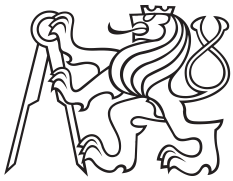
Bachelor Project



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Control Engineering

Autonomous Vehicle Localization

Martin Jeřábek

Supervisor: Doc. Ing. Tomáš Haniš PhD.
Field of study: Control Engineering
May 2024

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Jeřábek Martin** Personal ID number: **503236**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Autonomous vehicle localization

Bachelor’s thesis title in Czech:

Lokalizace autonomního vozidla

Guidelines:

1. Get familiar with planar dynamics of all-wheel-steered vehicles and implement design and verification mathematical model.
2. Get familiar with autonomous vehicle dynamics and localization sensors and sensor data fusion algorithms.
3. Prepare framework for real world data acquisition, including sensors integration and data logging and maneuver design and execution.
4. Develop and implement vehicle localization and dynamics data processing and fusion.
5. Test implemented algorithms using real-world data.

Bibliography / sources:

[1] Dieter Schramm, Manfred Hiller, Roberto Bardini – Vehicle Dynamics – Duisburg 2014
[2] Hans B. Pacejka - Tire and Vehicle Dynamics – The Netherlands 2012
[3] Robert Bosch GmbH - Bosch automotive handbook - Plochingen, Germany : Robet Bosch GmbH ; Cambridge, Mass. : Bentley Publishers
[4] Tomáš Twardzik, „Fúze Senzorických Dat Polohy Pro Autonomní Vozidla“, diplomová práce ČVUT FEL, 2022
[5] G. Park, "Interacting Multiple Model Kalman Filtering for Optimal Vehicle State Estimation," 2022 2nd International Conference on Robotics, Automation and Artificial Intelligence (RAAI), Singapore, Singapore, 2022, pp. 87-91, doi: 10.1109/RAAI56146.2022.10093004……..
[6] G. Park, "Vehicle Sideslip Angle Estimation Based on Interacting Multiple Model Kalman Filter Using Low-Cost Sensor Fusion," in IEEE Transactions on Vehicular Technology, vol. 71, no. 6, pp. 6088-6099, June 2022, doi: 10.1109/TVT.2022.3161460

Name and workplace of bachelor's thesis supervisor:

doc. Ing. Tomáš Haniš, Ph.D. Department of Control Engineering FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **29.01.2024** Deadline for bachelor thesis submission: **24.05.2024**

Assignment valid until:
by the end of summer semester 2024/2025

doc. Ing. Tomáš Haniš, Ph.D. Supervisor's signature	prof. Ing. Michael Šebek, DrSc. Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Acknowledgements

I want to express my gratitude to all the people who helped me along the way of writing thi thesis. To my supervisor doc. Ing. Tomáš Haniš, Ph.D. for his guidance and patience with my sometimes malfunctioning cognitive abilities, my colleagues from the Smart Driving Solution Research Center Ing. Jan Švancar, Ing. David Vošahlík for sharing their knowledge and having the nerves to listen to my angry mumbling during the countless hours spent debugging code in the office. Also to all my friends and family for their encouragement and support to the point, when all there was left for me to do was only beating my laziness and do the actual work. Special thanks to my girlfriend, who tolerated the long days and nights spent in front of the computer working instead of spending quality time together.

Declaration

I declare, that this thesis and the work presented are my own created following the ČVUT guidelines and I have cited all used information and resources properly per the ethical principles for preparing university theses.

