary coil right at the start of overlap gives the maximum transferred energy with an efficiency of 82% which is close to the highest efficiency case where the trigger takes place

100mm after the start of overlap at this speed. As, for the higher speed, things are a little different, since the time needed by the pick-up to travel the 100mm seems quite short relative to the response of the system. This leads to a decrease in the total transferred energy at the 0mm trigger case w.r.t. the case of -100mm trigger.

The transfer efficiency didn't change much with the speed and for both speeds the highest transfer efficiency was obtained with the 100mm delayed trigger, but this was on the expense of the total transferred energy. Which is an interesting trade-off and it could be decided by the truck's controller when to start energizing the road-way coil depending on the state of the truck's batteries and some estimate of the amount of energy required to get to the next charging unit- for example, provided a dynamic connection could be established between the truck and the road-way unit as suggested later in this study.

Chapter 3

Truck modeling and autonomous control

3.1 Introduction

In chapter 1, it was shown that the lateral offset between the primary (road-way) coil and the secondary (truck's pick-up) coil needs to be minimized, to guarantee an acceptable energy transfer efficiency. Since the power that could be picked up at the secondary coil decays rapidly with the increase in the lateral offset between the two coils; for a given input power at the sending side, as shown in figure 3.1[10]. This was the main motivation of combining autonomous driving with the scaled down WPT system introduced in chapter 2.

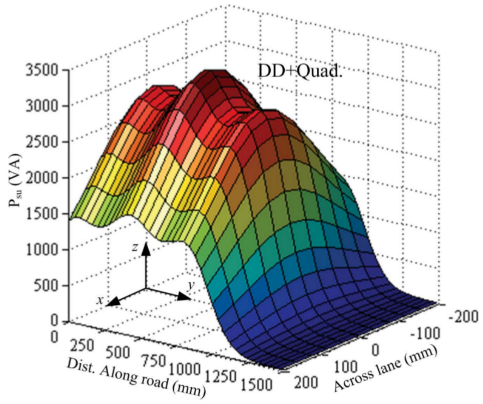


Figure 3.1: Example on the decay of available power to be picked up from an energized roadway coil setup consisting of 3 coils. DD+Quad refers to the magnetic/geometric design of the roadway coils. [10].

The work in this study builds on the work form [17], where three different autonomous path following techniques were compared; namely, SLAM-based path tracking (simultaneous localization and mapping), deep learning steering controller and computer vision steering controller. According to [17], the SLAM-based method worked better than the other two methods in a confined space with a fixed feature configuration over time -which in that case was the walls of a laboratory.

To follow up on the work in [17], one option was to test new techniques for autonomous lane keeping such as MPC (model predictive control) or deep reinforcement learning; which could be thought of as an extension to MPC with the model uncertainties implicitly taken care of during the training process. Another option was to revisit the computer vision model and enhance it's performance to make up for it's shortages as discussed in [17].

The second option was chosen; since MPC requires rigorous parameter estimation for the system, which is a tedious process for our case (with an off the shelf toy truck). Even in the case of using adaptive MPC, the adaptive uncertainty models are expected to increase the computational demand which is already relatively high for the basic MPC [18]. As for reinforcement learning, the biggest barrier was the long training time that would be needed to get to a satisfactory behaviour, therefore the option of working on an enhanced computer vision model was adopted since it would be less computationally expensive than MPC and the time needed to tweak the system parameters would be too short compared to the time required to train a deep AI model.

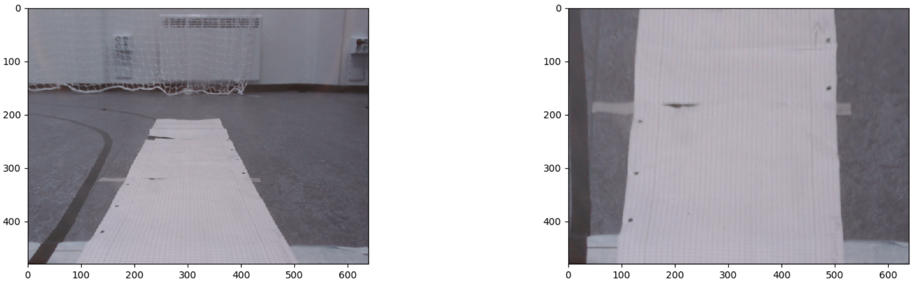
3.2 Simplified computer vision algorithms

3.2.1 Calibration

An on-board web-cam placed inside the cabin was used as the main source of perception for the truck. The raw image from the camera is initially transformed to a bird-eye view perspective, in order to allow for distance information estimation/extraction from the captured image. Such transformation requires a calibration; which was done using four 5mm grid sheets of paper of A4 size lined up in front of the truck, the two upper corners furthest away from the truck are then used to calculate the transformation matrix that would bring the two corners to the upper edge of the transformed image and align them with the two lower corners, as shown in figure 3.2. Knowing the size of the A4 sheets and the resolution of the image, a scaling factor is calculated in terms of pixel/meter which is then used to estimate distances between the truck and any point on the image given it's 'pixel' co-ordinates.

3.2.2 Image processing for lane detection

A mock-up track was built using black tape lines that represent boundaries of a driving lane as shown in figure 1.6. To have the truck determine the lane and extract the center line way-points, openCV (open source computer vision) [19]



(a) Image before transformation.

(b) Image transformed into the bird eye view perspective.

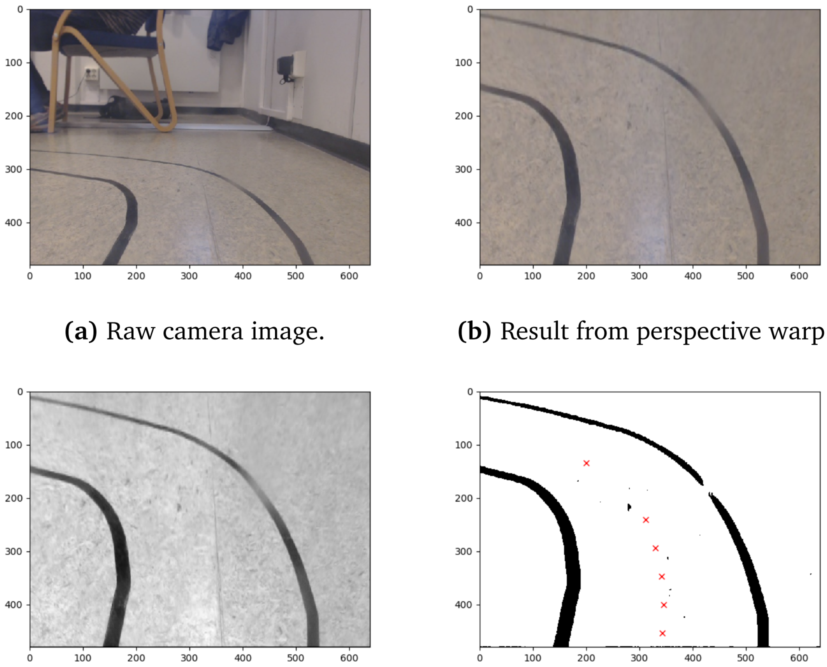
Figure 3.2: Illustration of the bird eye view transformation with the image used for calibration.

library was utilized; as openCV provides the most common image processing functions that were developed with real-time performance as the main focus. The algorithm for lane segmentation and point extraction implemented in this study, was based on the fact that the tape color was black on a relatively brighter background floor. This contrast was utilized by transforming the image from the default RGB (red-green-blue) color map to gray scale.

The gray scale image was then segmented by using an adaptive threshold mean function which applies a kernel of a given size that extracts the pixels with gray values higher than that of the mean value of the kernel size window and sets the intensity of the extracted pixels to a high value (255) while the intensity of the rest of the pixels within the window is set to a low value (0). The output intensity image is finally divided into ten windows along the y-axis direction; and for each window a histogram of the intensity is extracted and both the left-most and the right-most jumps (points with largest gradients in intensity) are extracted, as they would represent the left and right lane boundaries, respectively. The distance between the two points from each window is calculated to determine whether both sides of the lane were detected; if not, then the two points are merged into a single point indicating that only one side was detected- which is usually the case with sharp turns. Figure 3.3 shows an example of the pipeline that is used to extract the center-line of a well viewed lane. The algorithm implemented in this study is an enhanced version of that implemented in [20] as it depends on relative intensity differences in the image, which reduces the dependency on manual tweaking of the threshold parameter. This increases the robustness of the system to subtle changes in lighting conditions.

3.2.3 Image processing for charging pad detection

Figure 3.4 shows an example image of the special lane marking for the roadway charger coil and the steps undertaken to extract centerline points. Proper



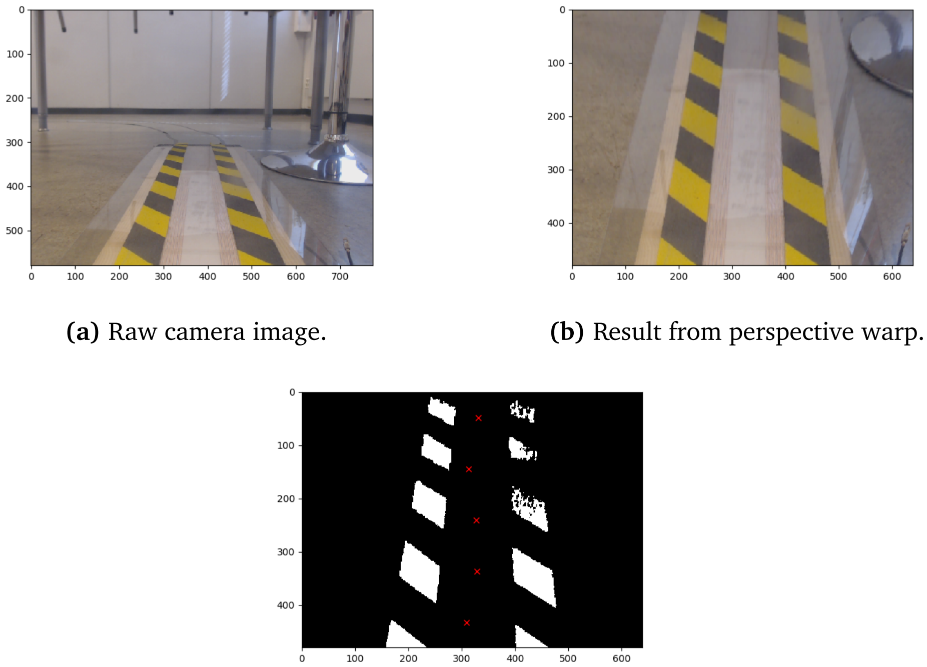
(a) Raw camera image.

