KTH Electrical Engineering

On-board recursive state estimation for dead-reckoning in an autonomous truck

FRANCISCO MARTUCCI

Master's Thesis at Automatic Control Lab
Supervisor: Jonas Mårtensson
Examiner: Bo Wahlberg

TRITA-XR-EE-RT 2014:012

Acknowledgements

First of all, I want to to thank my parents for giving me all the support and love I needed to finish this important goal for my future.

During this unique experience at KTH, I want to express also my gratitude to all my teachers and specially to my supervisor Jonas Mårtensson for helping me during this period in order to complete this Master Thesis project. His constant feedback over my work was the main reason of the success of this project.

I am thankful to my friends in the lab, Matteo Vanin, Jõao Alvito and Diogo Almeida gave me an excellent insight of the features of the Smart Mobility Lab and helped me overcome some difficulties during the development of this project.

Thank you all.

Abstract

Most of fully-autonomous vehicles are equipped with GPS devices in order to keep track of their exact location while driving towards any target destination. However, it is widely known that GPS systems are likely to fail under certain conditions, e.g., in long tunnels or during very bad weather conditions. In this master thesis work we present an Extended Kalman filter (EKF) framework for dead-reckoning in autonomous trucks equipped with a CPU, a gyroscope and four simulated sensors: a GPS, a magnetometer and two rotary encoders for velocity. The EKF will fuse the sensors measurements with a prediction that uses the kinematic model of a non-holonomic vehicle. In order to improve the estimation of the yaw angular position when a GPS outage is reported a new calibration method based on the rotation matrix is applied. This method is proven to effectively decrease the error while driving in GPS denied environments.

The tests are performed in a real-time embedded system, NI myRIO, that runs on-board of a 1:14 scaled Scania truck. The performance results confirm the correctness of our framework under short-term GPS outages, during many driving loops. Additionally, during long-term outages the estimation works pretty well for one loop and it has a good performance for multiple loops due to the unavoidable sensor drifting.

Referat

De flesta av fullt autonoma fordon är utrustade med GPS-enheter för att hålla reda på sin exakta position när man kör mot ett mål destination. Det är dock allmänt känt att GPS-system kommer sannolikt att misslyckas på vissa villkor, till exempel, under långa tunnlar eller under mycket svåra väderförhållanden. I detta examensarbete presenteras ett Extended Kalman filter (EKF) ramverk för dead-reckoning i autonoma lastbilar utrustade med ett gyroskop och fyra simulerade sensorer: en GPS, en magnetometer och två roterande pulsgivare för hastighet. EKF:n kommer säkring sensorerna mätningar med en förutsägelse som använder den kinematiska modellen av ett icke-holonomiskt fordon. För att förbättra uppskattningen av gir vinkelposition när en GPS-avbrott rapporteras testade vi en ny kalibreringsmetod baserad på rotationsmatrisen. Denna metod har visat sig att effektivt minska fel vid körning i GPS förnekade miljöer.

Testerna utförs i en realtid inbyggda system, NI myRIO, som går ombord på en 1:14 skalade Scanialastbil. Prestanda resultat bekräftar riktigheten av våra ramverk under kortfristiga GPS avbrott, under många driv loopar. Dessutom, vid långtidsavbrottuppskattningen fungerar ganska bra för en slinga och har en bra prestanda för flera slingor på grund av den oundvikliga sensorn drifting.

Resumen

La mayoría de los vehículos completamente autónomos están equipados con GPS para poder tener información de la posición exacta del vehículo mientras se viaja a algún lugar de destino. Sin embargo, es ampliamente conocido que los GPS son propensos a fallar bajo ciertas condiciones, por ejemplo, dentro de túneles largos o durante malas condiciones meteorológicas. En este trabajo de tesis de maestría se presenta un Filtro de Kalman Extendido (EKF por sus siglas en inglés) como base para la estimación de posición de camiones autónomos equipados con un CPU, un giróscopo y cuatro sensores simulados: un GPS, un magnetómetro y dos codificadores rotativos de velocidad. El EKF fusionará las mediciones de los sensores con una predicción que utiliza el modelo cinemático de un vehículo no holonómico. Con el fin de mejorar la estimación de la posición angular de derrape, en carencia de la señal del GPS, se aplica un nuevo método de calibración basado en la matriz de rotación. Este método ha demostrado reducir efectivamente el error mientras se conduce entornos carentes de señal de GPS.

Las pruebas se realizan en un sistema embebido de tiempo real, NI MyRIO, que corre a bordo de un camión a escala 1:14 de Scania. Los resultados de desempeño confirman la exactitud de nuestro enfoque durante fallas de señal de GPS a corto plazo. Adicionalmente, durante interrupciones en el servicio de GPS a largo plazo, la estimación funciona muy bien para una vuelta y sólo bien para múltiples vueltas, debido a que las mediciones del giróscopo inevitable se desvían con el tiempo.

Contents

Contents

Chapter 1

Introduction

Several thousands years ago humans started using the horse to move to their destinations; this was probably one of the first means of transportation in human history. Afterwards, the invention of the horse-drawn carriages brought comfort and increased safety to the travellers; it also meant the creation of the first vehicles in human history. Although they were safe enough, humans could not reach effectively their destination in terms of time. At least several thousands of years passed since the first horses were ridden for first time until the first modern car was built in 1886. In contrast to horse-drawn carriages these vehicles could be very harmful to humans.

During the following centuries the cars evolved from slow, heavy and inefficient to lighter, faster and highly efficient machines. Although safety has been a priority in the design of vehicles, accidents keep occurring; most of these accidents are caused either by human error or by problems in the vehicle manufacture. A solution to decrease the millions of deaths per year is what our technology-based world is asking for and hence safety has become one of the biggest challenges in the automotive industry. The answer to this huge problem is in the development of more intelligent transport solutions.

1.1. Autonomous vehicles

The idea of having vehicles or including innovative features in these that can decrease the number of accidents has been in the mind of researchers during the past century. An autonomous vehicle (AV), or self-driving vehicle, can be defined as a vehicle that partially or fully drives itself and which may ultimately require no driver at all [3]. Taking the horse-drawn carriages as reference, we can say that they are autonomous vehicle depending on the amount of control that the driver has upon it. For instance, if we are commanding continuously the movement of the horse, e.g., either by squeezing the horse with your calves or pulling the reins to some direction, then we are driving a non-automated vehicle. On the other hand, if the user is just sitting and waiting for the horse to move freely during a certain

time period then we are talking about an autonomous vehicle.

Naturally, there are vast differences between a fully autonomous vehicle and a vehicle that only has few autonomous features or no autonomy at all. In the following section, we explain the five different levels of vehicle autonomy created by the National Highway Traffic Safety Administration (NHTSA) [3]; the purpose is to define the terminology to be followed during this project.

1.1.1. Classification

It is important to notice that cars can be classified in different levels of autonomy. Every level is defined depending on the amount of control that the human has over the primary vehicle functions, i.e., brake, steering and throttle. The classification starts by the vehicles fully controlled by humans; followed by autonomous vehicles that require from partial to almost no interaction while driving; and finally, those self-driving vehicles with no human control inputs, fully-autonomous vehicles. Every hierarchy is explained in detail below:

- Level 0 (No automation): The user is in total control of the car primary functions as well as ensuring safe driving by continuously monitoring the status of the vehicle and its environment.

- Level 1 (function-specific automation): Although the driver keeps control of the vehicle as in level 0, some functions can be selected as additional safety features. In this level we find features that can automatically assume limited authority over a primary control, such as in autonomous cruise control (ACC) or under emergency situations such as in dynamic brake control (DBC). One or more functions can work at the same time, but they work independently from each other.

- Level 2 (combined-function automation): The driver is still responsible to monitor the safe operation in the driving environment. In this level at least two functions of level 1 work simultaneously. The driver has to be available for interaction at every time, because the system might relinquish the control with no warning.

- Level 3 (limited self-driving automation): Safety-critical functions are enabled at this level. These functions work for specific driving situations, depending on the traffic or environmental conditions. The driver has to be available for sporadic control of the vehicle but the safety relies mostly on the vehicle functions.

- Level 4 (full self-driving automation): The user has to provide a target destination and the vehicle will perform the whole trajectory with no need of user interaction. Therefore, the vehicle can be either occupied or not. At this level, safe operation rests solely on the automated vehicle system.

Historically all modern vehicles were located at level 0 at the beginning. At the time technology did not permit the manufacturers to design any sort of autonomous features for the vehicles. However, the need for better safety and some additional comfort led the investors to believe in the autonomous vehicles market. The history of how the vehicles passed from the level 0 to the level 4 is briefly presented in the following section.

1.1.2. History

During the middle of the 20th century engineers started to create ideas in order to add some autonomy to the cars. It was not until 1977 that the first real attempt to create a self-driving vehicle. It was a project research carried out by Tsukuba Mechanical Engineering Laboratory in Japan. With the help of some hardware the car followed white street markers while reaching speeds up to 30 km/h [2]. After this first trial different techniques have been used in the development of autonomous cars. These techniques have varied widely depending on the sensors that the developers have been able to use in the car. As technologies improve, the quality and precision of the sensors used in cars have increased to a large extent, while the size and weigh decreases. This can be translated to lower prices for massive production and a better acceptance in the market. However, the prices were still too high to fulfil the market's needs and autonomous driving was seen as something not feasible. Initially, the processing power of the vehicles' computers became a big challenge, but after the introduction of microprocessors this problem, as well as the problem of size and weight, were almost solved.

A huge turn was made when vision approaches were introduced in the research of autonomous systems. In 1980 a Mercedes-Benz robot van was designed by Ernst Dickmanns and his team at the University of Munich. This car achieved 100 km/h guided by vision in traffic-less roads. In 1995 Dickmanns drove a S-class Mercedes-Benz from Munich to Copenhagen and back. This used saccadic computer vision and transputers to react in real-time during the 1600 kilometres; with speeds up to 175 km/h and up to 178 km with no human intervention. This resulted in 95% autonomous driving. In the meanwhile, the DARPA-funded Autonomous Land Vehicle (ALV) in the United States designed successfully the first-road follower that used laser radars, reaching up to 30 km/h [2].

As a matter of fact, vision-based techniques are currently the most interesting and reliable techniques to be used in order to drive a car autonomously. During the last decade they have become very popular in the largest and most famous research groups in the autonomous vehicles industry. One of the uses of vision is to detect features as feedback data that can provide useful information to intelligent system in order to make online decisions regarding the trajectory to follow during driving. After analysing the current status of the system and the possible trajectory to follow, the control commands would be ready to be sent; these commands are commonly the throttle and the steering wheel angular position.

In 1996, Alberto Broggi of the University of Parma launched the ARGO Project,

where a modified vehicle was designed to follow the painted lane marks in an unmodified highway. The result was 1900 km in over six days on the motorways of northern Italy, average speed of 90 km/h. The car operated in fully automatic mode during 94% of the whole journey; the longest automatic stretch was 55 km. The vehicle was composed of only two black-and-white video cameras of low cost on board. The data of the camera was used in stereoscopic vision algorithms to analyse the driving environment [2].

Currently, we know that the driver is better than any autonomous system. AV development is not just the trajectories that the cars have to take, it also covers the interaction with the environment such as streets, weather, city laws and other drivers. All these factors affect the application of these projects but researchers and engineers are working very hard to overcome these barriers. The common goal of AV research besides coping with these factors is to reach high performance in each feature of the autonomous vehicle in order to provide safe, low-cost and intelligent solutions. The goal is to develop vehicles that eliminate those risk factors that are currently affecting the drivers. Some sectors have been promoting competitions in order to evaluate the idea of AV and they have resulted in excellent feedback for researchers.

Competitions

The funded projects in the United States of America, Demo I, II and III, accumulated information of many kilometres of self-driving in off-road environments that included all sort of obstacles. One of the best ways to promote the development of AV technologies has been the creation of competitions. During the following years between 2004 and 2007, one of the biggest competitions of self-driving the DARPA Grand Challenge took place. The competitors were teams from major universities and major manufacturers. This naturally resulted in a big step in the research of autonomous systems.

Nowadays, fully-autonomous vehicles cannot be found in the market yet. However, some vehicles have been able to drive hundreds of thousands of kilometres. In the following section a brief description is done about the latest technologies available for autonomous systems as well as the future implementations that are being studied internationally and locally.

1.2. Current state of technology

Internationally

There are fully autonomous cars on the streets of some countries; although still in experimental and development stage, they will come hopefully to the automotive market in the following decades. These cars have driven autonomously and logged hundreds of thousands of kilometres, which has demonstrated that technologies in this area have advanced substantially. The best example is Google's vehicles,

which operating fully autonomously, have been able to complete more than 800000 kilometres without a crash that can be attributed to the features that make the vehicle autonomous [3]. Moreover, there are many features that provide some level of autonomy to vehicles that are already being used on the automotive industry, as described in section 1.1.1.

Safety is not the only question mark in fully AV development; the user needs to reach his/her target destination correctly under all possible scenarios and in order to accomplish this goal the cars need to make decisions knowing the current state of the path to follow; decisions regarding for instance what route to take or what speed is more convenient depending on several external factors that the vehicle has to be able to consider. What happens if there is a car accident blocking the street and the only way to skip the traffic is giving a sudden U-turn or crossing the side walk, or some decision that goes over the typical. That is why another challenge in the development of autonomous vehicles is modelling human decisions to make these systems as intelligent as possible by covering a huge spectra of scenarios of driving situations.

Naturally, any kind of decision needs to be made based on an estimation of the current state of the vehicle and its environment. In order to meet this challenge sensors are the best tool to provide such information. The data fetched by sensors needs to be highly reliable because the decisions to be made are based on them. The reliability can vary due to several factors; disturbances might affect those readings, e.g. very sunny days or snow covering the streets makes very difficult to detect lane markers, even for humans. The manufacturer has to predefine also the minimum probability threshold to ensure safety and this value might change during the day or depending on the place. That threshold can be seen as the probability where the disturbance would not affect the capability of the vehicle (or one of its features) to complete its task or their tasks correctly.

Sensors: vision and lidar

Let us assume we have a complete map of an urban area with streets and buildings, as well as a positioning system with no error. Now if we try to drive from point A to point B we will need to detect all possible objects at the front of the vehicle, otherwise an accident might occur. Some kind of radar can provide information about the surrounding objects of the vehicle and some decision might be made, such as braking if an object is too close or avoiding it if possible. However, radar range is too narrow and some objects might not be detected. If we add the functionality of a lidar (light and radar) sensor then we will have a larger range to detect possible objects. We would know that there is some object at a certain distance with reference to the lidar unit, but we would not know what the object is. That is the idea behind why it is necessary to use vision in autonomous driving systems. Vision is composed of a camera and a computer with enough processing power that we could use to detect and, if required, even recognize objects in the surroundings of the vehicle.

It is known that vision is widely affected by weather conditions, the amount of traffic and many independent factors that can be found on the roads like pedestrians, animals or even tree shades. However, several algorithms have been used lately in order to improve vision-guided driving. A video-based lane detection is introduced in [5] using a fast vanishing point estimation method. The success of this approach is shown by the author of the paper in a video available at [1]. I encourage the reader to see this video to have a better understanding of the functionality of lane estimation for vision-guided vehicles.

Mark Campbell, the S.C. Thomas Sze Director of the Sibley School of Mechanical and Aerospace Engineering at Cornell University, gave a speech about Intelligent Autonomous Systems in October 2012. He mentioned that most of the finalists in one of the biggest self-driving competitions used a fusion of radars, spinning Lidar unit and vision in order to make decisions about the state of the environment. Using these different sensors provides better information about the environment because it fuses the advantages of these sensors. ''We are able to know not just what objects the vision detects but also how far they are'' said Campbell during his speech. I recommend the reader to watch the video of this speech for a better understanding of the status of current technologies; the video is available at [2].