In Prague, 10. May 2024

Abstract

The evolution of electronic driver assistance systems, from the second half of the 20th century, has significantly increased the demand for accurate vehicle state data. Nowadays the demand is critical with the advent of autonomous and overactuated vehicles, whose critical vehicle systems rely on accurate and reliable vehicle state estimation. This thesis addresses these needs by developing an in-house sensor fusion system for vehicle state estimation. The sensor fusion system will be developed on embedded vehicle platforms built by the SDS Research Center. The sensor fusion solution has two parts. The complementary filter merges data from GNSS, IMU, and wheel odometry to provide estimates of longitudinal and lateral velocities and vehicle heading. And the EKF, adapted from a Matlab-based offline original to a real-time C++ implementation modular to be integrable to multiple vehicle platforms and also into ROS2 environment. The EKF estimates all important vehicle dynamics states - slip angle, heading, yaw rate, longitudinal and lateral velocities, and body velocity. Experimental validation was conducted on the ToMi2 platform, a sub-scale vehicle designed for testing advanced driver assistance systems. Despite the physical and hardware limitations of the platforms, the implemented sensor fusion solutions showed reliable state estimation. The thesis demonstrated that the development of a real-time in-house sensor fusion system on embedded hardware can match the performance of professional equipment.

Keywords: sensor fusion, Matlab, Simulink, C++, GNSS, IMU, kinematic models, single track model, code generation, embedded platforms, overactuated vehicles, GPS outage, complementary filter, Extended Kalman filter, senzorická rekonfigurace

Supervisor: Doc. Ing. Tomáš Haniš PhD.
Budova E, Karlovo náměstí 13, Praha 2

Abstrakt

Vývoj elektronických asistenčních systémů v automobilech, počínaje druhou polovinou 20. století, významně zvedá poptávku po precizních měřeních dat o stavu vozidla. S nástupem autonomních aut a přeaktuovaných vozidel, jejichž kritické řídící szstémy na tato data spoléhají se spolehlivá dostupnost těchto dat stává zásadní. Tato práce, s ohlédnutím na výše popsaný problém, má za cíl vývoj systému pro senzorickou fůzi dat s použitím dostupné výpočetní techniky. Tento systém pro senzorickou fůzi dat bude vyvýjen pro platformy výzkumného střediska SDS. Tento systém má dvě hlavní části, komplementární filtr, používající data z GNSS,IMU a odometrie kol pro odhad směru vozidla a podélné a příčné rychlosti. Druhou částí je rozšířený kalmanův filtr, převzaný z původní verze napsané v Matlabu do podoby schopné fungovat v reálném čase a napsané v C++ tak, aby byla zajištěno snadné přemigrování szstému na jinou platformu a také jeho integrace do prostředí systému ROS2.Tento filtr přidává k už zmíněným zdrojům dat ještě vizuální odometrii a mezi jeho výstupz patří všechna důležitá data pro jízdní dynamiku - slip angle, heading, yaw rate, longitudinal and lateral velocities, and body velocity.
Validace experimenty byla provedena na platformě ToMi2 - zmenšeného model přeaktuovaného vozidla. Navzdory některým fyzickým a vybavenostním limitacím se povedlo prokázat, že navržené systémy fungují. Tato práce dokázala, že navrhnout systém pro senzorickou fůzi dat použitelný na vestavěných počítačových platformách a schopný funkce v reálném čase je možné s běžně dostupným vybavením a přístroji. Zároveň je možno s takovýmto systémem konkurovat i porfesionálním zařízením.

Klíčová slova: senzorická fůze, Matlab, Simulink, C++, GNSS, IMU, kinematický model, jednostopý model, přeaktuovaná vozidla, Extended Kalman filter, komplementrární filtr, vestavěné platformy, generování zdrojového kódu, výpadek GPS, senzorické rekonfigurace