(b) Result from perspective warp.

(c) Image transformed into grey-scale.

(d) Final intesnsity image & extracted points.

Figure 3.3: Illustration of the steps of finding the centerline points for a regular drive lane.



(a) Raw camera image.

(b) Result from perspective warp.

(c) Final intensity image & extracted points.

Figure 3.4: Illustration of the steps of finding the centerline points for the charging pad.

detection of this marking was crucial for determining where the charging begins, and also for extracting the center-line of the charging unit in order to align the pick-up coil on-board the truck with the road-way coil to maximize the efficiency of energy transfer and hence maximize the amount of energy transferred per pass over the charger. The yellow color in the markings was used for the detection of the charging lane, since it would stand out w.r.t. the other colors that the truck would encounter on its route. To do this, a masking filter is applied to every raw image captured by the camera, this filter transforms the image into an intensity image with the intensity of the pixels corresponding to yellow set to a high value (255) and the intensity for the other pixels is set to a low value (0). The total intensity of the image is then calculated and compared to a threshold value and the result of this comparison is used as a flag for charger detection. In case of detection of the charger, the same intensity image is used to detect the center-line way-points along the charging pad in a manner similar to that described in section 3.2.2 and is illustrated in figure 3.4.

Figure 3.5 shows the flow chart of the algorithm developed to extract the lane centerline from a processed intensity image. The code for this algorithm and that for the main autonomous driving ROS node that is described in section 3.3 are attached to the end of the thesis in appendix A.

3.3 Steering angle determination

3.3.1 Basic kinematic model

The truck could be modeled as a 4-wheel robot with Ackermann wheel configuration, where the trailer is not taken into consideration as the trailer dynamics is a large topic on its own and goes beyond this study.

The Akermann steering model is based on the kinematic constraints imposed by the configuration of the non-steerable (fixed) rear wheels and the steerable front wheels. This configuration as shown in figure 3.6[21] governs the motion of the truck in a circular arc with its center being located at the intersection point of the normal planes of all four wheels.

The kinematic equation of motion for such a configuration could simply be inferred from the simplified bicycle model, in which the two front wheels are merged to a single steerable wheel and the two rear wheels are merged into a single fixed wheel as shown in figure 3.7[22].

The relation between the steering angle δ and the turning radius R is deduced from the simplified geometry and could be approximated to the relation shown in figure 3.7[22]. This correlation is valid relatively slow speeds as the effects of the wheels' slippage and the system inertia could be neglected. Which would be applicable in our case since the truck's speed is limited to a low value due to the limitation in the available space and the low speed is also more suitable for the demonstration purposes of the study with the charging system included.

3.3.2 Steering angle calculation from detected lane points

To calculate the required steering angle for each point on the detected lane centerline, a radius of curvature is 'R' needed. This was calculated from a geometrical function that takes in as input two points and returns the radius and the center-point of an arc passing by the two input points. The center-point is needed to determine the direction of the turn and the radius is used to calculate the steering angle. The initial plan was to loop over all the detected points from each image and set the steering angle to the value corresponding to the respective point for a certain amount of time, that is calculated based on the truck's speed and the distance to the respective point. Then move to the next point until a new image frame is available. However, this methodology was not possible to implement as it was not possible to control in real-time when to update the steering angle as ROS (robot operating system) which is the framework used for development, does not support real-time computations. To overcome this shortage, a single point needed to be chosen from each image, and in order to have a good tracking behaviour the point closest to the truck was selected as a first iteration, since this was expected to guarantee that the truck doesn't cut corners- by overlooking the points closest

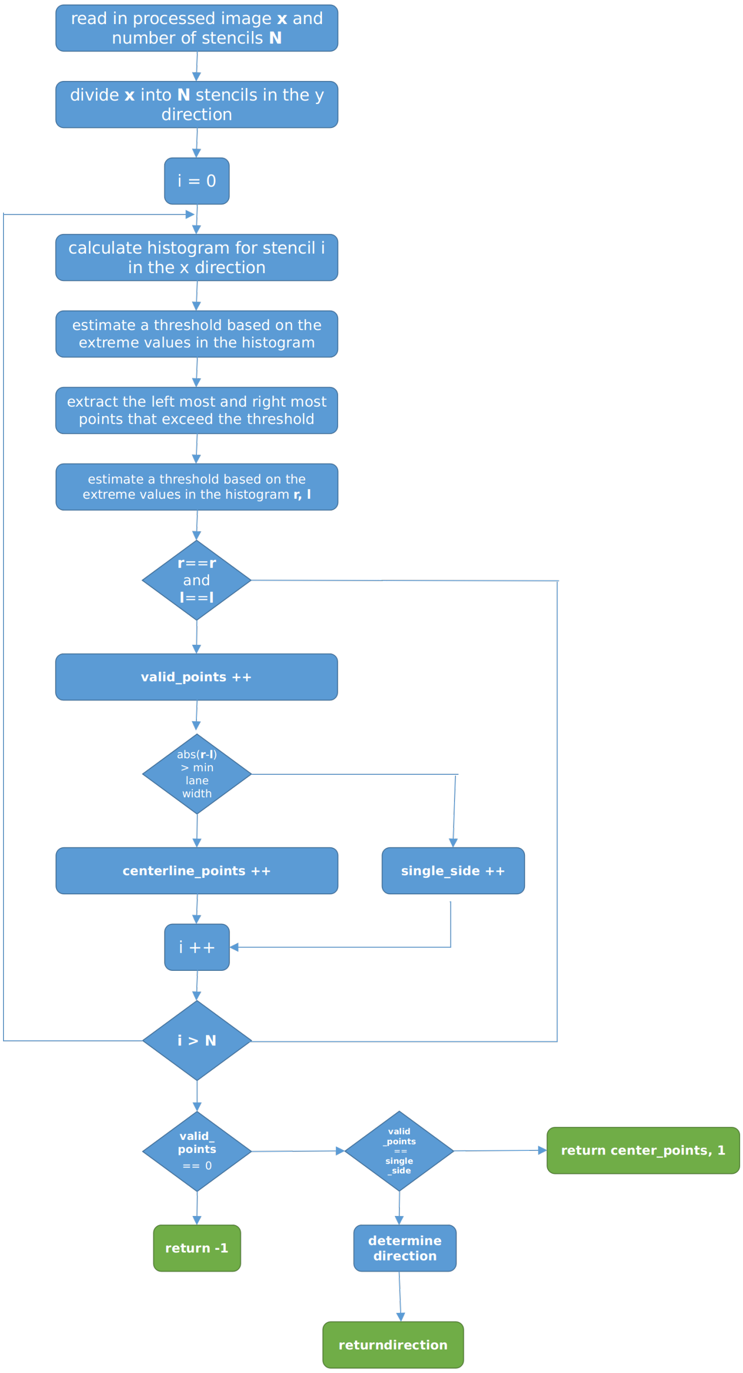

read in processed image **x** and number of stencils **N**
divide **x** into **N** stencils in the y direction
i = 0
calculate histogram for stencil i in the x direction
estimate a threshold based on the extreme values in the histogram
extract the left most and right most points that exceed the threshold
estimate a threshold based on the extreme values in the histogram **r, l**
r==r and **l==l**
valid_points ++
abs(**r-l**) > min lane width
centerline_points ++
single_side ++
i ++
i > N
valid_ points == 0
valid _points == single _side
return center_points, 1
return -1
determine direction
returndirection

Figure 3.5: Flow chart for extracting the centerline points from a processed image.

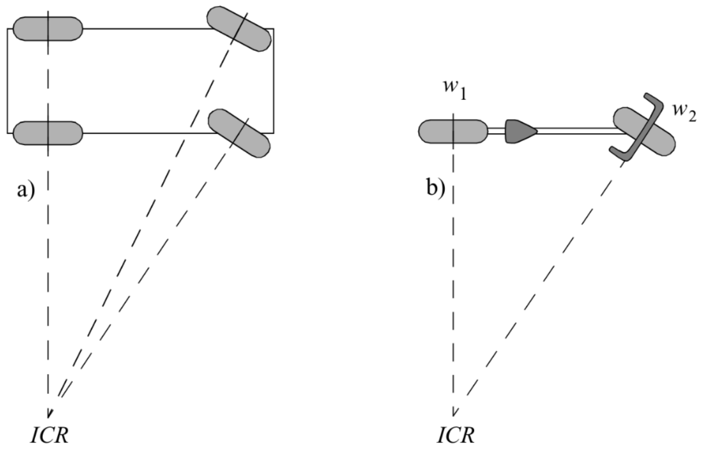


Figure 3.6: Ackermann wheel configuration and reduction to the bicycle model[21].

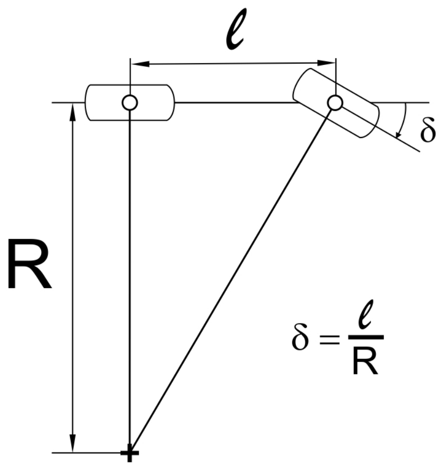


Figure 3.7: Geometric illustration of the calculation of the steering angle δ, where R is the turning radius and ℓ is the vehicle's wheel base[22].

to it. However, this selection led to a rough steering behaviour in some cases and a better procedure was developed and will be discussed in section 3.3.3.