Sensors: RTK-GPS

RTK (Real-time kinematic) technology was born in the early 1990's. The advantage compared to most other GPS technologies is that RTK allows us to reach centimetre-level positioning in real-time. The basic concept behind RTK is that we have a base station receiver set on a fixed point on an area and a surveyor who is operating the survey receiver, known as rover. There are communication links between a rover and the base station receiver. It takes measurements from satellites in view and then broadcasts them along with its known position to the rover receivers. The rover receiver also collects measurements from the satellites in view and processes them with the base station data. The rover then computes its location relative to the base. The corrections from the base station receiver can be sent to an unlimited number of rovers. In Real-Time Kinematic surveys, data processing occurs in the field as data is logged, providing immediate centimetre-level results in the form of coordinates. Real-time positions on the rover receiver are calculated as fast as 20 times per second or as little as once per second [26].

Locally

KTH has deepen its work on the development of intelligent transportation solutions since the creation of the Smart Mobility Lab. In this laboratory, students have had the opportunity to take part in different projects. One of these is a course that lasts a whole semester: the automatic control project course. Several transport

[1] `https://www.youtube.com/watch?v=6XaUYmHYWbsb`

[2] `https://www.youtube.com/watch?v=Jimh-YDZNF4&list=PL33ADE787A1607934&index=11`

features have been tested in these courses and every year the students have to work on a totally new project. An example is the deep research in vehicle platooning and semi-autonomous vehicles [1], where the researcher have been able to test different kinds of high-traffic situations as a way to compare results in a more realistic environment. In the following sections a brief description of the projects developed during the last years is given.

Automatic control project course 2012

A loading dock was built. This dock was composed of controlled trucks, a quadrocopter and a stationary tower crane. The purpose of this system was to simulate a loading dock. Initially, the trucks received a call from the freight hub, to move to it. While running the trucks are surveilled by the quadrocopter until the platoon leader reaches the final destination at the loading dock. The quadrocopter returns to its landing site and informs the central that the trucks are at their destination and they can be loaded by the freight hub.

In this project, all the devices are controlled by a central computer, which communicates through wireless nodes. This means that control signals are sent directly to each device in order to complete their tasks.

More information about this project can be found in [22].

Automatic control project course 2013

A hardware-in-the-loop (HIL) platform for developing and testing traffic control algorithms was established in this project. The purpose was to create traffic scenarios with model trucks which were as realistic as possible by using xPC. There is interaction between the trucks in real-time simulation, by using vehicle-to-vehicle (V2V) communication. The complexity of this platform can be increased to study traffic optimization algorithms and cooperation.

This project was focused in scaled trucks working with model trucks (provided by xPC) in different environments. Once again, all the processing was done directly in a central computer for all the trucks.

In order to compute the current position of the vehicles the lab was equipped with a Motion Capture (MoCaP) system. This system tracks the position of markers in order to provide excellent estimations of position.

For both projects it was impossible to drive whenever the data of the MoCap was lost. This meant that the scaled trucks had to stop when theirs markers could not be detected by the MoCap system. Moreover, using a central computer and sending information through wireless nodes added a delay factor that was important in the performance of the system under high traffic or under high velocity tests, where the vehicle may not able to react in time.

More information about this project can be found in [23].

Grand Cooperative Driving Challenge 2011

The Scoop project was a co-operational project between Scania CV AB and KTH. The project aimed to participate in the Grand Cooperative Driving Challenge (GCDC) in May 2011. The latter is an international challenge of cooperative driving, organized by TNO from the Netherlands. Every team designs a vehicle that takes part in platooning of vehicles, through urban city and highway driving scenarios. Platooning has to be autonomous.

In this competition three Swedish competitors ended up in the four first places, which speaks by itself the high level of development locally in autonomous systems.

More information about this project can be found in [18].

1.3. Motivation and objectives

The end goal of this project is to develop a system that can used in scaled autonomous trucks. This system will be adapted to real-life situations. Currently, we have at the Smart Mobility Lab at KTH an excellent position estimation, provided by the Motion Capture system with a undermilimetric variance. The main idea is to use this sensor as the position ground truth but it is also possible to simulate different sensors by using these data.

Initially, all the decisions were made in a central computer running a LabVIEW application, which communicated with the truck by using motes. These motes are wireless sensor network nodes. Each truck had an on-board mote which was communicating to its pair at the central computer. The central computer, while running the LabVIEW application, sent continuously the control signals to each truck mote. This meant that there was no autonomy in the system. In order to make this feasible we have to move all these functionalities to an embedded hardware device. By using such device we could make the driving decisions on-board of the truck instead of in the central computer. By adding an embedded device in each truck we could develop distributed autonomous systems and make even more interesting projects in the SML.

Some real-life problems could be added to the simulation environment, such as bridges, communications losses, obstacles, which could be then solved by the decision on the autonomous vehicle. With the help of the MoCap data we could, e.g., recreate the reception of an RTK-GPS by adding some simulated noise to the current MoCap position. This noise could be extracted from real data from an RTK-GPS. Another idea is to create more advanced scenarios for distributed vehicles in the SML and test different theoretical approaches. That is the reason why throughout the report we might call the data coming from the MoCap system the ‘‘GPS data’’. We are recreating the data of this GPS device by using the MoCap system of the lab.

Having a vehicle that is independent of a central computer means that the car has to be able to keep control of its functions and perform safety procedures in case of emergencies or in case of failure. It is also very important to have a good

knowledge of the current position at all time. That is the main motivation of this master thesis. We want to have a good estimation at all time. This means that we have to be able to estimate our current position even when there is no available GPS data, also called as ‘‘outage’’. For this reason, we need to study state estimation techniques suitable for our needs and analyse what sensors might be feasible to use in order to obtain a good performance.

After building this interesting platform of distributed autonomous vehicles, we open the doors for future projects in the Smart Mobility Lab that could be adapted to even more realistic situations. Currently, Linköping University, Scania, SAAB, KTH and Autoliv are working along in IQMatic project; the final goal is autonomous driving of tractor trailer trucks. Every organization has their own tasks and KTH might contribute largely by testing state-of-the-art technologies in the SML by using our platform. By having distributed autonomous vehicles driving in the lab we might simulate high density traffic and perform vision-guided driving, platooning, vehicle detection, pedestrian detection, and countless innovative ideas.

The expected outcomes of this project are the following:

- To build a test framework on the Smart Mobility Lab at KTH, which can be extended to more advanced research on autonomous vehicles.
- Configure a scaled truck to be able to drive autonomously, which is run by an on-board CPU connected to different sensors.
- This truck will be tested for different driving situations on the simulated tracks of the SML, initially with no autonomy and then gradually start adding up autonomous features
- In this extendible test framework the truck will have long-term navigation, in hard situations, when some sensors are not available. For example lack of current position data.
- Finally, the end goal of developing this system is to function as research tool for students and researchers at KTH. They will be able to test different technologies for their research in platooning and autonomous vehicles in these scaled trucks with on-board computers.

Creating such framework would provide a more reliable system for the future tests to come in the SML.

1.4. Project structure

The project will be divided in chapters. The first chapter provides important information of what composes the test framework, this will be very important because it will be the basis of our practical implementation. In the next chapter, we introduce the problem of truck motion by defining a kinematic model to be used.

Afterwards, we will discuss more about the theory behind recursive state estimation as well as its importance for dead-reckoning situations in autonomous driving. Additionally, we will explain which estimator suits better for our goals in order to keep track of the vehicle pose at every time.

After the theoretical section is given we continue to provide a detailed explanation of the practical implementation, where we give a detailed explanation of necessary assumptions to take into account before performing the tests. Afterwards, we test the integrated system and discuss the results of such tests.

We conclude with a general discussion of the obtained results of the whole project as well as comprehensive suggestions in my point of view of at what the future research in the SML should be aiming.

Chapter 2

Truck environment interaction

The Smart Mobility Lab (SML) was created for researchers and students who develop and test intelligent transportation solutions in simulated traffic situations with remote controlled model cars [11]. The lab is equipped with a motion capture system that is used for tracking the pose of objects with markers, who are inside a 6 metres by 8 metres area. In the centre of this 16 square metres area there is an image projector, which is used mainly to project maps of roads on the floor. The lab is also equipped with scaled trucks with markers on top that have been used to test different technologies; there are several computers that are programmed to communicate with these trucks and send commands so as to control them.

In this chapter we will explain in detail of what our simulation environment is composed. We will start by explaining the functionality of every component of the system. Followed by explanation of important assumptions that were made for our experimental setup. The whole system is composed of several blocks and the correct integration of these blocks is necessary in order to reach a better performance of the vehicle.

2.1. System components

To summarize, the system was composed of several sensors connected to an on-board computer, that runs on top of a scaled truck and sends to a central computer information of its current status. A detailed explanation with the components of the whole system is given in the following list:

The scaled truck

The vehicle to be used during the simulations is an 1:14 scaled R/C tractor truck version of the R620 Scania truck. The model features a ladder frame chassis with aluminum channels, 3-speed transmission, suspension with metal leaf springs and friction dampers, and rear axles equipped with differentials [24]. This scaled truck is the TAMIYA Scania R620 Highline model truck with trolly as shown in

Fig. 2.1, and will be referred to from now on as "the truck" for simplicity. These can

Figure 2.1. TAMIYA Scania R620 Highline model truck with trolly. Extracted from: [13]

be classified as non-holonomic vehicles. The torque of the motor is applied to the rear-wheels to provide angular velocity, while the steering commands are applied to the front-wheels.

The on-board computer

NI myRIO is an embedded hardware device designed specifically to help students design real, complex engineering systems. It also features NI industry-standard reconfigurable I/O (RIO) technology. The enclosed version of NI myRIO (NI myRIO-1900), as shown in Fig. 2.2, places three multiple I/O connectors, wireless capabilities, a dual-core ARM real-time (RT) processor, and a customizable Xilinx FPGA in the hands of students [17]. This computer runs LabVIEW applications and com-

Figure 2.2. National Instruments myRIO-1900. Extracted from: [9]

municate to a central computer, where the results of the processing and the current status of the truck are displayed.

The central computer

This computer sends the orders to start; stop in case of emergency; a reference velocity; and a desired map reference. It displays the results of the pose estimation, the ground truth of the pose, and some variables that describe the status of the vehicle; thanks to a network-shared public variables using Wi-Fi communication with the on-board computer.

Motion capture system for pose estimation

In order to keep track of the current state of the vehicle we will use the Qualysis Motion Capture system (to whom we refer as ''the MoCap'' or ''the GPS''). This systems is composed of 12 Oqus cameras which detect infrared markers. Therefore if we position some markers on the object, with the information of every camera, we are able to track the object current pose, i.e., x, y and z-position as well as roll, pitch and yaw-angular position with under millimetres error.

Gyroscope and infrared sensor

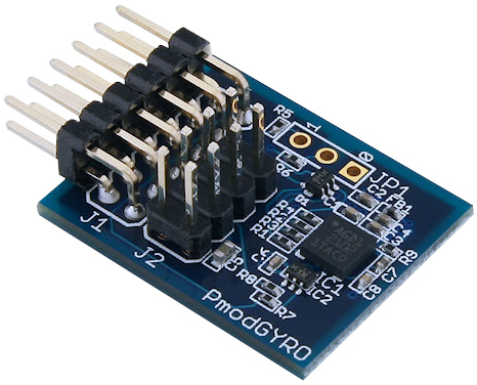
Measuring the yaw angular position of the truck is possible by using a gyroscope. The gyroscope to be used is the PmodGYRO, which is a peripheral module featuring the STMicroelectronics®L3G4200D, MEMS motion sensor. The L3G4200D provides a three-axis digital output gyroscope with built-in temperature sensor with 250/500/2000 dps selectable resolutions [8]. In order to detect possible objects or vehicles either on the front or on the sides of the truck we use the GP2Y0A02YK Sharp long distance measuring sensor. This sensor has a detection range between 20 and 150 cm.

Simulated sensors

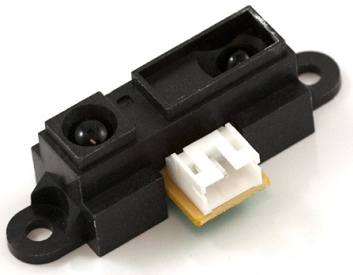
All these following sensors are simulated by using the MoCap system:

- Rotary encoders: the velocity of the vehicle can be computed directly as the derivative of the ground truth position in x and y over time.

- Camera: when the real webcam is not used the camera information is provided as a calculation of what the camera would see at its front in terms of trajectory points. This is given by the current pose of the truck. The trajectory points have to be inside of a circle of radius r and inside the angle of vision of the camera ψ. These parameters can be changed during simulation, depending on the specifications of the system.



(a) STMicroelectronics® L3G4200D. Extracted from: [19]



(b) GP2Y0A02YK Sharp infrared sensor. Extracted from: [21]

Figure 2.3. Gyroscope and infrared sensor

- Magnetometer: once again the MoCap provides the information, in this case of the ground truth yaw angle. When the truck is running with no GPS the error given by the gyroscope increases due to its drifting. The purpose of the simulated magnetometer is resetting this error when we reach some reference angle θ_{rst}.

In the following section we provide a detailed explanation of the importance of state estimation as a way to keep updated information at all time of each variables of the system.

Chapter 3

Truck pose estimation

In this chapter we will describe the basis of the kinematic model that describes the motion of our autonomous vehicle. Afterwards, we will show how to convert this model into a discretised version that can be used in computational algorithms for estimation. Later on, we explain the controllability issues of the system when following a trajectory. This section is followed by a description of recursive state estimation, where different algorithms are compared. This comparison will lead to define which algorithm suits better our project.

3.1. Truck motion

In our project we will use as a vehicle a Tamiya Scania R620 Highline model truck with trailer. This truck can be defined as a non-holonomic vehicle. In robotics a system is non-holonomic if the controllable degrees of freedom are less than the total degrees of freedom [10]. The model truck configuration is represented by the position x and y, by the orientation of its main body in the plane θ, and by the angle of the steering wheels ϕ. Initially, linear velocity inputs, v_1, and steering angular velocity, v_2, are used for motion control. The non-holonomic nature of the car-like robot is related to the assumption that the robot wheels roll without slipping. The scaled trucks have been developed by Scania; these scaled trucks are rear-wheel driving vehicles, whose kinematic model [15] is described as follows,

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \tan\phi/L \\ 0 \end{bmatrix} v_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} v_2 \tag{3.1}$$

The dynamics of v_2 can be neglected assuming that this control signal is fast enough to make instant changes in the steering angle. Moreover, we have to take into account that $\{\phi \in [-\pi/6, \pi/6]\}$, and hence, we can ignore the singularity at $\phi = \pm\pi/2$; with these approximations of the model, the state-space vector for the pose X becomes $X = [x, y, \theta]$ and for the control signals, $U = [v_1, \phi]$. The continuous-

time non-linear model of the non-holonomic car to be used during this thesis project is the one shown below,

$$f(X,U) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} v_1 + \begin{bmatrix} 0 \\ 0 \\ 1/L \end{bmatrix} \tan\phi \tag{3.2}$$

This model has to be linearised, because this result will be used in the future in some estimation techniques that require linearised models. One example is the Extended Kalman filter, which will be discussed further below.