Překlad názvu: Lokalizace autonomních vozidel

Contents

Figures

Tables

Chapter 1

Introduction

1.1 Motivation

With the invention of electronic driver assistants in the 1970s, such as the Anti Lock Braking System (ABS), a demand for collecting data about vehicle states appeared. The demand for reliable data collection grew as the driver assistants were evolving into more capable and more complex systems able to control the vehicle without the need for driver input, such as Adaptive Cruise Control, automatic parking assistants, etc. To bring some method to the madness and categorize these systems, the Society of Automotive Engineers (SAE) came up with the 5 levels of automated driving[20]. Starting at level 0 these systems can be summarized as passive indicators of the surrounding environment with visual or audio indication, such as ultrasonic parking sensors or the Blindspot Information System. The middle ground is level 3, where the driver and the car can simultaneously cooperate. When the right scenario appears, the car can automatically perform a set task, but if any danger or unexpected situation occurs, the driver is alerted and requested to take over. Level 4 means that the driver can be omitted from the driving tasks, a good example is Automatic Valet Parking, where the driver can already be out of the vehicle. Full autonomy belongs to level 5, at this stage, a person sitting in the car is no longer considered a driver, but more a passenger. The car can perform all driving tasks on its own, thus the interface to directly operate the car is no longer mandatory. A comprehensive description and visual layout of these levels can be seen on 1.1.
All the systems mentioned above need reliable information about the vehicle states to be able to function accordingly (to save on words written and keep the readability at a reasonable level, this information about vehicle states will be called data from now on). Data from, let‘s call them traditional, sensors in cars like wheel RPM, odometry, GNSS positioning, and cameras, is sufficient for lower-level autonomous systems. The closer to full autonomy it gets, the more sensors are utilized[21], such as LiDaRs, stereo cameras, etc. To enable proper functionality of the autonomous systems and advanced driver assistants the data collected has to match strict reliability and precision requirements, this is often done by utilization of sensor fusion[22].
Professional-grade measuring units use a variety of sensor fusion algorithms

to satisfy the requirements in question. Since these are really capable systems with large computation potential and all sorts of safety licenses, the prices can get really expensive. For example, the internal navigation system Vbox 3iS by VBox Automotive[23], can measure just about anything possible in the car and has an integrated Kalman filter for sensor fusion retails in the hundreds of thousands Kč range. This, in combination with the fact, that these units tend to be fairly large in size, poses two problems for the development of embedded testing vehicle platforms. Another disadvantage is that since these units are meant to offer robust solution in a variety of conditions, such an aim for robustness usually means losses in the accuracy department.
This thesis aims to solve these problems by developing an in-house sensor fusion system for estimating vehicle states on vehicle platforms created by my colleagues of the Smart Driving Solutions Team.

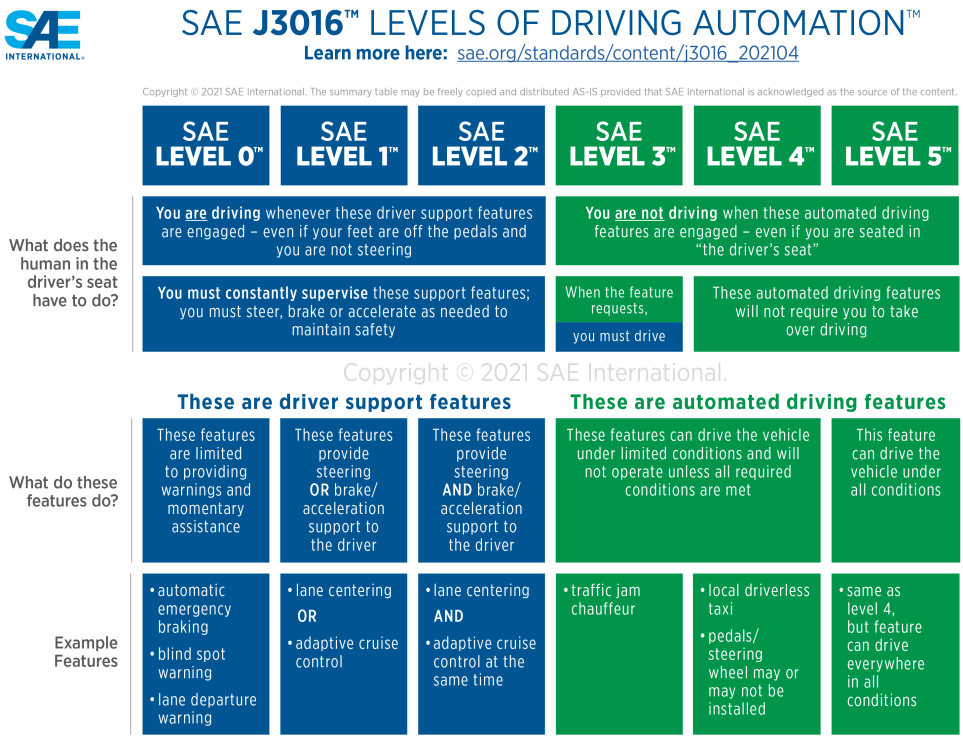


Figure 1.1: Levels of autonomous driving as established by the Society of Automotive Engineering[1].