It is also worth mentioning in this section that in the case of determining a single side of the lane boundaries- as in the case of sharp corners, the steering angle is set to it's maximum value in the direction of the detected side. The direction is obtained by comparing the x-coordinates of the the detected points to determine whether the value increases or decreases as we move forward from the first detected point.

3.3.3 Enhancement of the steering behaviour

To overcome the problem of rough steering behaviour mentioned in section 3.3.2, an averaging of the steering angle was introduced. For each image a single angle is calculated from a weighted average between the closest and the furthest points from the truck, with the furthest point getting a higher weight, which led to a much smoother steering behaviour without much corner cutting when moving along the track. The weighting factor needed to be reduced when passing over the charging pad in order to minimize the lateral offset between the two coils all the way along the charging pad, and the fact that the charging pad is straight led to the reduction of the roughness in the steering behaviour.

3.4 Experimental track setup and test results

To test the lane tracking and charger detection algorithms, a mock up track was setup using standard electrical black tape. The lane width was kept around 25cm corresponding to a standard Norwegian lane of width 3.5m scaled down to match the scale of the truck (1:14). The track contained some sharp turns in order to test the robustness of the system. The charger pads were placed on one side of the track as shown in figure 1.6.

Figure 3.8 shows the truck's trajectory for 7 consecutive rounds around the track. The overall performance of the tracking system seems consistent especially for the straight portion with the charging pads. However, the steep turns are sometimes missed, this issue is addressed in section 3.5. The odometry technique used here was not very accurate as it depended on the speed contorller that came with the truck, and this gives wrong readings on slopes and was the reason why the loops seem not to close; which is not the case in reality. The inaccurate odometry was also the reason for not placing all seven cycles on a single chart as they would look artificiallay offset from eachother defying the purpose of showing the cycle traces.

To improve the robustness of the system, a 2D Lidar sensor was used to detect obstacles and stop the truck in the case of an expected collision. The Lidar was mounted in a tilted position -towards the ground, on the top of the cabin, in order

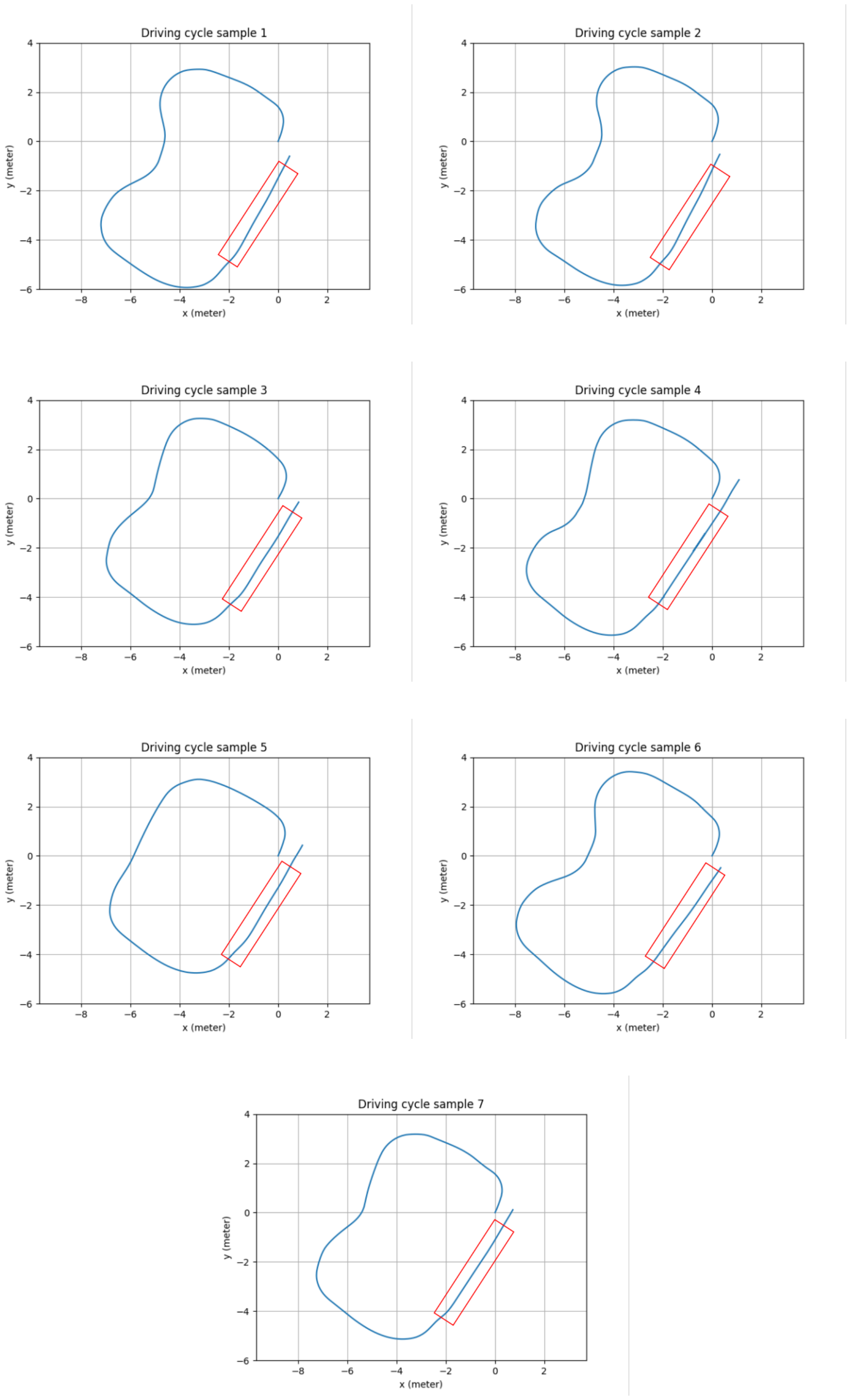
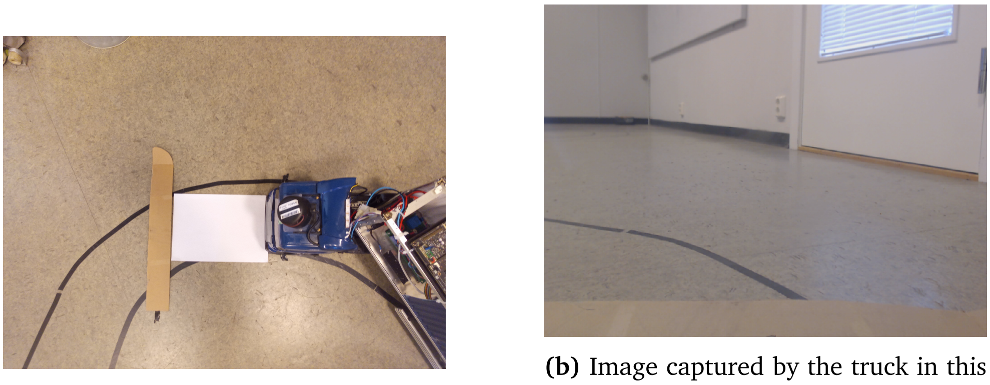


Figure 3.8: Truck's trajectory for seven consecutive driving cycles. A red regtangle in each figure is used approximately mark the charging pad.



(a) True bird eye view over the truck.

(b) Image captured by the truck in this position.

Figure 3.9: Illustration pf the blind spot in front of the cabin, that extends for 30cm ahead of the truck; as indicated by the 'unseen' A4 sheet

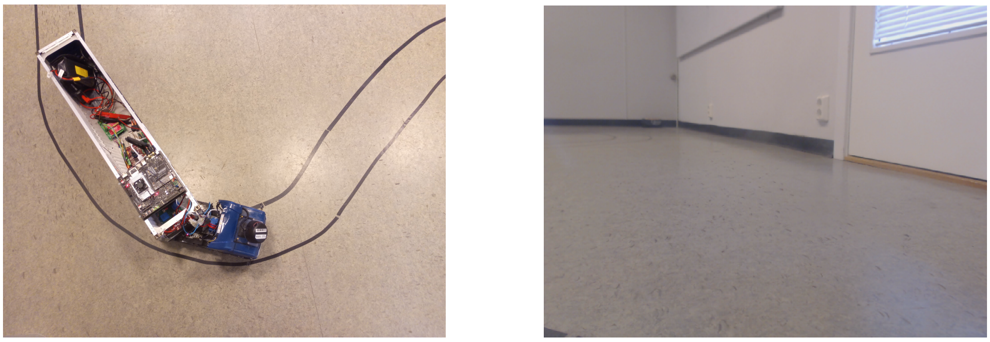
to be able to detect short objects passing in front of the truck. This setup required the Lidar point data analyzed to be limited within a range of -30:+30 degrees relative to the center-line of the truck. Figure 1.5a shows the truck's cabin with the Lidar mounted on top, together with the camera that was mounted inside the cabin.

3.5 Some drawbacks and possible solutions

As seen in figure 3.8, the truck sometimes "overlooks" the sharp turns within the track, this is attributed to the complete dependence on the camera image. The biggest problem with the camera image in this case is the presence of a relatively large blind spot that extends for 30cm right in front of the truck. This blind spot is illustrated in figure 3.10 which shows a top view of the truck with an A4 sheet and a brown paper strap placed in front of the truck at one of the sharp corners, together with the image captured by the truck from this position.

Figure 3.10 is similar to 3.9 with the truck at a different sharp corner, and the image captured by the truck's camera showing very little information about the lane ahead of the truck, from this position.

A possible solution to such situations would be to create a short term map from images with clear paths- with more than 2 way-points ahead, and to store that map even after the next image is captured. Then, in the case the newer image has less information about the lane ahead in comparison with the stored map, the truck reverts back to the stored map after applying some sort of dead reckoning to transform the stored set of points to their updated locations, taking into account the truck's displacement in the period between the two camera frames. Such a technique, however, would require sophisticated odometery in order for it to work properly, and it wasn't implemented in this study as the tracking performance was acceptable as the more important aspect of evaluation was the ability to



(a) Raw camera image.

(b) True bird eye view over the truck.

Figure 3.10: A case where the truck overlooks a sharp turn.

accurately track the center-line of the charging pad. Another possible solution is to use a smaller camera and mount it outside of the cabin, in a location close to the front bumper, in order to minimize the blind spot and apply some sensor fusion techniques to integrate the images from the two cameras into a single image with richer information.