3.1.1. Model linearization

Given the model in (3.2) we can use Taylor theorem in order to linearise around some operating point $X_0 = [x_0, y_0, \theta_0]$ and given a certain control input $U_0 = [v_{1_0}, \phi_0]$. Let us first define, the Jacobian matrix of F_X and F_U, respectively, as

$$J_{F_X}(x,y,\theta) = \begin{bmatrix} \frac{\partial \dot{x}}{\partial x} & \frac{\partial \dot{x}}{\partial y} & \frac{\partial \dot{x}}{\partial \theta} \\ \frac{\partial \dot{y}}{\partial x} & \frac{\partial \dot{y}}{\partial y} & \frac{\partial \dot{y}}{\partial \theta} \\ \frac{\partial \dot{\theta}}{\partial x} & \frac{\partial \dot{\theta}}{\partial y} & \frac{\partial \dot{\theta}}{\partial \theta} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -v_1 \sin\theta \\ 0 & 0 & v_1 \cos\theta \\ 0 & 0 & 0 \end{bmatrix} \tag{3.3}$$

and,

$$J_{F_U}(v_1,\phi) = \begin{bmatrix} \frac{\partial \dot{x}}{\partial v_1} & \frac{\partial \dot{x}}{\partial \phi} \\ \frac{\partial \dot{y}}{\partial v_1} & \frac{\partial \dot{y}}{\partial \phi} \\ \frac{\partial \dot{\theta}}{\partial v_1} & \frac{\partial \dot{\theta}}{\partial \phi} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ \tan\phi/L & (v_1(\tan\phi)^2 + 1)/L \end{bmatrix} \tag{3.4}$$

The new linearised state-space can be described as $\tilde{X} = X - X_0$ and $\tilde{U} = U - U_0$. The linear model can then be written as $f(X,U) - f(X_0,U_0) \approx f(\tilde{X},\tilde{U})$

$$f(\tilde{X},\tilde{U}) = J_{F_X}(x,y,\theta)\Big|_{X_0}(X - X_0) + J_{F_U}(v_1,\phi)\Big|_{U_0}(U - U_0) \tag{3.5}$$

$$f(\tilde{X},\tilde{U}) = \underbrace{\begin{bmatrix} 0 & 0 & -v_{1_0}\sin\theta_0 \\ 0 & 0 & v_{1_0}\cos\theta_0 \\ 0 & 0 & 0 \end{bmatrix}}_{\mathbf{A}_t} \tilde{X} + \underbrace{\begin{bmatrix} \cos\theta_0 & 0 \\ \sin\theta_0 & 0 \\ \tan\phi_0/L & v_{1_0}/(L\cos^2\phi_0) \end{bmatrix}}_{\mathbf{B}_t} \tilde{U}$$

Summarizing,

$$\mathbf{A}_t = \begin{bmatrix} 0 & 0 & -v_{1_0}\sin\theta_0 \\ 0 & 0 & v_{1_0}\cos\theta_0 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{B}_t = \begin{bmatrix} \cos\theta_0 & 0 \\ \sin\theta_0 & 0 \\ \tan\phi_0/L & v_{1_0}/(L\cos^2\phi_0) \end{bmatrix}$$

and for simplicity, the linearised variables are

$$\tilde{X} = \mathbf{X}_t = \begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \end{bmatrix} \text{ and } \tilde{U} = \mathbf{U}_t = \begin{bmatrix} v_1(t) \\ \phi(t) \end{bmatrix}$$

All the equations above are defined in continuous-time and the subscript t is used to describe this fact. Computations of electronic devices are governed by time clocks with a limited maximum frequency; this makes impossible to use continuous-time logic in these devices. Therefore, we need to translate all these models to their discrete-time version.

3.1.2. Discretization

In order to make a proper analysis in any embedded device with a sampling time T_s, we need to transform this continuous-time state-space system to its analogous discrete-time state-space system, as follows,

$$\mathbf{A}_k = e^{\mathbf{A}_t T_s} = \mathcal{L}^{-1}\{(s\mathbf{I} - \mathbf{A}_t)^{-1}\}\Big|_{t=T_s} = \begin{bmatrix} 1 & 0 & -T_s v_{1_0} \sin\theta_0 \\ 0 & 1 & T_s v_{1_0} \cos\theta_0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{B}_k = \mathbf{B}_t T_s = \begin{bmatrix} T_s \cos\theta_0 & 0 \\ T_s \sin\theta_0 & 0 \\ T_s \tan\phi_0 / L & T_s v_{1_0} / (L \cos^2\phi_0) \end{bmatrix}$$

$$\mathbf{C}_k = \mathbf{C}_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{D}_k = \mathbf{D}_t = \mathbf{0}|_{3x2}$$

After computing all these matrices we get state-space, at time k,

$$X_{k+1} = \mathbf{A}_k X_k + \mathbf{B}_k U_k \tag{3.6}$$

where,

$$\mathbf{X}_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} \text{ and } \mathbf{U}_k = \begin{bmatrix} v_{1_k} \\ \phi_k \end{bmatrix}$$

Here X_k is the pose at time k and U_k the control inputs; with these equations in our hands we are able to make predictions of what the pose X_{k+1} would be, given a set of control inputs U_k. In the following section we will use this model as the basis of our prediction for estimation.

3.2. Recursive state estimation: Gaussian filters

In many engineering applications and especially when working with control we need to observe the states of a system. In most real applications only the outputs of the plant can be measured, rather than the state vector itself. In such cases the state estimation process becomes important for control implementation and/or for monitoring applications [12]. State Estimation is a widely known problem, which is a very important part of fields such as probabilistic robotics. The central idea of it is to estimate the current state given sensor data; it is estimating quantities from information fetched by sensors that even though are not directly observable, can be inferred. The reason why state estimation is needed is that we cannot rely completely on information provided by sensors for several reasons; the data from the sensors might be either partial or corrupted by noise. The end goal of state estimation is to recover the current state given this data.

Let us illustrate the need of state estimation by using the example of trajectory control of a truck. This example is very important for explicative purposes for the upcoming chapter of this thesis. This is an introductory description of how our system is controlled. Let us assume that the truck has to move from its current pose A, $[x_k, y_k]$, to pose B, $[x_B, y_B]$, in a known map. We know the truck current velocity and its position with certain error, by using just a global positioning system (GPS). The current yaw angular position of the truck, θ_k is measured by a gyroscope; we can get the reference angular position as $\theta_{ref_k} = \text{atan2}(y_B - y_k, x_B - x_k)$, where

$$\text{atan2}(y, x) = \begin{cases} \text{atan}(y/x) & \text{if } x > 0 \\ \text{sign}(y)(\pi - \text{atan}(|y/x|)) & \text{if } x < 0 \\ 0 & \text{if } x = y = 0 \\ \text{sign}(y)\pi/2 & \text{if } x = 0, y \neq 0 \end{cases} . \tag{3.7}$$

Let us assume now that the velocity controller provides constant velocity. In order to reach the goal we have to calculate the error bearing to the goal B, i.e., $\theta_{E_k} = \theta_{R_k} - \theta_k$. A steering controller will use this error in order to generate the control signal to the wheels so as to decrease the error and to align the truck to the goal. The problem occurs when one of the sensors stops working properly. For instance, if we lose GPS data then we will lose our current position and then θ_{E_k} will be wrong. This will lead to wrong control inputs and hence the truck will fail to reach target B. That is just one of the reasons why it is so important to keep track of the state of the system.

State estimation is critical for several other reasons: one reason is that sensors are affected by different kinds of noises; another reason is that these sensors might fail during short time periods. Such failures during just a few milliseconds might cause system instability or simply an undesired behaviour. Additionally, state estimation provides some advantages to the system: control becomes easier avoiding radical changes in the states, which decreases behaviours that lead to instability; allowing better control actions to be selected. It is also sort of a testing block, where

undesired states can be detected; allowing other blocks to take action. Another advantage is that we could detect when a component fails; allowing the system to make certain actions to cope with it [6].

An algorithm or system that yields an estimate $\hat{x}(s)$ is called an observer or state estimator. The most general algorithm for calculating the belief (internal knowledge about the states environment of the robot) from measurements and control data is given by the Bayes filter [25]. Due to the fact that this algorithm calculates the belief at time t from the previous belief $t-1$, we can call it as a recursive filter. The other inputs of such filter are the last available measurements, z_t, and controls, u_t. For a more detailed explanation of the algorithm, please refer to [25]. In the following section a brief discussion is done about three popular recursive state estimators that are derived from the Bayes filter.

3.2.1. Kalman filter

Probably the best studied approach for implementing Bayes filters is the Kalman filter (KF). This technique was invented by Swerling (1958) and Kalman (1960), for filtering and predicting in the so-called linear Gaussian systems [25]. The KF provides to multi-sensor systems the possibility to combine dynamic low-level redundant data in real time [20]. The KF is part of an important family of recursive state estimators called Gaussian Filters, which are recognized as the most popular filters; they are based on the idea that beliefs can be represented by multivariate normal distributions. KF uses the statistical characteristics of a measurement model in order to recursively estimate data from different sensors that are optimal in a statistical sense. The recursive nature of the filter makes it appropriate for use in systems without large data storage capabilities [7].

In theory it is known that the Kalman filter is an optimal algorithm. However, this holds only if the used model is a very accurate replica of the real behaviour of the system; and if the noise is zero mean Gaussian noise. Additionally, the optimality holds if the observations are linear functions of the state; and if the next state is a linear function of the previous state. We know in practice that most of the cases that does not hold. However, although not optimal KF works quite well in several nonlinear applications.

Given a model in state-space representation, the state transition function, at time k, is,

$$X_k = \mathbf{A}_k X_{k-1} + \mathbf{B}_k U_k + e_k \tag{3.8}$$

where e_k is the process noise which is assumed to be drawn from a zero mean multivariate normal distribution with covariance

$$\mathbf{Q}_k = \begin{bmatrix} Q_1 & 0 & 0 \\ 0 & Q_2 & 0 \\ 0 & 0 & Q_3 \end{bmatrix} \tag{3.9}$$

i.e., $\mathbf{e}_k \sim \mathcal{N}(0, \mathbf{Q}_k)$. In the case of our non-holonomic vehicle, the state-space vector for the pose X is $X = [x, y, \theta]$ and for the control signals is $U = [v_1, \phi]$.

On the other hand, the observation model is described as,

$$z_k = \mathbf{C}x_k + \delta_k \tag{3.10}$$

where δ_k is the observation noise associated to each measurement, assumed to be zero mean Gaussian white noise with covariance

$$\mathbf{R}_k = \begin{bmatrix} R_1 & 0 & 0 \\ 0 & R_2 & 0 \\ 0 & 0 & R_3 \end{bmatrix} \tag{3.11}$$

i.e., $\delta_k \sim \mathcal{N}(0, \mathbf{R}_k)$. Both the state transition and the observation model have to be linear functions. This is due to the fact that when working with Gaussian random variables (RV) any linear transformation among these generate another Gaussian random variables. This is the main assumption of basic Kalman filters. Q_1 and R_1 are the associated covariances to x; Q_2 and R_2 are the associated covariances to y; and Q_3 and R_3 are the associated covariances to θ.

A brief description of the algorithm is done in appendix A. This description will work as a basis for further details of the implementation of the extended version of the KF, the EKF. Unfortunately, we know that state transitions and observations are mostly described as non-linear functions. For this reason, KF becomes non-optimal to the majority of the robotic problems. Relaxing this assumption of linearity decreases the uncertainty of the estimation.

Another approach that copes with non-linear dynamics, known as the Extended Kalman filter (EKF) is explained under some detail in the following section.

3.2.2. Extended Kalman filter

EKF eliminates the constraint of using linear functions to describe the state transition, g, and the observation, h. Let us now move from linear to non-linear state transitions and observations. In this approach the state distribution is propagated analytically through a linear approximation of the system around the operating point at each time instant; this linearisation is done via first order Taylor expansion. For this algorithm we need to approximate g and h to linear functions G and H, which are tangent to $g(U_k, \mu_{k-1})$ and $h(\mu_{k-1})$, respectively; U_k are the inputs and μ_k is the mean of the Gaussian that describes the states probability density function (pdf) of X_k.

In our case the only difference between the algorithms of the Extended Kalman filter and the Kalman filter is the nonlinear function $g(U_k, \mu_{k-1})$ that replaces $A_k\mu_{k-1} + B_kU_k$ [25]; the observation matrix becomes $h(\mu_{k-1}) = C_k\mu_{k-1}$, because C_k is already linear, in our case. This can be written as follows,

$$\begin{cases} \mathbf{X}_k = g(\mu_{k-1}, U_k) + \varepsilon_k \\ z_k = h(\mu_{k-1}) + \delta_k \end{cases} \tag{3.12}$$

This non-linear kinematic model (3.2) is linearised as explained in section (3.5) using Taylor expansions around some arbitrary linearisation point, $[U_k, \mu_{k-1}]$. These assumptions as discussed below will add uncertainty to the system. For more details please refer to appendix B, where the algorithm is described step by step.

This linear approximation, depending on the nature of the pdf, might result in less accurate estimates of the mean and covariance of the resulting random variable; this occurs for instance when the probability density function of the random variable that describes the state is highly non-Gaussian. These problems are dealt by using more advanced extensions of the KF that represent posteriors by using mixtures of Gaussians, as the multi-hypothesis EKF; this would be now a problem of trade-off between computational efficiency and having less uncertainties in state estimation.

Another factor to take into account is the degree of local non-linearity of the functions that are being approximated [25]; for instance approximating a sinusoidal function around very small values will be more accurate than for large values. When the linearisation of a function via Taylor expansion is not as accurate as we expect then some other approach for linearisation is needed. One approach that deals with this problem is known as the Unscented Kalman filter (UKF).

3.2.3. Unscented Kalman filter

The Unscented Kalman filter makes use of a different strategy so as to linearise functions. The idea is that it examines the function around selected points and provides a linearisation based on the results of these examined points. This approach requires no calculations of Jacobians. When the systems are linear the UKF and the simple KF are equivalent in terms of efficiency to estimate. However, when the systems are non-linear the UKF often provides better results in terms of estimation than the EKF, while having the same computational complexity.

This approach is not analysed further in this thesis project, but we encourage the interested reader to read [25] to get a better explanation of the advantages of other Kalman filter approaches. We wanted to discuss it to open the door for future researchers. We chose as framework to use the Extended Kalman filter for our thesis because the implemented version was reliable and the Taylor linearisation did not decrease the performance of the system.

It is important to add that all these state estimation theories work as a very important part of a whole system; without a reliable way to observe our system we will certainly not be able to drive autonomously. On the other hand, before the state estimation occurs the vehicle has to drive over a trajectory with some desired velocity; to accomplish such goal we are required to send control commands to the vehicle specifying how fast it should move and which direction it should take. Without an efficient way to send these command we will fail to reach autonomous driving.

That is the reason why the next chapter describes digital PID controllers, which is a very known approach that is widely used in industry. In our case, we will use two PID controllers in order to send the vehicle velocity as well as steering wheel

commands to the truck.

Chapter 4

Steering and velocity control

We require to design for our non-holonomic vehicle two controllers: one controller for the steering angle based on a forward desired position and we use another controller for the velocity given a reference velocity.

The non-holonomic model presented in Section 3.2 the steering angle ϕ was proved to be controllable by [15] in the time-invariant case of constant velocity v_1. Additionally, v_1 is controllable in the case of constant ϕ. Otherwise, the system mathematically is not controllable [15]. In our case we can see both variables as independent process. The reason is that we can assume that after the velocity controller reaches the desired velocity that the user inputs, we can give more priority to control ϕ rather than v_1. For simplicity, we select PID controllers, which are by far the most used controllers in industry.