1.2 Basic Introduction

Sensor fusion can be described as a process of combining data from multiple sources so that the result has less uncertainty and is more accurate and reliable than if the sources were used individually. Complementary and Extended Kalman filters will be used in this project. **The Complementary filter** is a

method to combine data from two different sources with independent errors to achieve more accurate results than if the data were used separately[24]. **The Kalman filter**, named after its inventor Rudolf Kalmam is an algorithm used for state estimation of linear dynamic systems. Extended Kalman filter adds the ability to predict nonlinear systems by linearizing the system in the corresponding operating point[25].

1.2.1 Vehicle State Estimation

To enable the functionality of all the complex and autonomous systems it is first needed to have a robust and reliable sense of the inner and outer vehicle states. Usually obtained through sensor measurements and consequent data fusion, to yield the best possible results. This process is called **vehicle state estimation** and in this thesis, the estimated states will be,

$$\mathbf{x} = \begin{bmatrix} \beta \\ \psi \\ \dot{\psi} \\ p_x \\ p_y \\ v \end{bmatrix} \tag{1.1}$$

where $\mathbf{x}$ is vector of states, β is slip angle of the car, ψ is heading, $\dot{\psi}$ is yaw rate, p_x and p_y are position coordinates and v is vehicle velocity.

1.3 State Of The Art

The advent of overactuated platforms and autonomous vehicles introduced new requirements for both vehicle dynamics control and the demand for various sensor measurements. Due to their more complex longitudinal and lateral dynamics, the overactuated platforms have to measure and control more states than the traditional vehicles. This fact means the need for more advanced state estimation (described in section 1.2.1 is growing. The following sections show, how are these problem currently approached in the industry and research.

1.3.1 Overactuated Vehicles

REE Automotive

The Israeli-based company is currently one of the major players in the development of EV overactuated utility vehicles. The pivotal point of their innovation is the **REEcorner**[26] - a comprehensive unit of all the critical vehicle components (steering, powertrain, brakes, suspension, and control), as shown in figure 1.2. Thanks to the integrated ECU, the wheel can be completely independent of the other wheel modules, enabling four-wheel independent steering to increase maneuverability. Another advantage of

having this neat package is the modularity of the chassis platform. The vehicle can be built with various footprints and sizes since all is packaged in the corner. This freed-up space translates to a flat cargo bed leading to better space usage than traditional trucks. The modular flat platform housing 4 REEcorner units can be seen in figure 1.3. The company strives to make its