Another drawback that was discovered in the testing phase, was that the camera wasn't perfectly mounted int the center of the cabin, which could be seen when the truck was initially tested on the charging pads. This was simply salvaged by manually adding an offset to the extracted center-line points, in order to compensate for the slight physical shift in the camera's position.

Chapter 4

Connection between the charging system and the autopilot

4.1 Introduction

As mentioned earlier, one main purpose of this study was to establish a reliable communication between the truck's autopilot and the road-way charger controller unit; as such a connection would open up the space for various applications that utilize the autonomy of the truck to enhance the performance of the energy transfer process. In addition to that, the flexibility of the charging system would be improved by allowing some of its parameters to be dynamically controlled, depending on the lively communicated requirements from the truck side. A good example to this would be the ability to vary the power supplied to the road-way coil depending on the state of the approaching truck's battery and the distance to the next charging point; which could be calculated on board and communicated over to the road-way controller on spot.

For the experimental system at hand, a 2-way communication setup was to be developed in order to have the autopilot trigger the charging unit on when the truck approaches the charger and off as soon as it starts moving away from it. BT (Bluetooth) communication was a good candidate for our case since the Jetson board controlling the truck is equipped with a BT controller and the Zynq board controlling the road-way unit is equipped with multiple UART (universal asynchronous transmitter receiver) controllers, one of which could be used to attach a standard BT to serial port module.

4.2 Bluetooth communication setup

The established BT communication link used RFCOMM (radio frequency communication) transport protocol, since it provides a reliability level similar to that of TCP; meaning that if a portion of the data sent from one of the peers cannot be delivered within a fixed time limit, the connection is terminated returning an error [23]. Which is important in our case so as to make sure that the state of the road-way unit is always predictable without the need of physical probing. In sections 4.2.1 and 4.2.2, the details of BT setup and the modifications that were introduced to both the road-way unit and the truck's autopilot are presented.

4.2.1 Modifications in the road-way unit

The software architecture for the road-way controller- as described in section 2.3 consists of two main tasks; namely the high priority and low priority tasks. The BT polling was introduced in the low priority task so as not to disrupt the control sequence of that takes place in the high priority task. The BT module is initialized and configured at system start-up by calling the respective functions from the board support package provided by Xilinx Zynq- the board supplier.

In the polling routine, the software waits for a specific sequence of characters to be sent over from the truck in order to assert a newly introduced flag "truck_approaching". This flag is shared between both the low and high priority tasks and it is used to inform the high priority task about the request received from the truck to switch the unit on or off. This was implemented as a finite state machine in the low priority task as shown in figure 4.1.

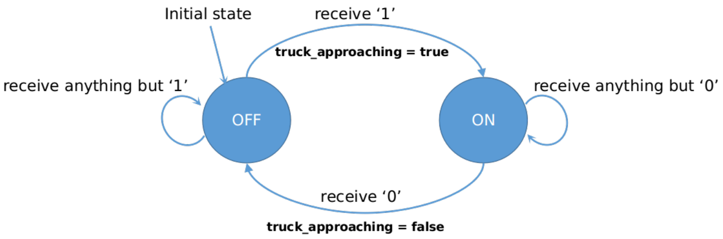


Figure 4.1: FSM descibing the control sequence for the low-priority task (responsible for the BT communication) in the road-way unit.

The flag "truck_approaching" was introduced into the road-way controller's state machine that was shown in figure 2.4; and the modified version is shown here in figure 4.2.

A simple debug mode was implemented in the BT polling routine which could be activated by sending the character sequence 'dn' (debug on) and deactivated by sending 'df' (debug off). In this mode the road-way controller immediately responds to any received message to inform the user of its state as shown in table 4.1.

To test the 2-way communication capability, the polling subroutine was further modified to send the integration of the total energy consumed by each of the coils during the period the system was in the 'on' state. The integration takes place in the high-priority task and the integrals are reset as soon as the operating mode is switched to 'off' soon, after the they are sent over to the truck.

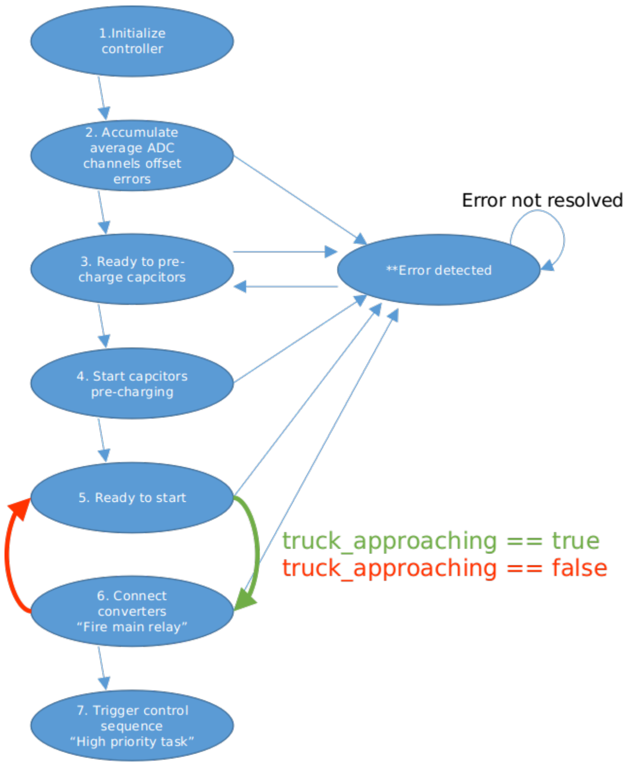

1.Initialize controller
2. Accumulate average ADC channels offset errors
3. Ready to pre-charge capcitors
4. Start capcitors pre-charging
5. Ready to start
6. Connect converters “Fire main relay”
7. Trigger control sequence “High priority task”
**Error detected
Error not resolved
truck_approaching == true
truck_approaching == false

Figure 4.2: FSM from figure 2.4 with the inclusion of the new flag that is controlled through BT communication.

	Reply message (depending on the current state)	
	ON	OFF
Received message: ‘1’	‘already on’	’on’
Received message: ‘0’	‘off’	‘already off’
Received message: ‘df’	-	-
Received message: otherwise	‘message ignored’	‘message ignored’

Table 4.1: Reply BT messages from the roadway unit when in debug mode.

It was noticed, however, that in some cases a delay needed to be introduced on the truck's side before receiving the integrals from the road-way controller. This could be attributed to the fact that the low-priority task could get preempted by the high-priority task during the sending instruction, which corrupts the sent message and raises an exception on the receiving side- the truck. Extending the receiving timeout on the truck side to 100 milliseconds resolved the issue. The 100 milliseconds were estimated after running a quick latency test that is reported later in section 4.3.

4.2.2 Modifications in the truck's autopilot

The BT programming on the Jetson board was developed in python following the guidelines in [23], that describes the use of a python library named pyBluez in order to establish an interface to handle the BT communication through serial message passing. A state machine similar to that on the road-way controller was introduced to the truck's autopilot, with the only difference being the trigger that controls the transition between the states, which in this case is the detection of the charging pads- in the camera's image. The state machine for the truck's autopilot is shown in figure 4.3. The BT connection is initialized by connecting to the BT module on the charger unit as soon as the autopilot mode is activated. In case connection link gets broken, an exception is raised and the truck can be driven in manual mode only in this case.

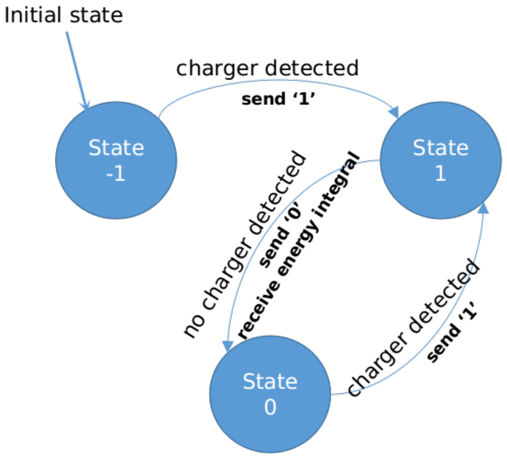
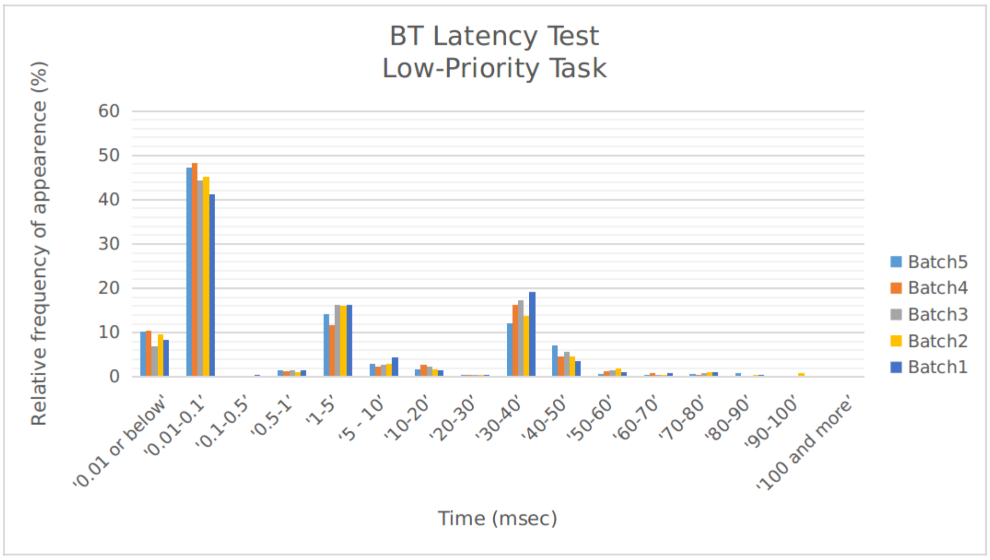


Figure 4.3: FSM for the BT communication control on the truck's side.