4.1. PID control

The basic implementation of a PID controller in the *Laplace*−domain is given by the following equation:

$$F_{PID}(s) = \frac{U(s)}{E(s)} = K_m \frac{K_d s^2 + K_p s + K_i}{s} \tag{4.1}$$

where K_p, K_i, K_d are the proportional, integral and derivative gains respectively. K_m is an additional gain for the controller, that is useful, e.g., for unit scaling. $U(s)$ is the resulting control signal, and $E(s)$ the error signal. This equation is normally presented in *Laplace*−domain, but the real-time devices with a fixed sampling time can only handle a discrete-time equation. Therefore we have to discretise by initially changing from *Laplace*− to z−domain and finally changing from z−domain to k−domain, which is a discrete representation of time. The procedure is explained in the following section.

4.1.1. Discretization

From $\mathcal{S}$ to $\mathcal{Z}$

In order to map to the $z-$plane we can use different approximations:

- Backward difference

$$s \approx \frac{z-1}{zTs} \Leftrightarrow z \approx \frac{1}{1-sTs} \tag{4.2}$$

- Forward difference

$$s \approx \frac{z-1}{z} \Leftrightarrow z \approx \frac{1}{1-s} \tag{4.3}$$

- Tustin's method

$$s \approx \frac{2(z-1)}{T_s(z+1)} \Leftrightarrow z \approx \frac{1+sTs/s}{1-sT_s/s} \tag{4.4}$$

where T_s is the sampling time of the discrete-time system. It can be seen as the duration of a complete loop in the computer. In our case we used the Backward difference approximation. This leads to,

$$\begin{aligned} F_{PID}(z) = \frac{a_0z^2+a_1z+a_2}{b_0z^2+b_1z+b_2} = \frac{U(z)}{E(z)} = \frac{a_0+a_1z^{-1}+a_0z^{-2}}{b_0+b_1z^{-1}+b_2z^{-2}} \\ (b_0z^2+b_1z+b_2)U(z) = (a_0z^2+a_1z+a_2)E(z) \end{aligned} \tag{4.5}$$

From $\mathcal{Z}$ to $\mathcal{K}$

The time shifting property of the $z-$transform can be applied now,

$$\mathcal{Z}\{x[k-n]\} = z^{-n}X[z] \quad \text{and} \quad \mathcal{Z}[c_1x_1[k]+...+c_nx_n[k]] = c_1X_1(z)+...+c_nX_n(z) \tag{4.6}$$

The $z-$transform has a set of properties in parallel with that of the Fourier transform (and Laplace transform).

Applying (4.6) to (4.5) yields,

$$\begin{aligned} b_0u_k + b_1u_{k-1} + b_2u_{k-2} &= a_0e_k + a_1e_{k-1} + a_2e_{k-2} \\ u_k &= \frac{1}{b_0}[a_2e_{k-2} + a_1e_{k-1} + a_0e_k - (b_1u_{k-1} + b_2u_{k-2})] \end{aligned} \tag{4.7}$$

where

$$\begin{cases} a_0 &= K_iT_s + K_p + K_d/T_s \\ a_1 &= -K_p - 2K_d/T_s \\ a_2 &= K_d/T_s \\ b_0 &= 1/K_m \\ b_1 &= 1/K_m \\ b_2 &= 0 \end{cases}$$

After finding this control approach that can be handled by a real-time device, we need to continue to describe what kind of system we are going to use. The following chapter explains of what the system is composed and we highlight issues that might decrease control performance. We end up the next chapter with a brief explanation of how to tune the parameters of the controllers in order to cope with some issues of the system.

Chapter 5

Practical implementation

Due to the fact that we are working with a self-driving vehicle we need to fetch reliable information from external sensors about the car's current position, e.g, something similar to global position system (GPS) data. Additionally, information about the current motion of the vehicle given by internal sensors such as a gyroscope (to compute the yaw angular position) and rotary encoders (to compute the velocity of each wheel of the vehicle). From a explicative perspective let us first assume that we have no GPS data but we know precisely the starting location of the vehicle; also we have a good kinematic model of the system and the information provided by all the sensors of the vehicle have null associated error. If these ideal assumptions were true we could track the exact position of the vehicle on a flat surface at any time. However, we know there is neither a perfect model nor perfect sensors. But we can use an estimation approach to cope with this. Each one of these estimation tools have associated uncertainties, with which we as engineers have to struggle to cope. The performance analysis to be done in the following sections will consist in moving modularly from ideal to real-life situations.

5.1. Proprioceptive (internal state) sensors

In this category we include the rotary encoders for the velocity as well as the gyroscope. In a non-ideal framework, every sensor will add an error to the estimation of the position. The larger the error, the larger the probability of having erroneous estimations. Provided good resolution in the encoders, there are still some physical factors that cannot be noticed and even the best sensors will still have effects that we cannot neglect. One example of these is the loss of traction in the wheels causing an erroneous reading in the encoders, which is common under high speed tests or during sharp turns.

On the other hand, the gyroscopes output the angular velocity, which after integration results in angular position, provided of course that we know the initial angular position. In our case, we are interested in the yaw rotation (around the vertical axis), for the rotations of the vehicle. This sensor gives a well short-term

estimation of the yaw angle. However, this angle will tend to increase, even after calibration, which in the long run causes a large error. Due to this error the estimation of the position of the vehicle will be wrong. Some vehicles use a fusion of accelerometer and gyroscopes to calculate the angles of rotation pitch and roll; however, for yaw calculation this approach will not work because the axis of rotation of the accelerometer is parallel to the gravity axis.

There are several ways to cope with this drifting of the gyroscope, though. The option we chose was to use a magnetometer that detects the direction of the magnetic field at a point in space. After reaching some magnetic field at a known angle, the gyroscope could decrease its error to zero by replacing the output of the gyroscope already in radians, gyro_k by the angle to the point detected by the magnetometer, magn_k. This zero error yaw angular position measurement, θ_{zk}, can be easily computed in any real-time device by using the following equation in the output block,

$$\theta_{zk} = \text{gyro}_k - \text{error}_k$$

where,

$$\text{error}_k = \text{magn}_k - \text{gyro}_k$$

By using this approach we decrease the error of the current θ_{zk} for the future estimations. However, there is no way to recover for the error that this drifting caused to the previous pose estimation. This error is unavoidable as long as the sensors have some error.

5.2. Exteroceptive (external state) sensors

The main sensor to use in this case is the camera that would compose vision. Vision is a complex part of an autonomous system and requires a camera that detects features on a map; and a microprocessor or computer with enough computing power and software to recognize that detected object among a number of objects located in a database. In this computer all the sensor information will be fetched and processed in order to make decisions regarding the speed and steering angle of the vehicle. Let us assume that we have fixed landmarks with known identification (ID) and their (x, y) positions that are located on a map. Every time we find one of these landmarks during driving, we would have enough information so as to estimate the value of the bearing to it, as well as many other functionalities that could be used mainly to improve the driver safety. These landmarks can be traffic lights, curbs, street signs, etcetera.

5.3. Assumptions to be considered

Important assumptions were done under the development of the whole system. Initially, we set a fixed driving environment with selectable maps of roads, which

are composed of different types of lanes, dashed white lines, waypoints and curbs. Every lane has a default driving direction and it is composed of waypoints that are fixed and that are the reference to be followed by the vehicle. If the road has two directions every lane is separated by double white lines on the streets. The curbs delimit the borders of every road, except on intersections.

As we know the ground truth pose of the vehicle any time, $X_k = [x, y, \phi]_k$, we can also extract from the map the closet waypoint, WP_k, and then the car would be able to follow WP_{k+n} with a reference velocity, V_{ref} provided by the user. n is the amount of points ahead we see. Some special cases have to be taken into account in the design of the application, such as those cases at the end of a loop, when we reach the last waypoint, WP_N and we are required to continue following the next waypoints. The logic applied in this cases is to calculate the modulus after the division of k over N in order to get $\text{WP}_{iN+k} = \text{WP}_k$, where $i \in \mathbb{N}$.

5.3.1. On the estimation algorithm

Using as a basic assumption, the truck to be used in our project is equipped with proprioceptive (PC) and extereoceptive (EC) sensors. This in order to be able to perceive their own motion, as well as the state of the outside world. As mentioned above, our scaled truck will be equipped with a RTK-GPS-like sensor which measures x and y as z_x and z_y, respectively, scaled directly to metres in the x and y world-frame. A gyroscope will provide data to use in order to calculate the yaw angular position of the vehicle, with respect to a given initial yaw. The yaw angular rate $z_{\dot{\theta}}$ is given in radians/sec. However, we process this data in order to have a valid measurement of the yaw angular position, z_θ, in $\{-180, 180\}$ degrees. A simulated magnetometer will eliminate the inherent drift error of the gyroscope every time the ground truth of the yaw angle between $\{180° \pm 0.5°\}$ or between $\{0° \pm 0.5°\}$, i.e., when the magnetometer would detect the magnetic field of reference. We decided to use this approach in order to remove the drifting of the yaw angle in very specific angles (like north and south), giving more trust to the internal gyroscope of the vehicle.

These sensors generate the observation matrix $\mathbf{C}_k$ that maps the true state-space into the observed space. $\mathbf{C}_k$ equals the identity matrix because we can measure the pose (x and y) and the yaw angular position (θ) at every time, k.

GPS denied environments assumptions

We assume that while driving, we will keep track of the position of the vehicle using the MoCap data, as if it were RTK-GPS data. For this we added some noise to the signal of the MoCap so as to add an uncertainty to the signal. We assume additive white Gaussian noise with standard deviation equals to 3/14 centimetres; the 14 comes from the scaled size of the truck and 3 cms from typical values of commercial RTK-GPS. When the MoCap data is available the x and y position will be measured using the RTK-GPS. Whereas ϕ, the yaw angular position, will be

always obtained by reading the digital output of the gyroscope and resetting the drifting error depending on the magnetometer output signal.

During estimation of the pose, X_k, might occur that the MoCap data is lost because the camera cannot find the markers of the vehicle. When the signal is lost or when the MoCap cannot track the markers on the object we have to rely completely on the measurements of the real on-board sensor, the gyroscope. This means that there will be a higher uncertainty in the calculation of the velocity, because there are no on-board rotary encoders, as well as in the calculation of the yaw angle in the long-term due to its inherent drift.

It might also occur that during pose estimation we simulate that the GPS signal is not available, as if the vehicle were located in GPS denied environments. The rotary encoder, the camera and the magnetometer are simulated sensors in these applications and they depend strongly on the MoCap system to be considered as ground truth data. In case of simulated MoCap data losses these three sensors will keep using the MoCap data for testing purposes.

In an EKF terminology, the prediction is done by using as input the previous pose estimation. However, during the update phase of the EKF algorithm, the measurements are assumed to have the same values of the states calculated in the prediction phase. This is done in order to ignore the NaN measurements, in order to avoid numerical problems and sharp changes on the signal.

5.3.2. On the PID parameters selection

In this section we give a brief analysis on how to choose the gains K_m, K_p, K_i and K_d for the different controllers.

Steering angle PID control

The steering wheel angle is controlled by a digital PID that provides the steering angle command, ϕ_k, at time k as in (4.7). This approach uses an error signal e_{θ_k} equals to the reference signal, θ_{ref_k}, minus the current yaw angular pose θ_k. θ_{ref_k} is the angle from the current position in $\{x_k, y_k\}$ to the following waypoint on the map to follow x_{wp}, y_{wp}, i.e. $\theta_{ref_k} = \text{atan2}(y_{wp} - y_k, x_{wp} - x_k)$.

The proportional parameter of the PID controller, K_p, tells us how much of the error signal we want to project to the output of the controller. A really large value will give very reactive responses that might lead to an oscillatory behaviour when driving. On the other hand, a small value will result in slow responsiveness and the truck will not manage to converge to the desired yaw angular position of reference. The constant of the derivative action, K_d establishes the amount of prediction of the system behaviour and thus improves settling time and stability of the system. Its inherent sensitivity to measurement noise is something that we have to keep in mind [4] and that is why we should keep its value low. The integral term, K_i, will be necessary for the case when we need to turn with large steering angles. The need of having an integral term comes due to the fact that the steering command is

not exactly the value that the wheel will turn. Assuming we have a constant θ_{ref_k} if we set the command to be for instance $\phi_k = 30°$ and the future values of e_{θ_k} are still large that means that we are not converging to the reference value. The integral term will try to help us to converge to the desired reference, though. That takes us to one important issue of the steering angle control, saturation. When the integral part of the control signal ϕ_k increases for a long time the steering wheel will saturate.

When performing mathematical analysis, the final value theorem (FVT) is an easy way to compare expressions in frequency domain with those in time domain, as time approaches infinity. This is written as,

$$\lim_{k\to\infty} f(k) = \lim_{s\to 0} sF(s) = \lim_{z\to 1}(z-1)F(z) \tag{5.1}$$

Let us go back to the previous example, in this case $\phi_k = 35°$. As the vehicle cannot reach $\phi_k > 30°$, then $e_k > 0°$ for $k > 0$ until $k \to \infty$. Let us take the s-domain representation of the PID controller (4.1) and apply the FVT to a fixed step response $E(s) = \frac{\theta_d}{s}$.

$$\lim_{k\to\infty} f(t) = \lim_{s\to 0} sF_{pid}(s)E(s) = \lim_{s\to 0} \theta_d \frac{K_d s^2 + K_p s + K_i}{s} \approx \infty \tag{5.2}$$

This can be translated to an infinite control signal sent to the steering wheel when $k \to \infty$. Moreover, if the output of the controller is saturated and the reference suddenly changes to a negative value the control will be slow to react to this change because it has to recover from a very large value.

For all the mentioned above, it is important to write some code in the real-time program to avoid this saturation. In our case we limit the reference to $\theta_{ref_k} \in \{-30°, 30°\}$ as well as the control signal.

Velocity PID control

As in the previous case, the velocity is controlled by a digital PID controller. Its input is the error, $e_{v_k} = v_{ref_k} - v_k$; where v_{ref_k} is the user desired velocity and v_k is the observed velocity. The output of this controller is the input velocity command to the vehicle, v_{1_k}. It is important to mention that the relationship between the control signal v_{1_k} and v_k is not linear. Therefore, we need to have and integral term to be sure that v_1 reaches v_{ref_k} faster. We are flexible with settling times but the faster the better; but we want to avoid overshoots and all sort of oscillations. For this reason we want to evaluate the behaviour of adding a derivative term in the controller, that would act as a dampener on the control effort. The more the PI part of the controller tries to change the control signal, the more the D part counteracts the effort. This improves the overshooting.

In order to have a very good control, it is essential to have a good estimation and vice versa. On the other hand, the failure on one blocks will cause a failure

in the next step of the other block. These conflicts are one of the reasons why it is so hard to debug a whole observer plus controller system. This has to be taken into account when testing the system and proving the correct functionality of each individual block.

5.3.3. Considerations on vision

Let us now imagine that we want to follow an object while driving, which can be, e.g., a white dashed-line on the street or a car at the front. Therefore, we could use the bearing angle to such object as error information in order to control the steering angle. Vision in our project uses as landmarks the curbs of the streets and based on this data we can compute a trajectory to follow. This information is always given by the MoCap. Although some tests were run with a real camera, these are not used in this project because the results were not as fast as expected and they are left for future work at the lab.

With the information from the MoCap we developed a simulated camera that performs lane detection. One example is shown in Fig. 5.1 running the simulated camera on a map that replicates a simpler version of the Scania trail track located in Södertälje, Sweden. The shape was adapted to overcome lab constraints. This is due to the fact that it was not possible to have the real curves for a truck with such scale.