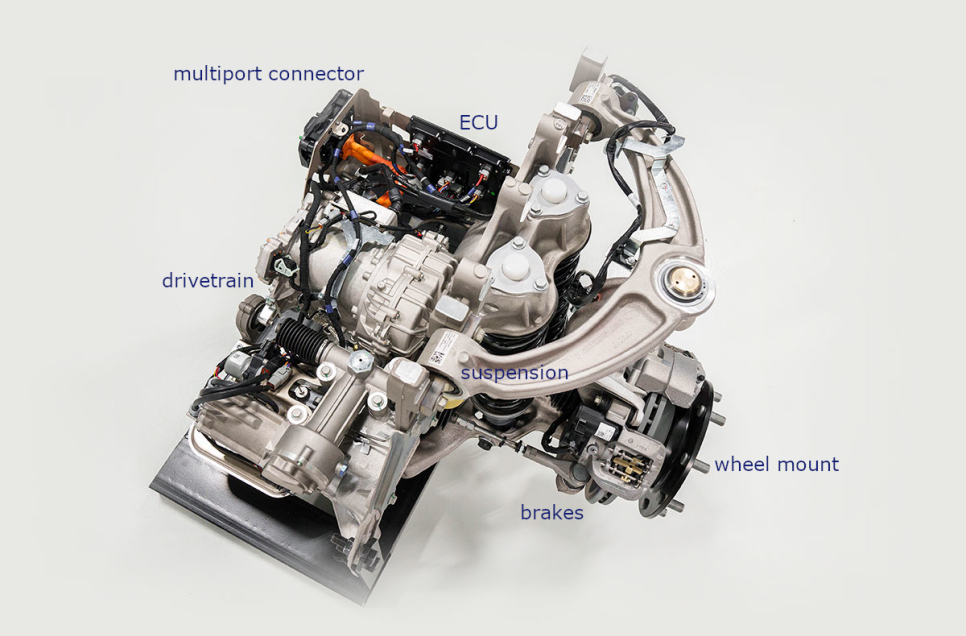


Figure 1.2: Construction of the REEcorner with all vital driving systems integrated in one unit[2].

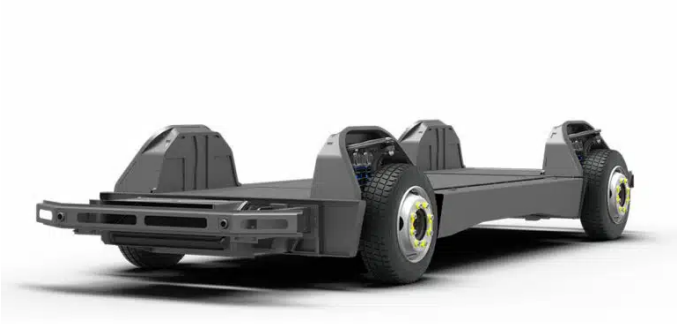


Figure 1.3: Modular REE automotive platform utilizing 4 REEcorner units[3].

first large shipments to customers in 2025.

Hyundai Mobis e-Corner

At CES 2024, Hyundai's sister company Mobis unveiled their concept of an overactuated vehicle built on the Hyundai Ionic 5 platform, called *Mobion*. Like the REEcorner (1.3.1), Mobis developed an integrated wheel module

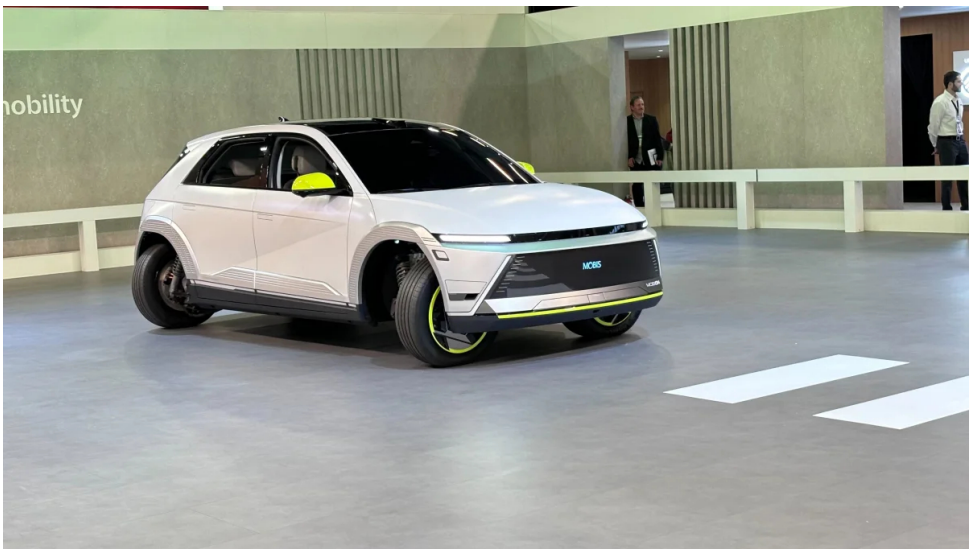


Figure 1.4: Hyundai Mobion equipped with the e-Corner module at CES 2024, showcasing the 90° steering angle[4].

housing all critical vehicle components utilizing by-wire technology called the e-Corner. The main difference from REEcorner is that the e-Corner has in-wheel motors and each wheel can turn up to 90°. This large steering angle unlocks the potential to execute interesting maneuvers, the ”piece de resistance” of the Mobion car[27].