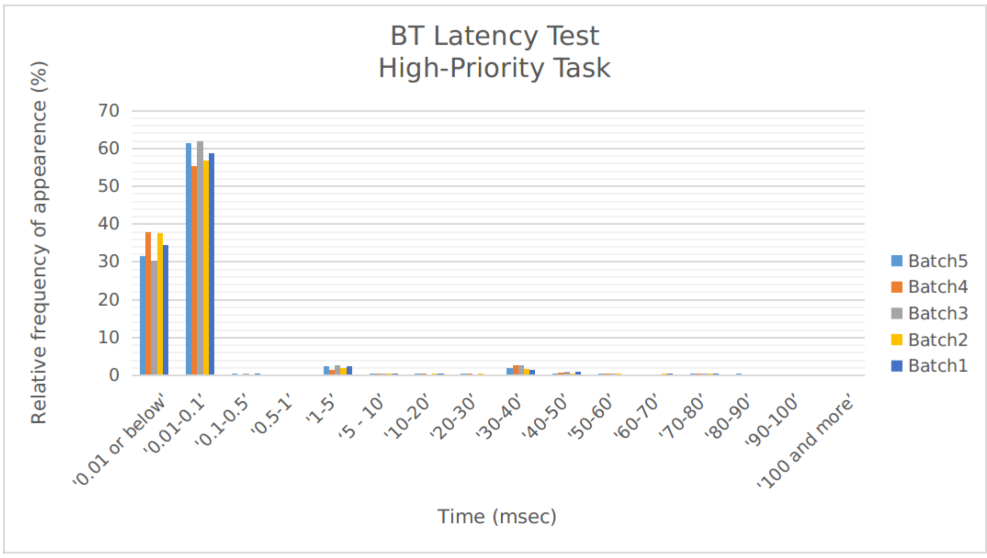
4.3 2-way communication latency study

A quick study was conducted to estimate the worst case latency in a 2-way communication between the truck's autopilot and the road-way charger unit with the setup described earlier. To measure the latency, a small python script was developed to initiate the BT link and send a message from the Jetson board and start a timer, then receive an expected message from the road-way unit -with a large timeout allowed- and stop the timer as soon as the message is received. The time recorded is assumed to be the overall latency in the of response from the road-way unit; as the receive instruction on the Jetson board is a blocking instruction that will keep the script in a busy state until the message arrives form the other side.The test was repeated 5 times with 500 consecutive send/receive cycles. A parameter added to the test was whether the BT polling/replying on the road-way unit took place in the low or the high priority task. The results for both cases are shown in figure 4.4.

It can be inferred from the figure 4.4 that placing the BT polling in the high priority task decreases the frequency of occurrence for latency values more than 10 milliseconds to much lower values compared to the case of the low priority task doing the polling; which is expected to be due to the less interruptions that delay message reception/sending when the high-priority task handles the polling/replying. The small number of cases with high latency -with the high-priority task doing the polling- is probably due to some operating system house keeping tasks that might be preempting the high priority task on few occasions.



(a) Turn around latency in BT communication with the polling on the roadway unit taking place in the low-priority task.



(b) Turn around latency in BT communication with the polling on the roadway unit taking place in the high-priority task.

Figure 4.4: Results from the BT latency analysis.

Chapter 5

Real-time testing

5.1 Introduction

This chapter summarizes the important experimental findings obtained with the truck autonomously driving the test track, and controlling the energizing/ de-energizing of the roadway unit. The ability of the autonomous driving system to maintain a relatively small center line offset between the roadway and the pick-up coils is evaluated by comparing the steady state energy transfer rate (power) picked up while driving, to that picked up when the pick-up coil is perfectly aligned with the roadway coil- in a static test.

Two scenarios for the triggering strategy were tested, the first being similar to what was described in section 4.2.1. This scenario turned out too inefficient when considering the overall energy transfer of the system. This was due to elevated parasitic losses, that were the result of relatively long idling periods in both coils.

In the second scenario a modification was introduced to the system to separate the triggering for each of the coils, which led to significant reduction in the idling losses that were seen in the case of the baseline scenario.

5.2 Effectiveness of autonomous driving

To evaluate the ability of the auto-pilot in minimizing the lateral offset between the roadway and pick-up coils, the steady state dynamic power picked up, is compared to the power picked up n the static case with the coils perfectly aligned. This was done for each of the two roadway coils separately, since the two coils have different parameters and it was important to quantify difference in power transfer efficiency between the coils. The 'ideal' static pick-up power was similar for the two coils- as regulated by the controller on the receiving end, as shown in figure 5.1. However, the input power for coil 2 was higher than that for coil 1 , indicating that the parasitic losses in coil 2 are higher, which led to lower power transfer efficiency compared to coil 1 as shown in table 5.1.

	Coil1	Coil2
Input DC power (VA)	49.5	51.5
Pick-up power (VA)	36.5	36.5
Static power transfer effeciency (%)	73.7%	70.8%

Table 5.1: The two triggering scenarios for the coils of the roadway unit.

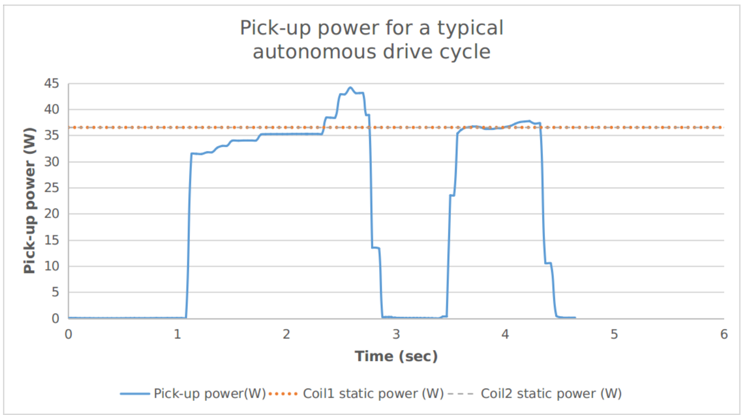


Figure 5.1: Pick-up power from a sample drive cycle compared to the 'ideal' static pick-up power.

Figure 5.1 shows that the steady state pick-up power from the transient maneuver is very close to the ideal situation, with the perfectly aligned coils -for both roadway coils; indicating that the system is quite robust against minor lateral offsets that could be introduced by any inaccuracies in the autonomous driving system. This also ensures that the autonomous driving system is operating as intended and is effective in minimizing the lateral offset between the pick-up coil and the centerline of the charging pad, which was qualitatively shown earlier in section 3.4.

5.3 Effectiveness of the real-time communication

The connection between the truck and the charger unit was utilized to switch the charger on when the truck is approaching, and switch it off when the truck starts moving away from it. This strategy was implemented in order to evaluate the potential gains in overall energy transfer efficiency, through utilizing the available perception information form the autopilot to minimize the idling losses in the charger coils.

It is worth noting here, that for a real system, the idling times are expected to be very limited, to prevent public exposure to strong magnetic fields. This is due to the hazardous radiation rates that would be generated in the case of a full scale coil that would extend for several tens of meters along the road. This means that not all the results shown in this section can be extrapolated to a full-scale system. Nonetheless, the results from the experimental system are still informative, and present an interesting showcase for the potential applications of having a fast and reliable communication link between the truck and the roadway charger. Also, it could be argued that the transient dynamics and the trends in the energy transfer efficiencies captured in this study, could well be carried on to full scale systems.

As mentioned earlier, two different triggering scenarios were tested, referred to as 'baseline' and 'modification', from this point on. The differences between the two scenarios are summarized in table 5.2.

	Baseline		Modification	
	Coil1	Coil2	Coil1	Coil2
Turn-on	As soon as the charger pad marking is detected	As soon as the charger pad marking is detected	Wait for 5 seconds after the marking is first detected	Wait until no more power is picked up from coil1
Turn-off	When the charger pad marking is no longer detected	When the charger pad marking is no longer detected	When the picked up power from coil1 drops below a certain threshold	When the picked up power from coil2 drops below a certain threshold

Table 5.2: The two triggering scenarios for the coils of the roadway unit.

The performance of the system is compared for the two scenarios by looking at the two transient power curves shown in figure 5.2, which clearly shows the differences in the idling periods of each of the coils through out the complete charging maneuver.

Table 5.3 summarizes the comparison between the two scenarios in terms of overall energy transfer efficiency; by integrating the losses from each of the coils, and comparing that to the amount of picked up energy from the respective coil.

	Baseline		Modification	
	Coil1	Coil2	Coil1	Coil2
Total converter input energy (J)	132.4	161.6	117.7	65.8
Picked up energy (J)	61.1	31.5	74.6	40.9
Idling losses (J)	41.2	109	15.4	8.3
Coil overall transfer efficiency (%)	46.1%	19.48%	63.4%	62.2%
Coil net transfer efficiency (%)	67%	59.7%	72.9%	71.1%
Overall system efficiency (%)	35%		63%	

Table 5.3: Energy transfer efficiency comparison between the 'baseline' and 'modification' trigerring scenarios.

The net transfer efficiency for a coil is calculated by dividing the pick-up energy by the net energy (integration of the input power only when the truck is passing over the coil). This was higher for coil 1 in both scenarios, which is attributed to the lower parasitic losses in coil 1 as shown in table 5.1, and also to the fact that coil 1 is longer wich means that the truck spends more time in the region with steady power transfer rate compared to the case of coil 2, increasing the overall transfer efficiency. This effect could also be seen when comparing the two scenarios, since the truck's speed seemed slower in the case of the modified scenario, probably due to lower battery voltage. The slower speed, allowed for more time to be spent in the steady power transfer rate region for both coils in this case. Which led to higher net efficieny for both coils (72.9%,71.1%) compared to the baseline scenario (67%,59.7%). It is important to note that the sampling rate on the truck side is much lower than that on the roadway side, this is due to to the large latencies in the CAN bus that is used to send the pick-up power measurements from the on-board Zynq board to the Jetson board. Such discrepency would lead to some inaccuracies in the integrated values, however, the overall trends in the transinet dynamics would be preserved.