The simulation block is filled with data points of the map: the left and right curbs as well as the waypoints; additionally we use information of the current estimated position of the vehicle and the camera block provides as the output the data set of points that a camera would see as curbs at the front of the vehicle inside a region of certain radius and angle of vision. After this the block provides automatically the trajectory to follow as the center point of the detected trajectory. A video showing this result can be found at [1]. As default for the upcoming tests we use this approach in order to follow waypoints, unless something else is stated.

After explaining the most important assumptions that were used during the design of the test environment we are then ready to test the performance of the system. We give a detailed description of how the experiments were performed followed by a well documented discussion of their results.

[1] Lane detection for autonomous vehicles using virtual camera: http://youtu.be/1-ikhbv_hl8

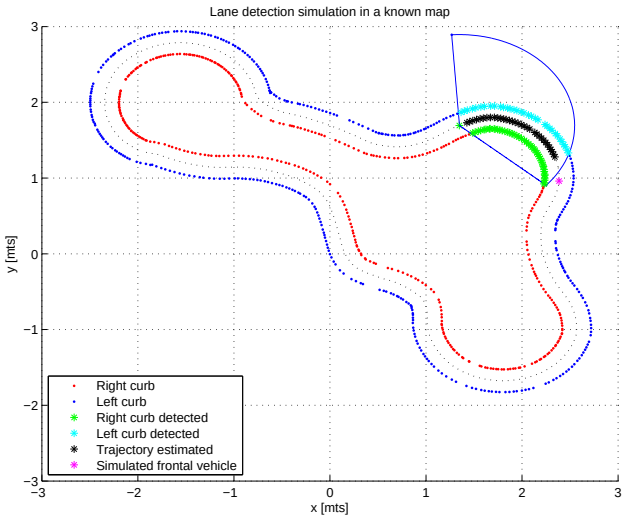


Figure 5.1. Lane detection for autonomous vehicles using virtual camera in MATLAB

Chapter 6

Performance evaluation

In the previous chapter we provided the theory for our multi-sensor approach for on-board autonomous truck navigation. We identified the issues of every sensor and we explained the assumptions that were done in order to reach the goals of driving on a pre-defined map. In this section we test our implemented system with an Extendend Kalman Filter (EKF) framework. As explained in theory one of the downsides of using EKF is the possible effect that the approximations could bring to the system. Therefore, we need to test that after discretisation and linearisation, the estimation is not strongly affected. We also need to take into account that the sensors might affect the estimation.

In this section we evaluate the performance of the EFK estimator and the controllers in quantitative terms, while understanding how the sensors are affecting these results.

6.1. Experimental setup

The performance evaluation is divided in three sections. Initially, we test the performance of the estimator, followed by the performance test of each controller. Finally, we test the closed-loop system; the truck pose is observed by the EKF estimator and the truck is controlled by the steering and speed controllers.

We quantitatively test our system on an on-board embedded device with the provided sensors using as ground truth for comparison purposes the information provided by the MoCap system as explained in 2.1. To this end, we implemented the EKF framework in myRIO, a Xilinx FPGA and dual-core ARM Cortex-A9 processor with a 1.6 GHz speed and two cores. The FPGA section is not widely used due to time constrains in the project, but it is widely recommended for future research, due to the high-speed advantages. Using this embedded device on-board of the truck let us control the vehicle in pose with a control loop of period, $T_s = 40$ ms.

When we say that the GPS data is available it means that the MoCap provides the pose of the truck with under millimetre error, in most of the map. However,

there were areas in the lab were we experienced what we called GPS outages; this means that the MoCap data is not available. For all the following tests we avoid these areas. The reason is simple: our system relies completely on the usage of the MoCap data in order to compute the encoder velocity. With an erroneous measurement of the velocity without any other velocity sensor, the estimation will fail.

In order to effectively analyse the performance of every main blocks of our system it is necessary to separate them and test them individually. One reason is to avoid conflicts between controller and estimation when analysing the results. The tests were run by attaching a gamepad to the main computer. This gamepad was used to send speed and steering inputs to the truck in order to test the estimator. We were able to send (positive and negative) velocity and speed references; we could send simulated GPS outages as well. Fig. 6.1 illustrates how the buttons were used; this is shown for future users to run their own tests.

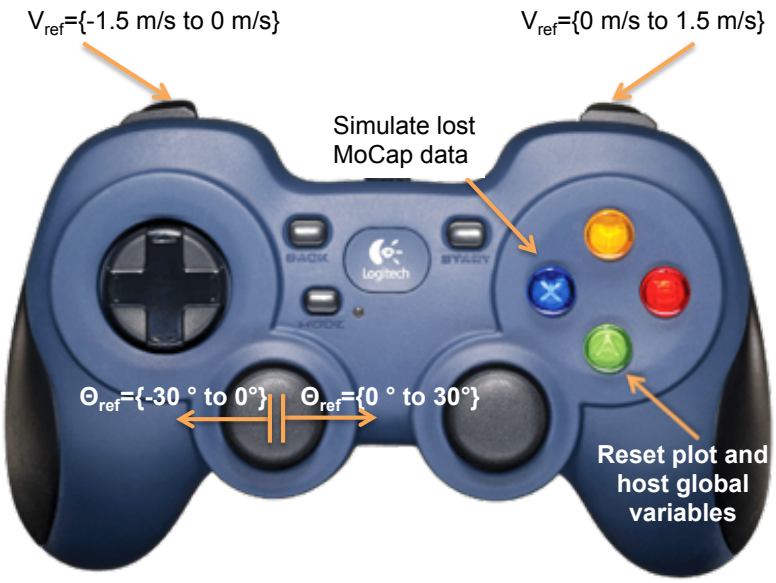


Figure 6.1. F310 Logitech Gamepad. Extracted from: [14]

On the other hand, so as to test the controller we sent fixed reference velocities, while the steering reference was given by the difference in angle between the waypoint to follow on the road-map and my current position, i.e., $\theta_{ref} = \text{atan2}(y_{wp} - y_k, x_{wp} - x_k)$. In order to understand the values of the results discussed in this section it is good to keep in mind that every parameter is scaled 14 times, which is the ratio real truck over modelled truck.

6.2. Experimental results

At this point we have a clearer image of how the experiments were set. In this section we will show the performance results of the estimation and followed by the performance of only controllers. Finally, these two parts are merged and tested.

6.2.1. Estimation performance

The initial setup uses the steering and velocity commands of the gamepad in order to control the truck. The estimation evaluation will consist on running the truck inside the lab and simulating GPS losses during some seconds or permanently. Keep in mind that when simulating the GPS outage, the MoCap keeps feeding the other simulated sensors with ground truth data, i.e., magnetometer and rotary velocity encoders. The initial setup is shown in Fig. 6.2

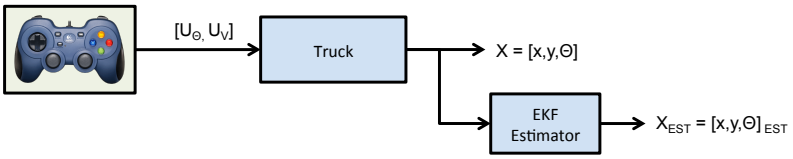


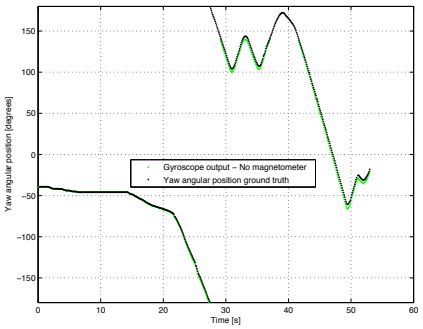
Figure 6.2. Block diagram for the system with controllers and EKF estimator

When making the initial test for the estimation performance we noticed in the results that the angle provided by the gyroscope was drifting. The mean error in the yaw angle for a 60 seconds loop was around 2.32 degrees. The problem with this drifting is that the general performance of the estimation was affected largely, because changes in the yaw angle affect directly $\hat{x}$ and $\hat{y}$ position.

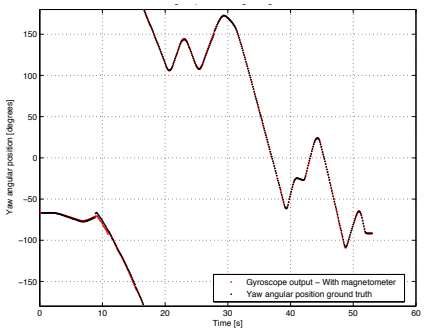
When doing any kind of pose estimation it is always necessary to have any kind of sensor that can eliminate the inherent drifting of the gyroscope. Some applications use accelerometers in order to eliminate the drifting by computing a mean value of the current angle and subtracting in it to the gyroscope measurement. However, this is not possible in $x - y$ applications where the angle to estimate is the yaw. This is due to the fact that the accelerometer cannot be used to compute the yaw angle.

However, we compensated for this error by using a simulated magnetometer. The purpose of the magnetometer was to detect certain headings, and use these values to decrease the drifting error the gyroscope accumulated. The magnetometer send a signal when the truck yaw angular position was between $\{180° \pm 0.5°\}$ or between $\{0° \pm 0.5°\}$. The results can be seen in Fig. 6.3

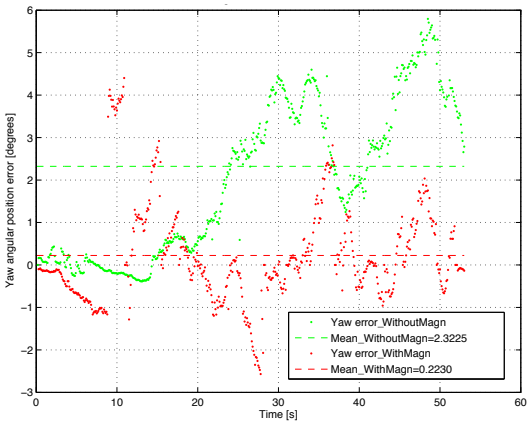
As it can be seen in the results the measurements provided by the gyroscope will be more reliable when a magnetometer is used. This magnetometer reduced the mean error to around 0.22 degrees.



(a) Not using magnetometer



(b) Using magnetometer



(c) Error comparison between both implementations

Figure 6.3. Yaw angular position given by the gyroscope

After providing reliable pose measurements as input to our EKF estimator, we are now ready to test our application. For this reason, we made a simple test by fixing the steering angle and the velocity references in order to test the behaviour of the system when there is a GPS outage. The results of this initial test can be seen in 6.4(b). The results show that when we are driving with circular motion and constant reference values the drifting of the estimation still appears, as it is clear just after three loops.

Calibration method 1: Sine function

The problem of unmanned ground vehicles equipped with GPS was considered in [16]. In this paper the authors propose a method that improves such drifting. This method aims to solve the problem of the current bias in yaw when there is a GPS outage. A typical problem in our estimation is the lack of a very good measurement of the real steering angle. The steering angle command given by the

gamepad or the controller output can range between -30 and 30 degrees. This relationship between the steering commands and the real steering output is highly nonlinear. The real steering angle depends on the vehicle angular velocity, friction and even the battery.

Therefore, the estimation will tend to drift even as soon as the outage occurs. The method proposed in [16] uses a calibration routine as a sine function, as

$$\hat{\beta} = \beta + \epsilon \sin(\beta + \gamma) \tag{6.1}$$

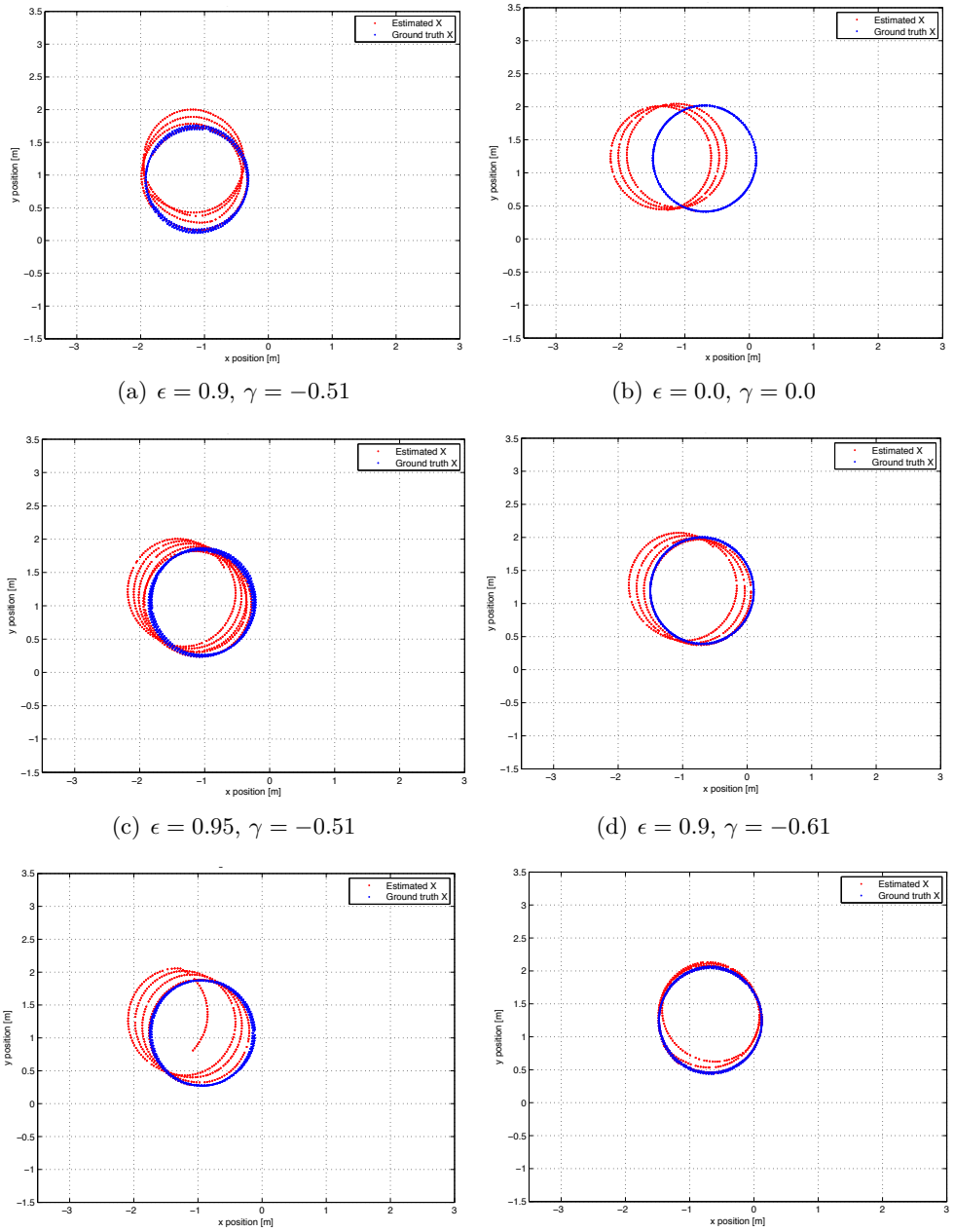
where β is the yaw angular position of the vehicle, $\hat{\beta}$ is the calibrated yaw and ϵ and γ are parameters to calibrate the yaw. In order to calibrate these parameters, we need to know first of all what the drifting angle is. For the first run we set the maximum negative $\theta_{ref} = -30°$ as steering angle and $V_{ref} = 0.5$ m/s as velocity references, which yields clockwise rotation. During this time we pressed the GPS outage simulator. The procedure to tune ϵ and γ in order to decrease the drifting was as follows: firstly, by fixing γ and then moving ϵ from -2 to 2, until the best performance was found; the results for this case are shown in the first column in Fig. 6.4. Afterwards, we fixed ϵ to the previous found value and then varying γ from -180 to 180 degrees. A hint in order to find γ easier was to trace two tangents to the initial bearing angles of the estimation and the ground truth after the GPS outage. The angle between these two tangents will be a good approximation for γ. The best result was obtained for $\epsilon_n = 0.9$ and $\gamma_n = -0.45$ as shown in Fig. 6.4(f).