Thanks to the e-Corner module, the car can shorten the turn radius to execute a tight turn, when the rear and front axles steer in the opposite directions. Axles steered in the same direction (also called the ”Crab walk”), can be used to execute turning at high speeds, helping with stability and reducing body roll. Another class of movements relates to parking. To effortlessly execute parallel parking into a tight spot, all four wheels can be turned 90° to slide into said spot. When the need to make a 180° turn in a constrained environment occurs, the motors can turn in an opposite direction to make a 180° pivot. This configuration also allows the possibility of a J-turn. The rear left wheel stays put, the rear right drives backward, and the front wheels are turned to the right and drive forward, performing a J-turn.

Hummer EV Truck

Hummer is a staple of the rugged SUV market, taking pride in its vehicles' robustness and toughness. In 2020, Hummer EV was unveiled, powered by up to 3 motors, ready to show the world, that electric vehicles can withstand the harsh conditions of off-road driving. It was one of the first cars to feature rear-wheel steering of up to 10°. This meant it could reduce its turn radius to perform tight turns when the rear axle counter-steered and when steered in the same direction, one of the main selling points of the car - the Crab Walk mode, enabled the possibility of avoiding obstacles by moving diagonally[28]. A photo of the crab walk mode can be seen in figure 1.5.

Figure 1.5: Hummer EV with wheels in the Crab Walk mode configuration[5].

1.3.2 Autonomous Systems

Tesla Autopilot

Tesla is generally known as a front-runner in the development of autonomous driving software with its Tesla Autopilot. It is classified as a level 2 autonomous system, meaning that driver oversight is always required and must be ready to take control at moment's notice[29]. The main feature is a self-driving mode, providing lane keeping, adaptive cruise control, and automatic lane changing interconnected with the navigation system to ensure the optimal route to given destination. Tesla claims that the Autopilot is capable to function without human intervention and oversight fully, but it cannot be made available due to the legislative obstacles the autonomous systems face. As opposed to other manufacturers Tesla uses only camera data for its system, omitting the use of ultrasonic sensors and LiDaRs[30]. The car is equipped with 8 cameras covering full 360° field of view and with 250 meters of maximum front-facing range. The data acquired is then processed by Tesla's in-house developed neural network which is said to provide all necessary data to enable the function of the Autopilot.

Mercedes Drive Pilot

The Mercedes Benz Drive Pilot boasts to be the first road-legal level 3 system on the market. Mercedes markets this system as conditionally automated driving. The conditions required include clear line markings on approved freeways, moderate to heavy traffic with speeds under 40 MPH (65 km/h), daytime lighting, and clear weather. When these conditions are met, the car is able to navigate and drive itself in the traffic without needing the driver's

attention. To provide data needed to execute these automated tasks the car is equipped with a battery of sensors. Opposite to Tesla, Mercedes uses various types of sensors to acquire its data. The cameras are accompanied by LiDaR, radars, ultrasonic sensors, and road moisture sensors to gather information about the car's state and surrounding environment. When the conditions are no longer met, the car issues a takeover request to the driver who has to respond in 10 seconds otherwise, the car performs an emergency stop in its current lane and calls for medical help. The Mercedes Drive Pilot is currently available only on selected roads in California and Nevada[31][32].