5.4 Conclusion

The test cases of the complete system, showed the ability of the autopilot in keeping the lateral offset within a small margin, which boosted the dynamic energy transfer rates to a point similar to that in the case of ideal static performance; which is a significant result, that if implemented in a full scale system would lead to beneficial gains on both the technical and economical sides.
Adding the robust communication link and utilizing it to minimize the idling times, worked as expected and led to significant improvement in the overall energy transfer efficiency, for the two cases compared. This result does not map directly into a real system, because the idling operation of the roadway coils in a full scale system would not be allowed. However, the established 2-way communication link could have various applications in a full scale system, some of which will be discussed in chapter 6.

The study also gives some insight into some important trends in the energy transfer efficiency, that would probably be reproduced in a full scale system, such as the effect of parasitic losses that have a strong impact on the net transfer efficiencies (excluding the idling losses). Also the transient nature of the system, as imposed by the motion of the vehicle over the charger, was found to play an important role in the overall enegy efficiency transfer, and should be thoroughly considered when designing a full scale system.

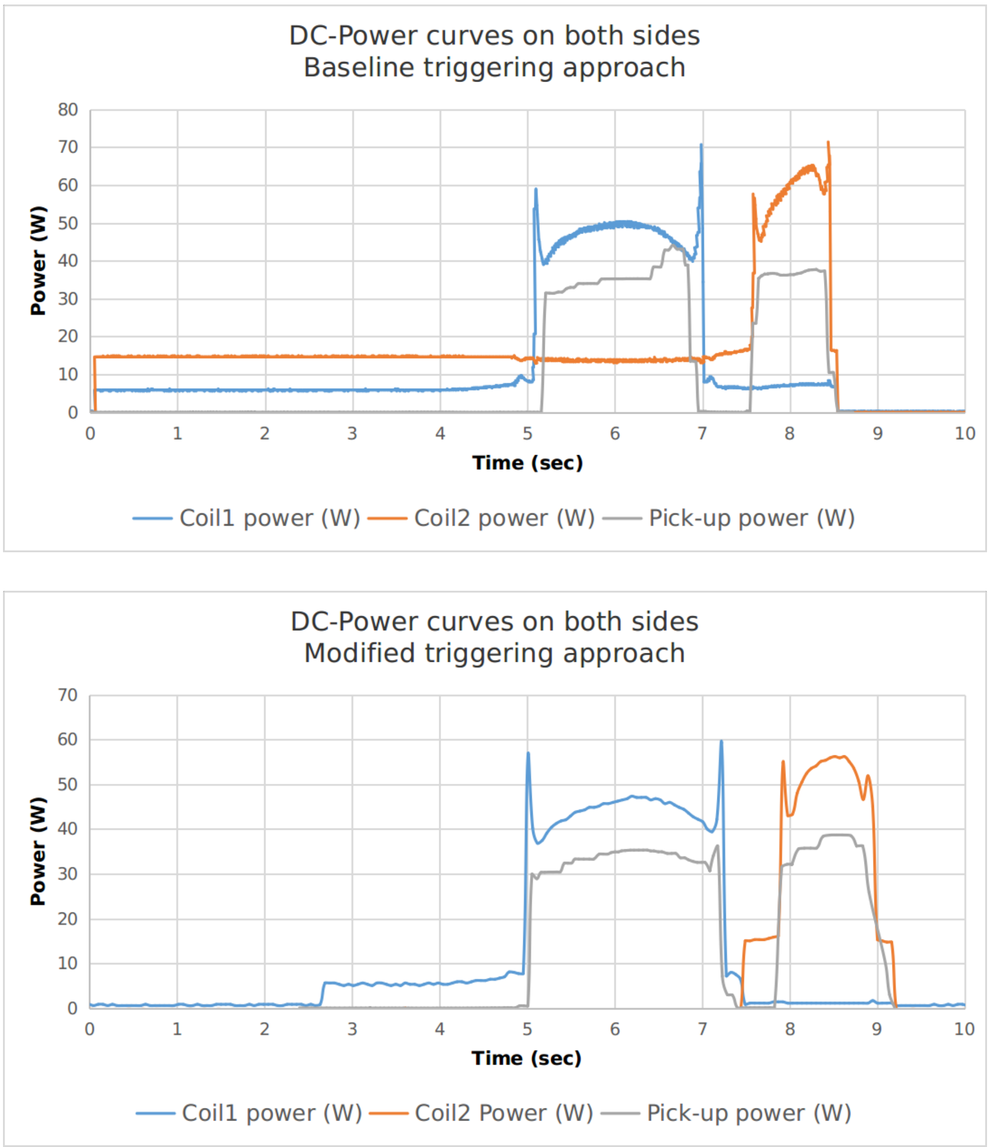


Figure 5.2: Input/pick-up power traces for the two tested triggering scenarios.

Chapter 6

Future work

The study shed some light onto the potential gains from integrating autonomous driving into vehicles with dynamic WPT capabilities. Further gains could still be achieved both for the experimental system and for full-scale implementations.

For the experimental system, a more sophisticated technique for autonomous driving could be applied to allow for autonomous operation at larger speeds, which is not possible with the current system which is based on a kinematic model. A good approach would be unsupervised machine learning, which takes away the hassle of manual identification of the system parameters and uncertainties. The problem with such an approach, however, would be the relatively long computation time needed for the training phase, which could be compressed to some extent by creating a simulation model to be used to initiate the training process, and transfer the learned behaviour to the actual truck model at some latter point. Another approach would be enhancing the odometry of the truck by applying more rigorous feature recognition and segmentation to the current computer vision system, and use the extracted information together with the IMU unit for more accurate pose estimation. The improved odometry would facilitate incorporation of a model the accounts for the dynamics of the system, which would increase the speed range. The steering behaviour could be enhanced by the incorporation of a closed loop steering controller; with the feedback signal read by an additional encoder or through the IMU by applying inverse dynamics, using the model mentioned earlier.

As for the possible full scale applications, the real-time communication between a truck and the roadway unit could be utilized in large scale optimization for route planning, by making the truck able to request a specified amount of energy to be transferred to it form each charger on its way separately, depending on real-time requirements which could vary due to external sources such as traffic or weather conditions.

Appendix A

Appendix.A: Code

Code listing A.1: Image processing and lane detection functions

```
import numpy as np
np.set_printoptions(threshold=np.inf)
import os, glob
import cv2
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from scipy.signal import argrelextrema

def undistort_image(img):
      mtx = np.array([[708.4340756775686, 0, 317.9663110540382], [0,
          ↪  724.3038241696117, 274.0865876256384], [0, 0, 1]])
      dist = np.array([0.09702126218642344, -0.1836878268546886,
          ↪ 0.01359685879119158, 0.007942342964235989, 0])
      undist = cv2.undistort(img, mtx, dist, None, mtx)
      return undist

def perspective_warp2(img,
                dst_size=(640,480),
                src=np.float32([(144/640,270/480), (465.0/640,
                    ↪ 270/480), (35/640,480/480), (580/640,480/480)
                    ↪ ]),
                dst=np.float32([(5/640,0/480),(645/640,0/480)
                    ↪ ,(35/640,480/480),(590/640,480/480)])):
   img_size = np.float32([(img.shape[1],img.shape[0])])
   src = src* img_size
   dst = dst * np.float32(dst_size)
   M = cv2.getPerspectiveTransform(src,dst)
   warped = cv2.warpPerspective(img, M, dst_size)
   return warped,M
```

```
def inv_perspective_warp2(img,
                dst_size=(640,480),
                src=np.float32([(241.8,208.25), (361.0, 208.25),
                    ↪ (90,480), (507.48,480)]),
                dst=np.float32([(40,0),(600,0),(40,480),(600,480)]))
                    ↪ :
   img_size = np.float32([(img.shape[1],img.shape[0])])
   src = src* img_size
   dst = dst * np.float32(dst_size)
   M = cv2.getPerspectiveTransform(src, dst)
   warped = cv2.warpPerspective(img, M, dst_size)
   return warped

def pipeline3(img): #look for the lanes
      x,y = perspective_warp2(img)
      plt.imshow(cv2.cvtColor(x, cv2.COLOR_BGR2RGB))
      plt.show()
      img_gray = cv2.cvtColor(x, cv2.COLOR_BGR2GRAY)
      th3 = cv2.adaptiveThreshold(img_gray,255,cv2.
          ↪ ADAPTIVE_THRESH_MEAN_C,\
            cv2.THRESH_BINARY,151,+20)
      return th3

def pipeline4(img): #look for the charging pod
      x,y = perspective_warp2(img)
      dark_yellow = (23,100,100)
      light_yellow = (30,255,255)
      mask1 = cv2.inRange( cv2.cvtColor(x, cv2.COLOR_BGR2HSV),
          ↪ dark_yellow, light_yellow)
      print(mask1.sum())
      if mask1.sum()>100000:
            return 1, mask1
      else:
            return 0, mask1

def get_hist(img):
   hist = np.sum(img[img.shape[0]//2:,:], axis=0)
   return hist

def draw_lane(pointsx,pointsy): #img,window_y,lane_center
      plt.plot(pointsx,pointsy,'rx')
      return
```

```
def find_lanes2(img,slices,lane):
            right_lane = np.zeros(slices)
            left_lane = np.zeros(slices)
            sx = int(img.shape[0]/slices)
            sy = img.shape[1]
            stencil = np.zeros([sx,sy])
            copy = np.zeros_like(img)
            index_l=0
            index_r=0
            window_y = np.linspace(0+sx/2,img.shape[0]-sx/2,slices)
                ↪ #window center (vertical)
            for i in range(slices):
                  stencil = img[i*sx:i*sx+stencil.shape[0],0:stencil
                      ↪ .shape[1]]
                  histogram = get_hist(stencil)
                  if lane == 2:
                        left_lane[index_l]=None
                        right_lane[index_r]=None
                        for x in range(len(histogram)):
                              if histogram[x]!=0:
                                    left_lane[index_l]=x
                                    break
                        index_l+=1

                        for x in range(len(histogram)-1,0,-1):
                              if histogram[x]!=0:
                                    right_lane[index_r]=x
                                    break
                        index_r+=1

                  elif lane==1:
                        ####
                        minima = argrelextrema(histogram, np.less)
                        if len(minima[0])>1 and max(histogram[minima
                            ↪ [0][0]],histogram[minima[0][1]])
                            ↪ <5000:
                              threshold = max(histogram[minima
                                  ↪ [0][0]],histogram[minima
                                  ↪ [0][1]])
                        else:
                              if min(histogram)!=max(histogram):
                                    threshold = min(histogram)
                              elif min(histogram) == max(histogram)
                                  ↪ or min(histogram)>5000:
                                    threshold = -1000 #handles one
                                        ↪ lane only

                        left_lane[index_l]=None
                        right_lane[index_r]=None
                        for x in range(len(histogram)):
                              if histogram[x]<=threshold:
                                    left_lane[index_l]=x
                                    break
                        index_l+=1

                        for x in range(len(histogram)-1,0,-1):
                              if histogram[x]<=threshold:
                                    right_lane[index_r]=x
                                    break
                        index_r+=1
                  min_lane_width = 150
                  valid_points = 0
                  points_found = 0
                  sl = 0 #single lane detector
                  lane_center = []
                  yout = []
                  single_lane_x = []
                  single_lane_y = []
                  direction = 1
                  l=0
                  r=0
            for i in range(slices):
                  if right_lane[i]==right_lane[i] and left_lane[i]==
                      ↪ left_lane[i]:
                        valid_points+=1
                        if (right_lane[i] - left_lane[i])>
                            ↪ min_lane_width:
                              points_found +=1
                              lane_center.append((right_lane[i] +
                                  ↪ left_lane[i])/2)
                              yout.append(window_y[i])
                        else:
                              single_lane_x.append((right_lane[i] +
                                  ↪ left_lane[i])/2)
                              single_lane_y.append(window_y[i])
                              sl+=1

                              #yout.append(window_y[i])
            if valid_points == 0:
                  direction = -1
            elif valid_points == sl:
                  lane_center = single_lane_x
                  yout = single_lane_y
                  print(’valid␣points␣=␣%d,␣single␣lane␣points␣=␣%d’
                      ↪  %(valid_points,sl))
                  print("only␣one␣lane␣is␣visible␣!")
                  if right_lane[-1] < right_lane[-2]:
                        r=1
                        direction = -10
                        print ("right")
                  else:
                        l=1
                        direction = 10
                        print ("left")
            if lane == 1:
                  return lane_center, yout, direction
            elif lane==2:
                  charger_x =[]
                  charger_y =[]
                  for i in range(len(left_lane)):
                        if(left_lane[i]==left_lane[i] and right_lane
                            ↪ [i]==right_lane[i] and (right_lane[i
                            ↪ ]-left_lane[i])>200):
                              charger_x.append((left_lane[i]+
                                  ↪ right_lane[i])/2)
                              charger_y.append(window_y[i])
                  return charger_x,charger_y
```