This calibration was done for the positive case as well, while keeping the same reference velocity. The method to find the values was as mentioned above. For this case we found $\epsilon_p = 0.8$ and $\gamma_p = 0.48$. After finding $\epsilon_{p,n}$ and $\gamma_{p,n}$ for both cases, we proceed to make these functions linearly dependent to the steering angle, in order to improve the estimation, as shown in Fig. 6.5. In the future these functions can be changed to polynomials of higher order, e.g., sigmoidal functions. This in order to reach better approximations for middle values.

These parameters were tested by driving with both positive and negative steering angles. As it can be seen in Fig. 6.6(a) before calibration the estimation drifted largely when driving clockwise and anticlockwise. However, after calibration the results are improved as shown in Fig. 6.6(b). It is important to add that if the linearization step mentioned above would not have been done the result would have been good just for one value of steering angle (the largest possible value), while for any other case the estimation would have drifted, even at zero steering angle.

Calibration method 2: Rotation matrix

A similar approach is introduced in this section. It could be seen in the previous method that finding the calibration parameters was not straight forward. In order to calibrate in this case we use the same test environment as described above. This time we fetch the data of this test and ran it in Matlab. The purpose of doing this is to find the whole degree of rotation of the drifting. In order to obtain the real drifting angle we have to know our position in the time instant when we lose the



(a) $\epsilon = 0.9$, $\gamma = -0.51$

(b) $\epsilon = 0.0$, $\gamma = 0.0$

(c) $\epsilon = 0.95$, $\gamma = -0.51$

(d) $\epsilon = 0.9$, $\gamma = -0.61$

(e) $\epsilon = 1.0$, $\gamma = -0.51$

(f) $\epsilon = 0.9$, $\gamma = -0.45$

Figure 6.4. Calibrating first ϵ and then γ so as to eliminate drifting in the yaw angular position

signal. Afterwards, in order to find ϵ we just have to rotate the ground truth pose data points until matching the estimated pose data points. This can be done by

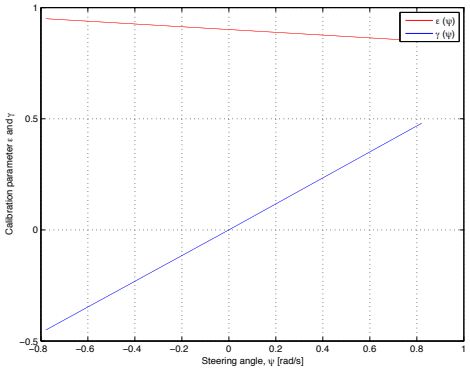
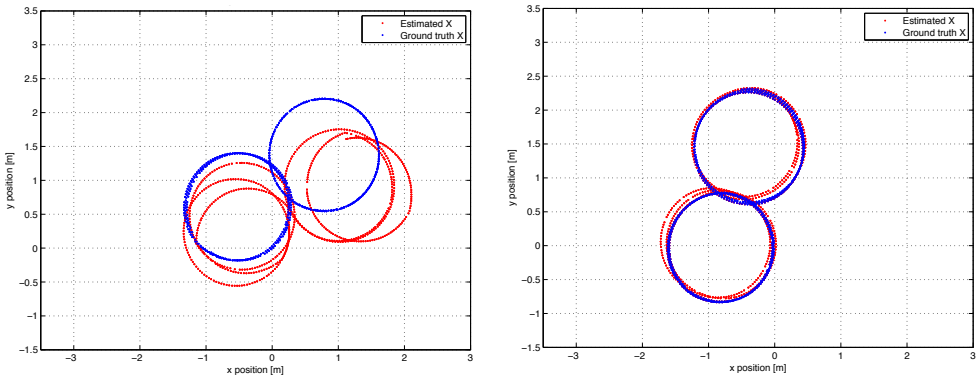


Figure 6.5. $\epsilon(\psi)$ and $\gamma(\psi)$ as linear functions of the steering angle



(a) Before calibration

(b) After calibration

Figure 6.6. Multiple loops clockwise and anticlockwise

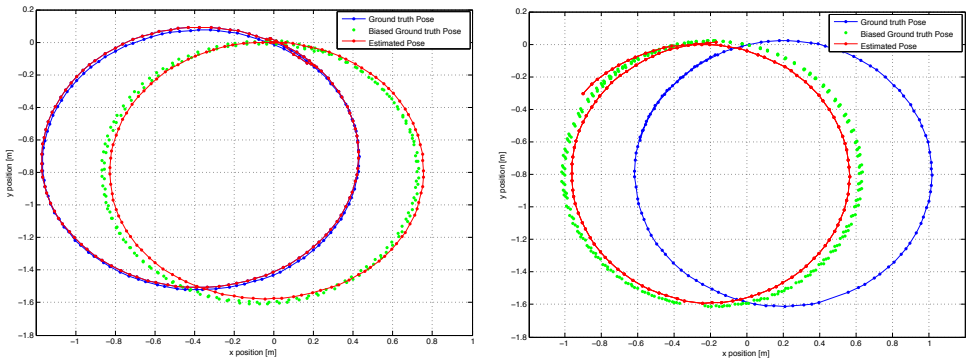
using the rotation matrix,

$$\mathbf{R} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}.$$

This means that given a set of $\mathbf{X} = [x, y]$ data points we could find their pairs, $\mathbf{X}_{rot} = [x_{rot}, y_{rot}]$, rotated α radians as follows,

$$\mathbf{X}_{rot} = \mathbf{R}\mathbf{X} \tag{6.2}$$

After rotating our ground truth pose data points we obtained our ‘‘biased ground truth’’, that emulates the estimated pose. The results are shown in Fig. 6.7. Now, we found out that the best value in theory for the drifting is around $\alpha = \pi/8$ for clockwise rotation. Whereas $\alpha = -\pi/4$ was obtained for anticlockwise rotation. As done before, we found a linear relation between α and the steering angle in order to provide better results for values between those values that were tested.

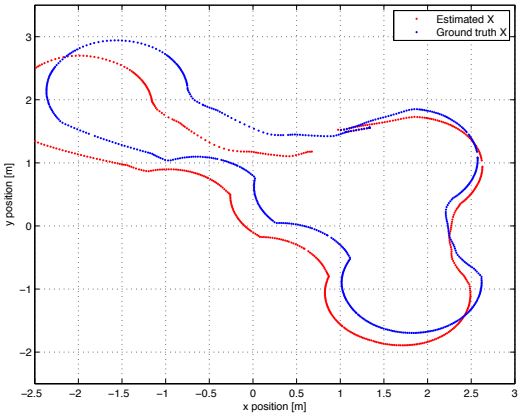


(a) Clockwise rotation (b) Anticlockwise rotation

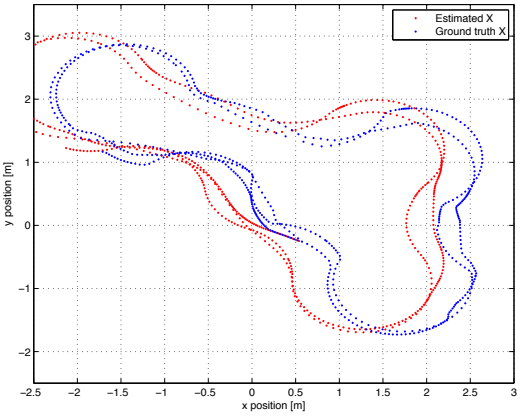
Figure 6.7. Determining drifting in the yaw angular position

Either using (6.1) or (6.2) provides better results to the estimation. However, the results found by using the second approach were slightly better, as well as simpler to tune. All the future estimations use the second calibration method. We use $\alpha(\psi)$ by subtracting it to the current yaw angular position in the EKF estimation algorithm, only during GPS outages.

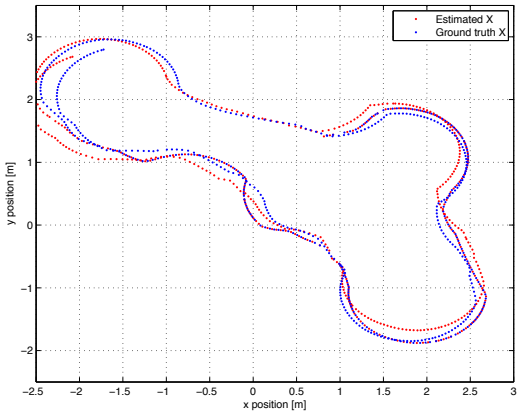
As driving in circle is not the main purpose of an autonomous truck, we used non-constant steering angle and velocity references in order to have a more complex test. In this test the purpose was to test the capacity of the estimation when driving 100% with no GPS signal before and after calibration. Without calibration the results would be as shown in the upper side of Fig. 6.8, where drifting will be very large even after the first loop. Whereas, after calibration the drifting is reduced even for multiple loops. If the GPS data is available during less than 10% of the loop the estimation improves largely as seen in 6.8(d), where the truck drove for a little bit more than five loops.



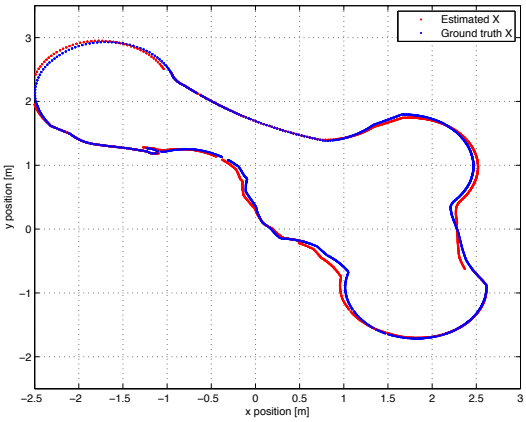
(a) Single loop driving 100% with no GPS before calibration



(b) Multiple loops driving with 100% GPS outage before calibration



(c) Multiple loops driving 100.00% GPS outage after calibration



(d) Single loop driving 92.32% GPS outage after calibration

Figure 6.8. Driving autonomously through the road map

The individual block for the EKF estimation was proven to be functional for fixed and variable steering and speed commands. The problems associated to the sensors and to the real values of the input signals (steering angle and velocity) of the estimator were compensated, decreasing the error in large extent. In the following section we have to analyse the performance of the controllers before merging both parts of the system.

6.2.2. Controllers performance

In this section the performances of the steering controller and the velocity controller are evaluated. Initially, the velocity PID controller gains, K_m, K_p, K_i, were tuned. After reaching the required performance for the velocity controller we tuned the steering controller gains. Finally, we test the two controllers working together while driving and following some waypoints on the road-map.

Velocity controller performance

The test for the velocity controller consisted in following waypoints on the road-map while changing the reference velocity. The block diagram for this case is shown in Fig. 6.9. Additionaly, for the implementation of the controller it was necessary to low-pass filter the observed output velocity signal. This because the velocity calculated by the simulated rotary encoder for velocity has a very noisy behaviour. After this filtering the controller improved largely its performance.

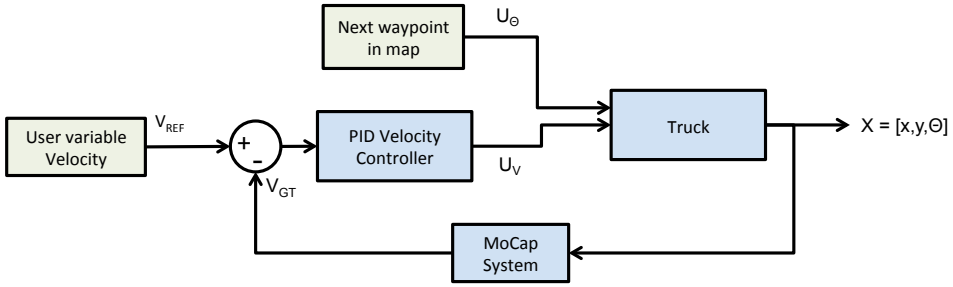
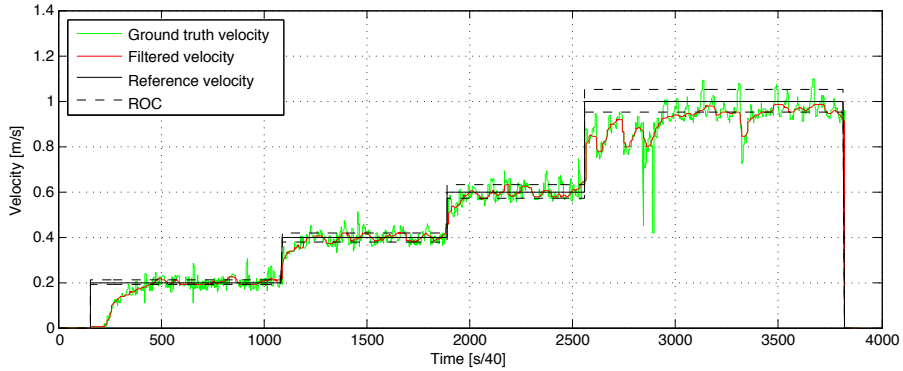


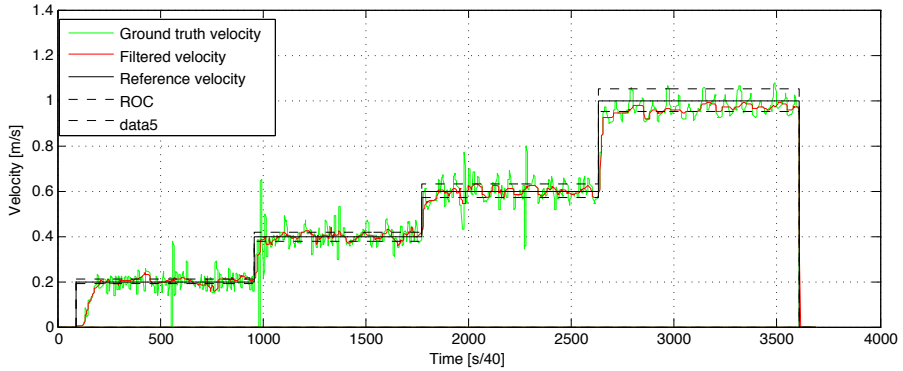
Figure 6.9. Block diagram for the system with velocity controller and fixed steering reference

As design specification we were not looking for a very fast controller but a controller that avoids all sort of overshooting or damping in the controlled variable. Several values of the controller gains were tested in order to reach such specifications. K_p is chosen as the maximum possible value before the response become oscillatory. The value of K_i has to be small enough to avoid oscillations around the desired velocity but large enough to reach such velocity. The oscillations are easy to detect just by hearing when the truck is moving, because it has a shaky behaviour. The selection of K_d or even its use was a big question mark. After testing several values of K_d there was not a clear answer to it because most of the values in a range between 0.05 and 0.2 provided very unstable performances. We decreased K_d to 0.01 and we found a good example of the importance of K_d in the velocity control. The comparison between well tuned PI and PID controller can be seen in Fig. 6.10.