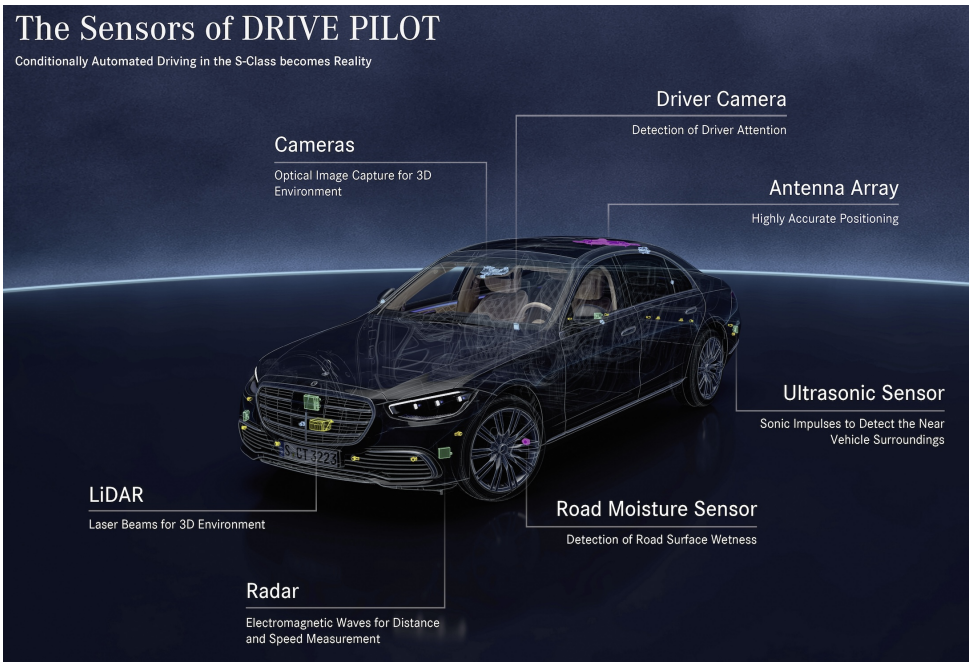


Figure 1.6: Mercedes Drive Pilot sensor instrumentation[6].

Waymo

The USA-based company prides itself in having the world's first Autonomous ride-hailing service. Its fleet of so-called autonomous taxis comprises of modified Jaguar I-pace vehicles equipped with Waymo's custom sensor units. The units house multiple LiDaRs, radars, and 29 cameras (1.8) to provide sufficient data for autonomous driving[8]. From the beginning of 2024, Waymo has offered its services in Los Angeles, Phoenix, and Austin, where it underwent extensive mapping and testing to get precise maps of the area and prove the system's functionality to the governing bodies[33]. To guarantee the safety of the passengers and the other vehicles and pedestrians on the road, Waymo has implemented many redundancies in the vehicle to ensure safety in the case of any system failure. Also, the car can use external speakers to communicate with other traffic members to further improve pedestrian safety.

Figure 1.7: Waymo-modified Jaguar I-pace serving as the world's first autonomous taxi[7].

1.3.3 Sensor Fusion

Several sensor fusion methods are used to obtain optimal results by combining data from multiple sensors to achieve better performance than is taken separately. Sensor fusion methods can be categorized by the way the measurements are combined[22].

Competetive - multiple sensors measuring the same data to increase robustness and reliability.

Complementary - different measurements combined to negate weaknesses of individual sensors - gyroscope and accelerometer data combined to get velocity measurement.

Cooperative - measurements from different sensors combined in a way that leads to output unattainable if used separately - images from individual lenses of stereo camera to get a 3D scene.

The two most used in the topic of vehicle state estimation (and also in this thesis) are a frequency-based method called the complementary filter and model-based method call the Extended Kalman Filter.

Complementary filter

As mentioned, the complementary filter is a frequency-based sensor fusion method. It combines sensor data with independent noise, best if one noise source is high and the other is low frequency-based. The corresponding signals are sent through low-pass and high-pass filters and then added together to form new measurements with reduced error[24].