Code listing A.2: Autonomous driving ROS node

```
#!/usr/bin/env python
'''
ackermann_drive_test.py:
   A ros script for automatic lane tracking of ackermann steering
       ↪ based robots
'''
import time
import sys
from cv_bridge import CvBridge, CvBridgeError
import rospy
from ackermann_msgs.msg import AckermannDrive
from sensor_msgs.msg import Image
from std_msgs.msg import Int64MultiArray
from sensor_msgs.msg import Imu
from nav_msgs.msg import Odometry
import matplotlib.pyplot as plt
# OpenCV2 for saving an image
import cv2
import numpy as np
from simple_trajectory import center2p, arc1pR, arc3p, distance
from lane_det_truck_thesis import pipeline3, pipeline4, find_lanes2,
    ↪  draw_lane
import bluetooth
# Instantiate CvBridge
bridge = CvBridge()
class lane_tracker:
      def __init__(self, args):
            # establish bluetooth connection
            bd_addr = "00:06:66:F3:79:D7"
            port = 1
            sock=bluetooth.BluetoothSocket( bluetooth.RFCOMM )
            try:
                  sock.connect((bd_addr, port))
            except:
                  print("Connection␣error,␣bluetooth␣failed␣to␣
                      ↪ connect!")
                  pass
                  sys.exit(1)

            self.lock_image = False
            self.ready = False
            self.battery_voltage = 0.0
            self.charger_current = 0.0
self.active_coil = 0
self.X_now = 0.0
self.Y_now = 0.0
self.offset_total = 0.0
self.counter = 0
self.current_offset = 0.0
self.power_old = 0.0
self.time_old = None
self.power_integral = 0.0
self.old_speed = 0
self.old_steering_angle = 0
self.raw_img = np.zeros((640,480,3))
self.processed_img = np.zeros((640,480,1))
self.state = -1 #initial state

if len(args)==1 or len(args)==2:
      self.max_speed = float(args[0])
      self.max_steering_angle = float(args[len(args)-1])
      cmd_topic = 'ackermann_cmd'
elif len(args) == 3:
      self.max_speed = float(args[0])
      self.max_steering_angle = float(args[1])
      cmd_topic = '/' + args[2]
else:
      self.max_speed = 0.0
      self.max_steering_angle = 0.449260 *2
      cmd_topic = 'ackermann_cmd'

self.speed = 0.0
self.steering_angle = 0
image_topic = "image_raw"
battery_voltage_topic = "CAN_bus/battery_voltage"
charger_current_topic = "CAN_bus/charge_current"
charger_power_topic = "CAN_bus/charging_power"

self.camera_sub = rospy.Subscriber(image_topic, Image,
    ↪ self.image_callback, buff_size= 640*480*3+5,
    ↪ queue_size=1)
self.battery_voltage_sub = rospy.Subscriber(
    ↪ battery_voltage_topic, Int64MultiArray , self.
    ↪ can_bv_callback, queue_size=1)
self.charger_current_sub = rospy.Subscriber(
    ↪ charger_current_topic, Int64MultiArray , self.
    ↪ can_charger_current_callback, queue_size=1)
```

```
            self.drive_pub = rospy.Publisher(cmd_topic,
                ↪ AckermannDrive,queue_size=1)
            self.imu_sub = rospy.Subscriber("odom", Odometry, self.
                ↪ odom_callback)
            rospy.Timer(rospy.Duration(0.01), self.pub_callback,
                ↪ oneshot=False)

            rospy.loginfo('ackermann_self-drive_node␣initialized---
                ↪ version:THESIS')
            filenumber = 1
            self.time_zero = rospy.get_time()
            filename = "cycle"+ str(filenumber)+"power.txt"
            f1 = open(filename,"w+")
            filename2 = "accelration"+str(filenumber)+".txt"
            f2 = open(filename2,"w+")

            while not rospy.is_shutdown():
                  if (self.lock_image == False and self.ready ==
                      ↪ True):
                        self.lock_image = True
                        self.ready = False
########## add a state machine to determine whether the charger is
    ↪ found
            # 1. check for the yellow strap
            # 2. if found : charger found (send bluetooth start) and
                ↪  switch to charger found state
            # 3. if in charger found state: check for the yellow
                ↪ strap
            # 4. if found : find the lane (yellow pipeline), else
                ↪ jump to no charger state and send bluetooth stop
            # 5. in no charger: look for the lane (black pipeline)
                        charger_exists, self.processed_img =
                            ↪ pipeline4(self.raw_img)
#handle driving
                        if charger_exists==1 :#and self.
                            ↪ battery_voltage <=7.95:
                              if self.state!=1:
                                    self.time_stamp = rospy.get_time
                                        ↪ ()
                                    self.state = 1
                                    filenumber += 1
                                    filename = "cycle"+ str(
                                        ↪ filenumber)+"power.txt"
                                    f1 = open(filename,"w+")
                                    filename2 = "accelration"+str(
                                        ↪ filenumber)+".txt"
                                    f2 = open(filename2,"w+")
                              yellow_lane_not_found = self.
                                  ↪ send_controls(self.
                                  ↪ processed_img) #track the
                                  ↪ charging marker
                              if yellow_lane_not_found == -1:
                                    self.processed_img = pipeline3(
                                        ↪ self.raw_img)
                                    self.send_controls(self.
                                        ↪ processed_img)
                        elif charger_exists==0:
                              # lane tracking
                              self.state = 0 #normal lane tracking
                              self.processed_img = pipeline3(self.
                                  ↪ raw_img)
                              self.send_controls(self.processed_img)
#handle communication with charger
                        if self.state == 1:
                              if self.active_coil == 0 and (rospy.
                                  ↪ get_time()-self.time_stamp)
                                  ↪ >=5.3:
                                    self.active_coil = 1
                                    if self.battery_voltage <=7950:
                                        ↪ sock.send("11\r\n")

                              if self.active_coil == 1 and (self.
                                  ↪ power_old-self.power_now)>20:
                                    self.active_coil = 2
                                    if self.battery_voltage <=7950:
                                        ↪ sock.send("12\r\n")

                              if self.active_coil == 2 and (self.
                                  ↪ power_old-self.power_now)>20:
                                    self.active_coil = 0
                                    sock.send("0\r\n")
                                    rospy.loginfo('␣\033[31;1m␣\n\n\
                                        ↪ n␣BT␣stop␣sent!\n\n\n␣
                                        ↪ \033[0m␣')
                                    time.sleep(0.25)
                                    data = sock.recv(32) # receive
                                        ↪ energy integtral
                                    #print (data)
                                    split = data.split('__')
                                    x1 = split[0]
                                    x2 = split[1]
                                    integral1 = int(x1)
                                    integral2 = int(x2)
                                    rospy.loginfo('␣\033[31;1m␣
                                        ↪ charger␣energy␣=%0.2f␣-␣
                                        ↪ %0.2f␣\033[0m␣',
                                        ↪ integral1/1000.0,
                                        ↪ integral2/1000.0)
                                    rospy.loginfo('␣\033[31;1m␣
                                        ↪ Battery_energy␣=␣%0.2f␣
                                        ↪ \033[0m␣', self.
                                        ↪ power_integral)
                                    self.power_integral = 0.0
                                    f1.close()
                                    f2.close()
                                    filenumber += 1
                                    filename = "cycle"+ str(
                                        ↪ filenumber)+"power.txt"
                                    f1 = open(filename,"w+")
                                    filename2 = "accelration"+str(
                                        ↪ filenumber)+".txt"
                                    f2 = open(filename2,"w+")
                        else:
                              if self.active_coil != 0:
                                    self.active_coil
                                    sock.send("0\r\n")
                                    rospy.loginfo('␣\033[31;1m␣\n\n\
                                        ↪ n␣BT␣stop␣sent!\n\n\n␣
                                        ↪ \033[0m␣')
                        #integrate power
                        time_now = rospy.get_time()
                        self.power_now = self.battery_voltage * self.
                        ↪ charger_current
                        f2.write("%0.6f,␣%0.4f,␣%0.4f\n"%(time_now-self.
                        ↪ time_zero,self.X_now,self.Y_now))
                        if self.time_old is not None and self.state==1:
                        #write power to file
                        f1.write('%0.6f,␣%0.4f,%0.4f,␣%0.4f\n'%(
                              ↪ time_now-self.time_zero,power_now,
                              ↪ self.battery_voltage, self.
                              ↪ charger_current))
                        delta_t=time_now-self.time_old
```

```
                      self.power_integral += (self.power_old +
                          ↪ power_now)/2*delta_t
                self.time_old = time_now
                self.power_old = power_now

    def image_callback(self, msg):
                try:
                # Convert ROS Image message to OpenCV2
                      self.raw_img = bridge.imgmsg_to_cv2(msg, "
                          ↪ bgr8")
                      self.ready = True
                except CvBridgeError, e:
                      print(e)

    def can_bv_callback(self,msg):
          offset = 0.075
          self.battery_voltage = msg.data[1]/1000.0-offset
          if msg.data[1] <7000:
                rospy.loginfo(’␣\033[31;1mBattery␣voltage␣too␣low!
                    ↪ ␣\033[0m␣’)
                sys.exit(1)


    def can_charger_current_callback(self,msg):
          if self.state == 0:
                self.offset_total += (msg.data[1])/1000.0 #average
                    ↪  offset value = 12/1000.0amps
                self.counter += 1
                self.current_offset = self.offset_total/self.
                    ↪ counter
          else:
                self.charger_current = msg.data[1]/1000.0-self.
                    ↪ current_offset #subtract average offset
                    ↪ value
                if self.counter == 100:
                      self.counter = 1
                      self.offest_total = self.current_offset

    def odom_callback(self,data):
          self.X_now = data.pose.pose.position.x
          self.Y_now = data.pose.pose.position.y

    def send_controls(self,img):
          min_distance_allowed = 2e-2*640/0.21 #2cm
```

```
speed_factor = 1.0
if self.state == 0:
      lane_center, window_y, single_lane_direction =
          ↪ find_lanes2(img,10,1)
elif self.state == 1:
      lane_center, window_y, single_lane_direction =
          ↪ find_lanes2(img,6,2)
if single_lane_direction == -1:
      rospy.loginfo('␣\033[31;1mInvalid␣image:␣lane␣
          ↪ lines␣not␣found!␣\033[0m␣')
      self.lock_image = False
      self.speed = 0
      self.steering_angle = 0
      self.pub_callback
elif (abs(single_lane_direction)==10):
      self.steering_angle = -self.max_steering_angle *
          ↪ np.sign(single_lane_direction)
      self.speed = 0.5
elif single_lane_direction == 1:
      p0 = [320,480+0.58*480/0.44] ###0.365### #28cm
          ↪ wheel base,23.4cm blind area in front of
          ↪ the truck, 480/0.44 is the scale image
          ↪ scale in pixel/m
      if self.state == 1:
            rad, direc, dummy_delta1 = center2p(p0[0],p0
                ↪ [1],lane_center[-1]-20,window_y[-1])
                ↪ #offset calibrated manually
            rad, direc, dummy_delta2 = center2p(p0[0],p0
                ↪ [1],lane_center[0]-20,window_y[0])
            factor = 0.75
            dummy_delta = factor*dummy_delta1+(1-factor)
                ↪ *dummy_delta2


      else: #regular lane -two sides available
            rad, direc, dummy_delta1 = center2p(p0[0],p0
                ↪ [1],lane_center[-1]-20,window_y[-1])
                ↪ ##### baseline #####
            rad, direc, dummy_delta2 = center2p(p0[0],p0
                ↪ [1],lane_center[0],window_y[0])
            factor = 0.3
            dummy_delta = factor*dummy_delta1+(1-factor)
                ↪ *dummy_delta2
            commands = dummy_delta
            if self.state == 1:
                  speed = 0.55/speed_factor #increase the
                      ↪ speed to be able to climb the ramp
            else:
                  speed =0.52/speed_factor
            self.speed = speed
            self.steering_angle = -commands#[i]
            self.pub_callback

def pub_callback(self, event):
      ackermann_cmd_msg = AckermannDrive()
      ackermann_cmd_msg.speed = self.speed
      ackermann_cmd_msg.steering_angle = self.steering_angle
      self.drive_pub.publish(ackermann_cmd_msg)
      self.lock_image = False
      if ( self.old_speed != self.speed or self.
          ↪ old_steering_angle != self.steering_angle):
            self.print_state()
            self.old_speed = self.speed
            self.old_steering_angle = self.steering_angle

def print_state(self): #'\x1b[1M\r'
      #sys.stderr.write('\x1b[2J\x1b[H')
      rospy.loginfo(
                    '\033[34;1mState:␣\033[32;1m%d,␣'
                    '\033[34;1mBat_volt:␣\033[32;1m%0.3f,␣'
                    '\033[34;1mSpeed:␣\033[32;1m%0.2f␣m/s,␣'
                    '\033[34;1mSteering␣Angle:␣\033[32;1m%0.2f␣
                        ↪ rad\033[0m',
                    self.state,self.battery_voltage,self.speed,
                        ↪  self.steering_angle)

def finalize(self):
      rospy.loginfo('Halting␣motors,␣aligning␣wheels␣and␣
          ↪ exiting...')
      ackermann_cmd_msg = AckermannDrive()
      ackermann_cmd_msg.speed = 0
      ackermann_cmd_msg.steering_angle = 0
      self.drive_pub.publish(ackermann_cmd_msg)
      f1.close()
      f2.close()
      sys.exit()
```

```
if __name__ == '__main__':
   rospy.init_node('simple_lane_tracking_node')
   simple_lane_tracker = lane_tracker(sys.argv[1:len(sys.argv)])
```