It is observable that in the PI controller the settling time is larger and during $V_{ref} = 1$ the performance is quite violent, going rapidly out of the convergence region. Therefore, we selected $K_p = 0.8$, $K_i = 0.4$ and $K_d = 0.01$ as the parameters of our velocity controller



(a) $K_p = 0.8$, $K_i = 0.4$, $K_d = 0.0$



(b) $K_p = 0.8$, $K_i = 0.4$, $K_d = 0.01$

Figure 6.10. Velocity controller PID vs PI performance

Steering controller performance

The test for the steering controller was quite similar as the velocity control test. In comparison to the velocity controller test, the reference cannot be provided online by the user but fixed already to follow waypoints on the road-map; the block diagram for this case is shown in Fig. 6.11.

In this case we show the results for different tuning gains. Initially, we set $K_d = K_i = 0$ and vary K_p until reaching a control signal avoiding oscillations. We require a fast controller in this case, otherwise we would miss the next waypoint in sharp curves.

We needed to design a fast reaction for controlling the steering angle and a lower reaction controller where high frequency oscillations could be filtered out in order to avoid drastic changes in velocity. Selecting $K_p = 0.8$ together with a smaller value of $K_d = 0.01$ than before to permit faster changes and $K_i = 0.2$ we had the best response for our steering controller. To make the analysis even better we connected the previous velocity controller to our steering controller, the performance is shown

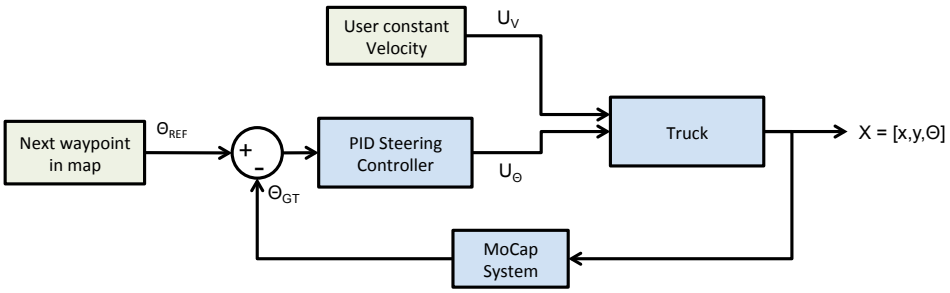


Figure 6.11. Block diagram for the system with steering controller and fixed velocity reference

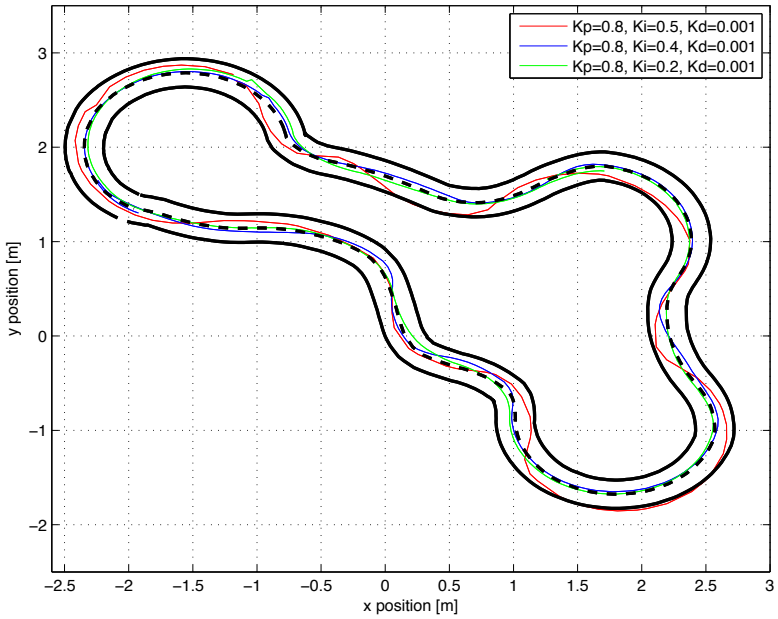


Figure 6.12. Steering control and velocity control performance altogether

in Fig. 6.12.

At this point, the EKF estimator as well as both controllers were proven to be functional and adequate for the needs of the system. Now, we have to integrate all the block together.

6.2.3. Integrated system performance

After both controllers were tuned and after testing the correct behaviour of our state estimator we can proceed to integrate them all in a whole system. This integrated system is shown in Fig. 6.13.

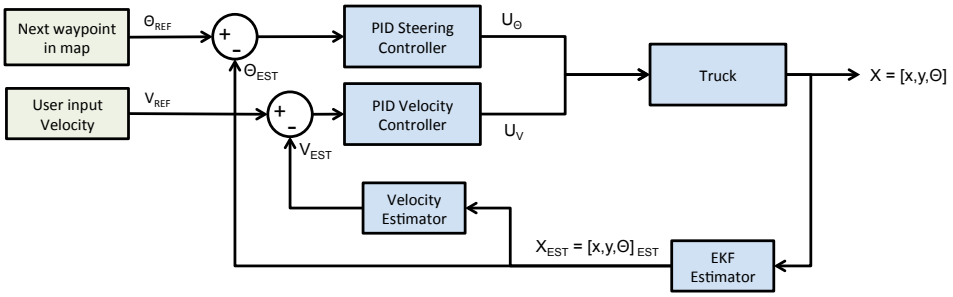


Figure 6.13. Block diagram for system with controllers and EKF estimator

It is important to add that in Fig. 6.13 the so-called ''Velocity estimator'' is our simulated rotary encoder for velocity whose output is connected to a low-pass filter in order to avoid high-frequency components in the velocity signal. Two important tests were done in this case. In both tests we would drive in the map following the waypoints with the help of the discussed controllers.

Test 1: Short-time GPS outages

In the first test we simulate short GPS losses while driving. The result is shown in Fig. 6.14.

We can see how the estimation works pretty well. A very important factor to take into account is that after recovering the signal the estimation comes back quickly to the real value. Therefore, we are able to drive with no problem for very long time with this approach.

Test 2: Long-time GPS outages

In this test we simulate completely lost signal of the GPS. The purpose of this test is to see how good the behaviour is under one loop. Especially if the drifting affects the behaviour under one complete loop. The results is shown in Fig. 6.15, where we illustrate the starting point of the GPS outage.

We can see in the simulation results that it is feasible to drive one loop with no GPS signal. Indeed, we can drive during several loops with the downside that after some loops we might leave the road indefinitely. The problem in this case is that we will still have a small error being accumulated even though we are calibrating the gyroscope; we are using a magnetometer for decreasing the error of the gyroscope measurements; and even further calibrating for decreasing the initial bearing error

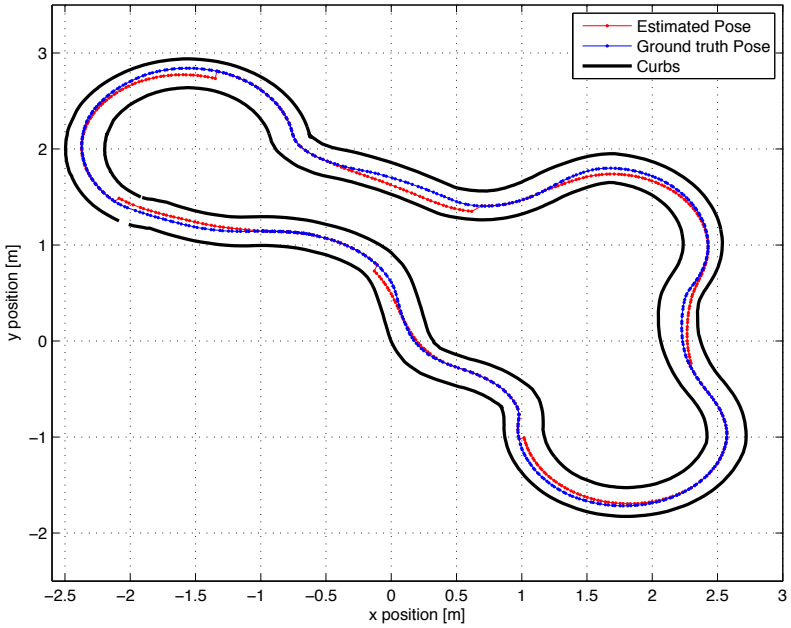


Figure 6.14. Driving autonomously through the road map and simulating short-time GPS outages in different tunnels

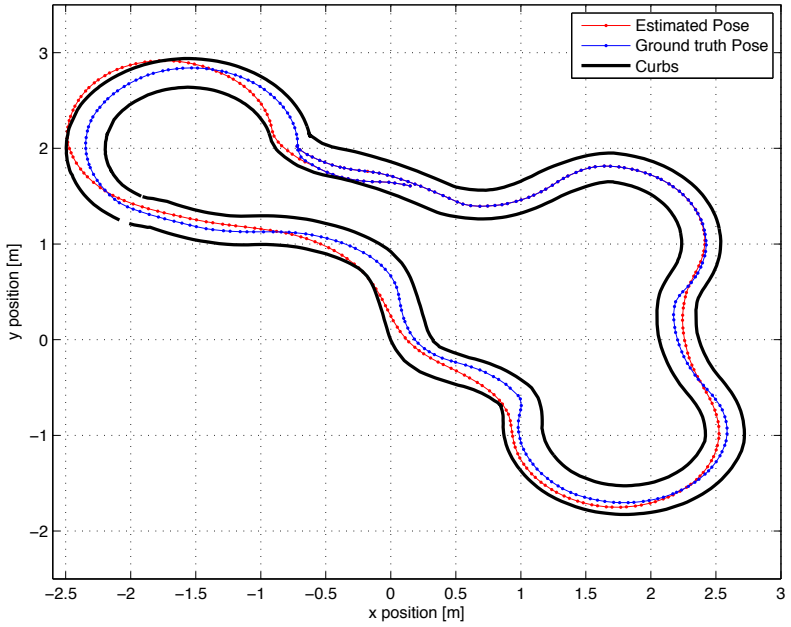


Figure 6.15. Driving autonomously through the road map and simulating complete GPS outage in a very large tunnel

when losing the GPS signal. This error is unavoidable with the equipment used in this project, but it can be improved by using other sort of sensors, like vision. These tests were run in the SML and a video showing their results is available at [1].

The vision performance chapter will be on hold for future research that can hopefully compensate this drifting after a large number of loops.

[1] On-board recursive state estimation for dead-reckoning in an autonomous truck: `http://youtu.be/nXpa6U7yVSU`

Chapter 7

Discussion and conclusion

In this thesis work, we presented an Extended Kalman filter framework for dead-reckoning in autonomous trucks equipped with a gyroscope and simulated GPS, magnetometer and velocity rotary encoders. We designed proper discrete-time controllers, one for the steering angle and one for the velocity. These controllers were designed to reach certain criteria, which were proven to be met in the performance evaluation. During the analysis of the velocity controller it was shown the importance of using velocity low-pass filtering in order to improve the controller performance. We took into account that due to the fact that both controllers have integral terms it is important to bound the control signals to the real values that the steering wheel and the motors can reach. This avoids wind-up effects in the controller.

The estimation itself worked very well after dealing with previously detected sensor issues. The gyroscope was calibrated in order to decrease the inherent physical drifting. This drifting was not eliminated in the first test it required also the help of a simulated magnetometer to eliminate this error during certain headings. After fixing the gyroscope measurements and ensuring that the velocity given by the encoder was properly filtered we moved to the control signals to verify that the physical values corresponded to the commands. We realised that it was not the case. Therefore, when estimating during GPS outages there was an initial error in yaw angular position that was propagated as a biased value during the whole estimation. This was in part given by the difference between the real steering angle command and the command sent by the real-time system. A previously studied bearing pre-calibration method is tested and proved functional. This worked as the initial idea to a new method that was introduced. This method used as a basis the rotation matrix and a comparison between the ground truth data and the estimated data after running one loop with no GPS signal. The method was also proven functional. Both methods improved the estimation accuracy by reducing the error of drifting associated to the bearing error when a GPS outage is reported for different driving situations. A weakness of both methods is the lack of robustness when the truck is turning with small angles. In the future, an auto-calibration method to find

the parameters to eliminate this bearing error would improve the results. Another idea might be to suggest polynomial functions of higher order in order to reach better approximations for middle values when creating the function that relates the parameters and the steering command.

Finally, it is important to mention that the test framework created in this thesis can be extended in the future for more complex situations, including distributed autonomous vehicles under real-life situations. The embedded device used in this project was proven to be very reliable in terms of processing power and meeting the desired time period; however, in some situations the wireless communication failed for unknown reason, which forced the program to abort. Nevertheless, when communications did not fail the real-time tasks were completed periodically in a very deterministic way, which helped us control the system with no problem. A special tip for the future users of myRIO is the importance of initializing correctly the network-published shared variables and to fully understand how they work before using it.

7.1. Applications and research outlook

This project forms a basis to provide a test platform for future research in the Smart Mobility Lab at KTH for autonomous truck driving with the focus on long-term navigation during GPS outages. However, this is just the beginning for soon-to-be very complex driving situations using several embedded systems. The usage of vision as discussed in the theory might improve vastly the control performance of the system provided that the real-time system has enough processing power to handle vision.

There are plenty advantages of vision in vehicles that belong from level one to four of autonomy as described in 1.1.1. The advantages are the number of applications that can be added to commercial vehicles in order to improve safety. Some examples are listed as follows:

- Lane detection
- Traffic sign recognition
- Light source recognition
- Vehicle detection
- Pedestrian detection
- Free space information
- Road surface information

Each one of these functions have as a main source of information vision. In this project we introduce the usage of lane detection in modelled trucks in the SML by

using simulated camera. A test platform was created in order to simulate camera data in front of the vehicle and to perform lane detection. This simulated block had as result datasets of points that worked as a trajectory to follow, in the shape of waypoints. Future research can be addressed to adapt this block to directly handle real application by using cameras.

In the future lane detection can be tested initially in a fixed environment, i.e., just an image of the road that will be processed in order to extract the curbs of the street. After this extraction such points will be used in order to make an estimation of the trajectory to follow. After successful completion of this algorithm we can try it directly on-board of the truck using a camera, while driving. This information will be analysed in the on-board computer and we will see if it is a feasible option in terms of CPU processing power. If it is not computationally possible we might try doing the processing in the main computer by sending the image through the network in a fixed time and then returning the detected trajectory to the vehicle as a simple array. The problem that has to be studied in this second option is how much the time delays affect the system performance.

The final goal is to have vision that: takes images as fast as possible without affecting the control performance; these images of the road will be then processed online, either in the on-board unit or in the central computer, and then we could detect edges with some vision algorithm, for instance a canny edge detector. These data points will be processed in order to deliver a dataset of waypoints as the one we are simulating in our fixed-maps, that will be our trajectory to follow.

Appendix A

Kalman Filter algorithm

In the following section the most important steps of the mathematical derivation for the implemented KF are briefly shown.