Bibliography

[1] Chun T. Rim and Chris MI, *Wireless power transfer for electric vehicles and mobile devices*, 8th Commercial Vehicle Workshop/Graz , 2018.

[2] Su Y. Choi et. al, *Advances in Wireless Power Transfer Systems for Roadway-Powered Electric Vehicles*, IEEE JOURNAL OF EMERGING AND SELECTED TOPICS IN POWER ELECTRONICS, VOL. 3, NO. 1, MARCH 2015.

[3] Steven E. Shladover, *The road-way powered electric transit vehicle – progress and prospects*, Transportation research record 1155.

[4] K. Lashkari et. al, *Inductive power transfer to an electric vehicle*, VIII International electric vehicle symposium, Washington DC, 1986. , 2018.

[5] Siemens, *Roadway Powered Electric Vehicle Project Track Construction and Testing Program Phase 3D*, California PATH research paper.

[6] OLEV project KAIST, *https://olev.kaist.ac.kr/en/*.

[7] Grant Covic, *Inductive power transfer (IPT) powering our future*, Presentation from the University of Auckland, faculty of engineering.

[8] Giuseppe Guidi and Jon Are Suul, *Transient control of dynamic inductive EV charging and impact on energy efficiency when passing a roadside coil section*.

[9] Jon Eivind Stranden, *Autonomous driving of a small-scale electric truck model with dynamic wireless charging*, Master's thesis in Cybernetics and Robotics, NTNU Trondheim, June 2019.

[10] Grant Anthony Covic and John Talbot Boys, *Modern Trends in Inductive Power Transfer for Transportation Applications*, IEEE JOURNAL OF EMERGING AND SELECTED TOPICS IN POWER ELECTRONICS, VOL. 1, NO. 1, MARCH 2013.

[11] Tobias Diekhans and Rik W. De Doncker, *A Dual-Side Controlled Inductive Power Transfer System Optimized for Large Coupling Factor Variations and Partial Load*, IEEE TRANSACTIONS ON POWER ELECTRONICS, VOL. 30, NO. 11, NOVEMBER 2015.

[12] Giuseppe Guidi and Jon Are Suul, *Minimizing Converter Requirements of Inductive Power Transfer Systems with Constant Voltage Load and Variable Coupling Conditions*, IEEE Transactions on Industrial Electronics, November 2016.

[13] Giuseppe Guidi and Jon Are Suul, *Transient Control of Dynamic Inductive EV Charging and Impact on Energy Efficiency when Passing a Roadside Coil Section*, IEEE Transactions on Industrial Electronics, November 2016.

[14] Giuseppe Guidi, *Small-scale model of Inductive charging system for long-haul trucks*, SINTEF Energy Research, ElinGO project appendix II.

[15] https://www.freertos.org/

[16] Prasanna Udupi, Akshay Kumar Rathore, *Dual Three-Pulse Modulation-Based High-Frequency Pulsating DC Link Two-Stage Three-Phase Inverter for Electric/Hybrid/Fuel Cell Vehicles Applications*, Research-Gate,September 2014.

[17] Jon Eivind Stranden, *Autonomous driving of a small-scale electric truck model with dynamic wireless charging*, Master's thesis in Cybernetics and Robotics, NTNU Trondheim, June 2019.

[18] Monimoy Bujarbaruah et. al, *Adaptive MPC for Autonomous Lane Keeping*, International Symposium on Advanced Vehicle Control, June 2018.

[19] https://opencv.org/

[20] Alaa Khalaf, *Self driving truck model with wireless charging capability*, Specialization project thesis in Cybernetics and Robotics, NTNU Trondheim, December 2019.

[21] Roland Siegwart and Illah R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*, MIT press.

[22] https://racingcardynamics.com/understeer-and-oversteer/

[23] https://people.csail.mit.edu/albert/bluez-intro/

[24] https://www.hackster.io/kemfic/curved-lane-detection-34f771

NTNU

Norwegian University of
Science and Technology