Prediction

$$\overline{\mu}_k = \mathbf{A}_k \mu_{k-1} + \mathbf{B}_k u_k \tag{A.1}$$

$$= \begin{bmatrix} \overline{x} \\ \overline{y} \\ \overline{\theta} \end{bmatrix}_k = \begin{bmatrix} x_k + T_s v_{1_k} \cos\theta_0 - 1.0 T_s v_{1_0} \theta_k \sin\theta_0 \\ y_k + T_s v_{1_k} \sin\theta_0 + T_s v_{1_0} \theta_k \cos\theta_0 \\ \theta_k + (T_s v_{1_k} \tan\phi_0)/L + (T_s \phi_k v_{1_0} (\tan^2\phi_0 + 1.0))/L \end{bmatrix} \tag{A.2}$$

$$\overline{\Sigma}_k = \begin{bmatrix} \overline{\Sigma}_{11} & \overline{\Sigma}_{12} & \overline{\Sigma}_{13} \\ \overline{\Sigma}_{21} & \overline{\Sigma}_{22} & \overline{\Sigma}_{23} \\ \overline{\Sigma}_{31} & \overline{\Sigma}_{32} & \overline{\Sigma}_{33} \end{bmatrix}_k = \mathbf{A}_k \begin{bmatrix} \Sigma_{11} & \Sigma_{12} & \Sigma_{13} \\ \Sigma_{21} & \Sigma_{22} & \Sigma_{23} \\ \Sigma_{31} & \Sigma_{32} & \Sigma_{33} \end{bmatrix} \mathbf{A}_k^T + \mathbf{Q}_k \tag{A.3}$$

$$\begin{aligned}
\overline{\Sigma}_{11} &= \Sigma_{11} + Q_1 - \Sigma_{31} T_s v_{1_0} \sin\theta_0 - T_s v_{1_0} \sin\theta_0 (\Sigma_{13} - \Sigma_{33} T_s v_{1_0} \sin\theta_0) \\
\overline{\Sigma}_{12} &= \Sigma_{12} - \Sigma_{32} T_s v_{1_0} \sin\theta_0 + T_s v_{1_0} \cos\theta_0 (\Sigma_{13} - \Sigma_{33} T_s v_{1_0} \sin\theta_0) \\
\overline{\Sigma}_{13} &= \Sigma_{13} - \Sigma_{33} T_s v_{1_0} \sin\theta_0 \\
\overline{\Sigma}_{21} &= \Sigma_{21} - T_s v_{1_0} \sin\theta_0 (\Sigma_{23} + \Sigma_{33} T_s v_{1_0} \cos\theta_0) + \Sigma_{31} T_s v_{1_0} \cos\theta_0 \\
\overline{\Sigma}_{22} &= \Sigma_{22} + Q_2 + T_s v_{1_0} \cos\theta_0 (\Sigma_{23} + \Sigma_{33} T_s v_{1_0} \cos\theta_0) + \Sigma_{32} T_s v_{1_0} \cos\theta_0 \\
\overline{\Sigma}_{23} &= \Sigma_{23} + \Sigma_{33} T_s v_{1_0} \cos\theta_0 \\
\overline{\Sigma}_{31} &= \Sigma_{31} - \Sigma_{33} T_s v_{1_0} \sin\theta_0 \\
\overline{\Sigma}_{32} &= \Sigma_{32} + \Sigma_{33} T_s v_{1_0} \cos\theta_0 \\
\overline{\Sigma}_{33} &= \Sigma_{33} + Q_3
\end{aligned}$$

Update

$$\mathbf{K}_k = \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix}_k = \overline{\Sigma}_k {\mathbf{C}_k}^T (\mathbf{C}_k \overline{\Sigma}_k {\mathbf{C}_k}^T + R_k)^{-1} \tag{A.4}$$

$$= \begin{bmatrix} k_{den} + \overline{\Sigma}_{22}\overline{\Sigma}_{33}R_1 - \overline{\Sigma}_{23}\overline{\Sigma}_{32}R_1 & -\overline{\Sigma}_{12}\overline{\Sigma}_{33}R_2 - \overline{\Sigma}_{13}\overline{\Sigma}_{32}R_2) & \overline{\Sigma}_{12}\overline{\Sigma}_{23}R_3 - \overline{\Sigma}_{13}\overline{\Sigma}_{22}R_3 \\ -\overline{\Sigma}_{21}\overline{\Sigma}_{33}R_1 - \overline{\Sigma}_{23}\overline{\Sigma}_{31}R_1 & k_{den} + \overline{\Sigma}_{11}\overline{\Sigma}_{33}R_2 - \overline{\Sigma}_{13}\overline{\Sigma}_{31}R_2) & -R_3(\overline{\Sigma}_{11}\overline{\Sigma}_{23} - \overline{\Sigma}_{13}\overline{\Sigma}_{21}))/K_{den} \\ \overline{\Sigma}_{21}\overline{\Sigma}_{32}R_1 - \overline{\Sigma}_{22}\overline{\Sigma}_{31}R_1 & -\overline{\Sigma}_{11}\overline{\Sigma}_{32}R_2 - \overline{\Sigma}_{12}\overline{\Sigma}_{31}R_2 & (k_{den} + \overline{\Sigma}_{11}\overline{\Sigma}_{22}R_3 - \overline{\Sigma}_{12}\overline{\Sigma}_{21}R_3)/K_{den} \end{bmatrix}$$

where,

$$\begin{aligned} K_{den} &= \overline{\Sigma}_{11}\overline{\Sigma}_{22}\overline{\Sigma}_{33} - \overline{\Sigma}_{11}\overline{\Sigma}_{23}\overline{\Sigma}_{32} - \overline{\Sigma}_{12}\overline{\Sigma}_{21}\overline{\Sigma}_{33} + \overline{\Sigma}_{12}\overline{\Sigma}_{23}\overline{\Sigma}_{31} + \overline{\Sigma}_{13}\overline{\Sigma}_{21}\overline{\Sigma}_{32} - \overline{\Sigma}_{13}\overline{\Sigma}_{22}\overline{\Sigma}_{31} \\ K_{11} &= (k_{den} + \overline{\Sigma}_{22}\overline{\Sigma}_{33}R_1 - \overline{\Sigma}_{23}\overline{\Sigma}_{32}R_1)/K_{den} \\ K_{12} &= (-\overline{\Sigma}_{12}\overline{\Sigma}_{33}R_2 - \overline{\Sigma}_{13}\overline{\Sigma}_{32}R_2)/K_{den} \\ K_{13} &= (\overline{\Sigma}_{12}\overline{\Sigma}_{23}R_3 - \overline{\Sigma}_{13}\overline{\Sigma}_{22}R_3)/K_{den} \\ K_{21} &= (-\overline{\Sigma}_{21}\overline{\Sigma}_{33}R_1 - \overline{\Sigma}_{23}\overline{\Sigma}_{31}R_1)/K_{den} \\ K_{22} &= (k_{den} + \overline{\Sigma}_{11}\overline{\Sigma}_{33}R_2 - \overline{\Sigma}_{13}\overline{\Sigma}_{31}R_2)/K_{den} \\ K_{23} &= (-R_3(\overline{\Sigma}_{11}\overline{\Sigma}_{23} - \overline{\Sigma}_{13}\overline{\Sigma}_{21}))/K_{den} \\ K_{31} &= (\overline{\Sigma}_{21}\overline{\Sigma}_{32}R_1 - \overline{\Sigma}_{22}\overline{\Sigma}_{31}R_1)/K_{den} \\ K_{32} &= (-\overline{\Sigma}_{11}\overline{\Sigma}_{32}R_2 - \overline{\Sigma}_{12}\overline{\Sigma}_{31}R_2)/K_{den} \\ K_{33} &= (k_{den} + \overline{\Sigma}_{11}\overline{\Sigma}_{22}R_3 - \overline{\Sigma}_{12}\overline{\Sigma}_{21}R_3)/K_{den} \end{aligned} \tag{A.5}$$

$$\begin{aligned} \mu_k &= \overline{\mu}_k + \mathbf{K}_k(z_k - \mathbf{C}_k\overline{\mu}_k) \\ &= \begin{bmatrix} \overline{x}_k - K_{13}(\overline{\theta}_k - z_{\theta_k}) - K_{11}(\overline{x}_k - z_{x_k}) - K_{12}(\overline{y}_k - z_{y_k}) \\ \overline{y}_k - K_{23}(\overline{\theta}_k - z_{\theta_k}) - K_{21}(\overline{x}_k - z_{x_k}) - K_{22}(\overline{y}_k - z_{y_k}) \\ \overline{\theta}_k - K_{33}(\overline{\theta}_k - z_{\theta_k}) - K_{31}(\overline{x}_k - z_{x_k}) - K_{32}(\overline{y}_k - z_{y_k}) \end{bmatrix} \end{aligned} \tag{A.6}$$

$$\Sigma_k = (\mathbf{I} - \mathbf{K}_k\mathbf{C}_k)\overline{\Sigma}_k = \begin{bmatrix} 1 - K_{11} & -K_{12} & -K_{13} \\ -K_{21} & 1 - K_{22} & -K_{23} \\ -K_{31} & K_{32} & 1 - K_{33} \end{bmatrix}_k \begin{bmatrix} \overline{\Sigma}_{11} & \overline{\Sigma}_{12} & \overline{\Sigma}_{13} \\ \overline{\Sigma}_{21} & \overline{\Sigma}_{22} & \overline{\Sigma}_{23} \\ \overline{\Sigma}_{31} & \overline{\Sigma}_{32} & \overline{\Sigma}_{33} \end{bmatrix}_k \tag{A.7}$$

$$= \begin{bmatrix} -\overline{\Sigma}_{11}(K_{11} - 1) - K_{12}\overline{\Sigma}_{21} - K_{13}\overline{\Sigma}_{31} & -\overline{\Sigma}_{12}(K_{11} - 1) - K_{12}\overline{\Sigma}_{22} - K_{13}\overline{\Sigma}_{32} & -\overline{\Sigma}_{13}(K_{11} - 1) - K_{12}\overline{\Sigma}_{23} - K_{13}\overline{\Sigma}_{33} \\ -\overline{\Sigma}_{21}(K_{22} - 1) - K_{21}\overline{\Sigma}_{11} - K_{23}\overline{\Sigma}_{31} & -\overline{\Sigma}_{22}(K_{22} - 1) - K_{21}\overline{\Sigma}_{12} - K_{23}\overline{\Sigma}_{32} & -\overline{\Sigma}_{23}(K_{22} - 1) - K_{21}\overline{\Sigma}_{13} - K_{23}\overline{\Sigma}_{33} \\ -\overline{\Sigma}_{31}(K_{33} - 1) - K_{31}\overline{\Sigma}_{11} - K_{32}\overline{\Sigma}_{21} & -\overline{\Sigma}_{32}(K_{33} - 1) - K_{31}\overline{\Sigma}_{12} - K_{32}\overline{\Sigma}_{22} & -\overline{\Sigma}_{33}(K_{33} - 1) - K_{31}\overline{\Sigma}_{13} - K_{32}\overline{\Sigma}_{23} \end{bmatrix}$$

Appendix B

Extended Kalman Filter algorithm

The main difference between the Kalman filter and the Extended Kalman filter algorithm is that the prediction uses the nonlinear version of the non-holonomic vehicle shown in Eq. (3.2) instead of the linearised version as in Eq. (A.1).

Therefore, in order to use the EKF we have to replace Eq. (A.1) in the prediction step by Eq. (B.1).

Prediction

$$\begin{aligned} \overline{\mu}_k &= g(\mu_{k-1}, u_k) \\ &= \begin{bmatrix} \overline{x} \\ \overline{y} \\ \overline{\theta} \end{bmatrix}_k = \begin{bmatrix} v_1 \cos(\theta_{k-1}) \\ v_1 \sin(\theta_{k-1}) \\ \frac{v_1}{L}(\tan(\phi_{k-1})) \end{bmatrix} T_s + \begin{bmatrix} \overline{x} \\ \overline{y} \\ \overline{\theta} \end{bmatrix}_{k-1} \end{aligned} \tag{B.1}$$

and proceed in the same way as in Appendix A because the observation matrix $\mathbf{C} = \mathbf{I}$.

Bibliography

[1] Assad Alam. Fuel-efficient distributed control for heavy duty vehicle platooning. 2011.

[2] Chloe Albanesius. Google car: Not the first self-driving vehicle. *PC Magazine*, October 2010.

[3] James M. Anderson, Nidhi Kalra, Karlyn D. Stanley, Paul Sorensen, Constantine Samaras, and Oluwatobi Oluwatola. Autonomous vehicle technology - a guide for policymakers. Technical report, RAND Corporation, 2014.

[4] K.H. Ang, G.C.Y. Chong, and Y. Li. Id control system analysis, design, and technology. *IEEE Trans Control Systems Tech*, 13(4):559–576, 2005.

[5] Burak Benligiray, Cihan Topal, and Cuneyt Akinlar. Video-based lane detection using a fast vanishing point estimation method. *2013 IEEE International Symposium on Multimedia*, 0:348–351, 2012.

[6] Ames Research Center. State estimation. Available at `http://www.nasa.gov/centers/ames/research/technology-onepagers/state-estimation.html`.

[7] R. Chi Ooi. Balancing a two-wheeled autonomous robot. Master's thesis, School of Mechanical Engineering, University of Western Australia, 2003.

[8] Digilentinc.com. *PmodGYRO™ Reference Manual*.

[9] Robot domestici. National instruments support. Available at `http://sine.ni.com/psp/app/doc/p/id/psp-1166/lang/sv`.

[10] Tamar Flash, Yaron Meirovitch, and Avi Barliya. Models of human movement: Trajectory planning and inverse kinematics studies. *Robotics and Autonomous Systems*, 61(4):330 – 339, 2013. Models and Technologies for Multi-modal Skill Training.

[11] Skolan för elektro-och systemteknik. Smart mobility lab. Available at `http://www.kth.se/ees/forskning/strategiska-forskningsomraden/intelligenta-transportsystem/smart-mobility-lab`.

[12] Katalin György, András Kelemen, and László Dávid. Unscented kalman filters and particle filter methods for nonlinear state estimation. *Procedia Technology*, 12(0):65 – 74, 2014. The 7th International Conference Interdisciplinarity in Engineering, INTER-ENG 2013, 10-11 October 2013, Petru Maior University of Tirgu Mures, Romania.

[13] Tower hobbies. Semi-truck and trailer rc kits. Available at `http://www3.towerhobbies.com/cgi-bin/WTI0001P?I=LXWXW4&P=8`.

[14] Logitech. Logitech f310 gamepad. Available at `http://gaming.logitech.com/en-us/product/f310-gamepad`.

[15] A. Luca, G. Oriolo, and C. Samson. Feedback control of a nonholonomic car-like robot. In J.-P. Laumond, editor, *Robot Motion Planning and Control*, volume 229 of *Lecture Notes in Control and Information Sciences*. Springer Berlin Heidelberg, 1998.

[16] V. Malyavej, P. Torteeka, S. Wongkharn, and T. Wiangtong. Pose estimation of unmanned ground vehicle based on dead-reckoning/gps sensor fusion by unscented kalman filter. 01:395–398, May 2009. Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2009. ECTI-CON 2009. 6th International Conference on.

[17] National Instruments. *Products - myRIO*.

[18] Scoop project 2011. Scoop - stockholm cooperative driving. Available at `http://scoop.md.kth.se/`.

[19] Robotshop.com. Gyro sensor pmodgyro l3g4200d. Available at `http://www.robotshop.com/media/files/images/gyro-sensor-pmodgyro-l3g4200d-3-axis-large.jpg`.

[20] Elsevier Science. *Data Fusion in Robotics & Machine Intelligence*. Elsevier Science, 1992.

[21] Sparkfun. Infrared proximity sensor - sharp gp2y0a21yk. Available at `https://www.sparkfun.com/products/242`.

[22] 2012 Project Course Students. Final report - automatic control, project course 2012. Skolan för elektro- och systemteknik.

[23] 2013 Project Course Students. Final report - automatic control, project course 2013. Skolan för elektro- och systemteknik.

[24] Tamiya. *Scania R620 6X4 Highline Full Operation Kit*. Tamiya, April 2009.

[25] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Intelligent robotics and autonomous agents. MIT Press, 2005.

[26] Trimble Navigation Limited. *Real-Time Kinematic surveying - Training guide*, revision d edition, September 2003.