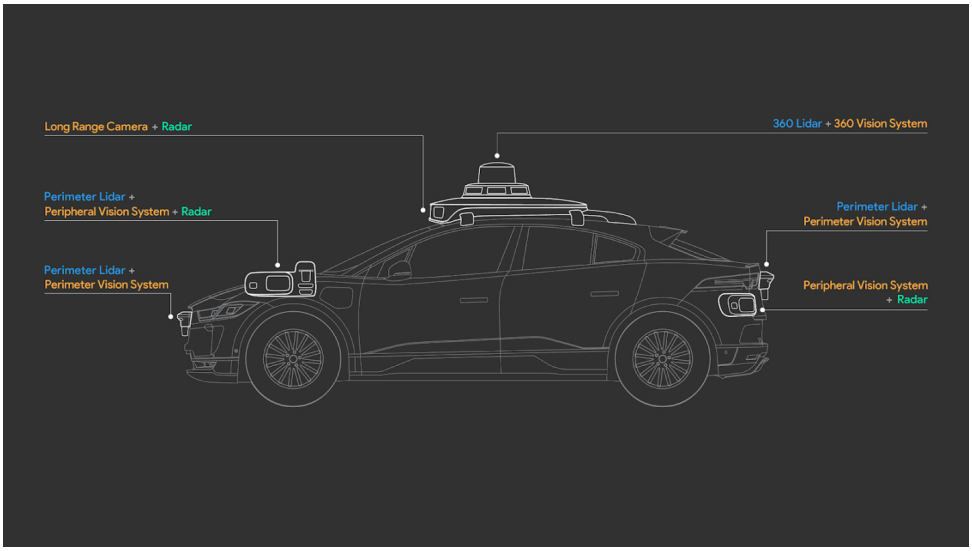


Figure 1.8: Description and localization of the sensors on the 5th gen Waymo Driver[8].

Extended Kalman filter

The Kalman filter, presumably the most frequent solution for state estimation is a model-based method measuring the system states and their covariances[25]. As can be seen in figure 1.9 the Kalman filter comprises of 2 main tasks.

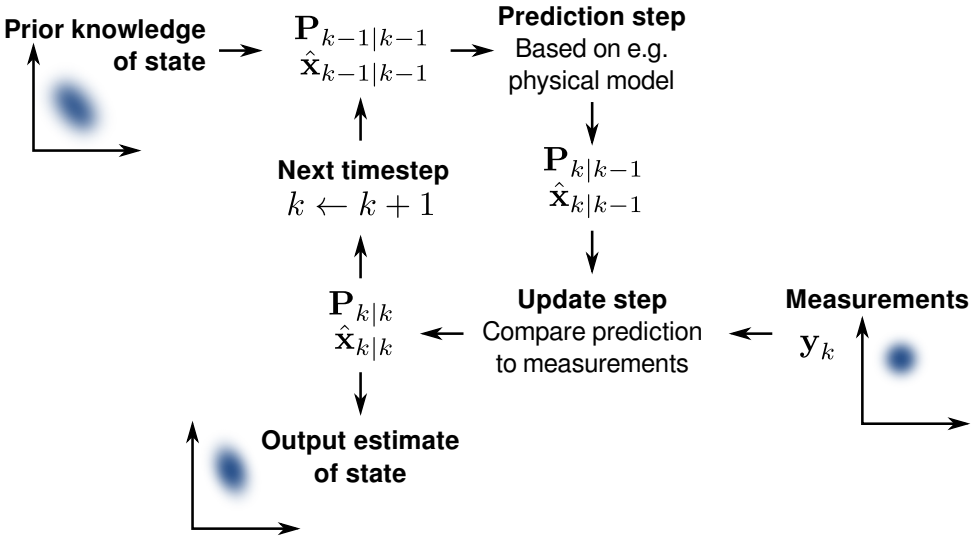


Figure 1.9: The basic visualization of phases in Kalman filter. $\mathbf{P}$ is the state covariance matrix and $\hat{\mathbf{x}}$ is vector of the predicted states[9].

Prediction step - calculation of the predicted estimate of the system states based given mathematical model of the system.

Update step - the predicted state is compared and adjusted by the measurement taken.

Unfortunately, even when the Kalman filter is the optimal linear state estimator, physical systems are more often than not nonlinear. To resolve this issue the Extended Kalman filter was developed. It takes over most of the Kalman filter's functionalities and solves the nonlinear system problem by adding linearization to the process. The system is linearized using the Taylor series in a suitable operating point and the rest of the computation is slightly tweaked to accommodate this change. This process will be explained more in-depth in section 4.2.

1.3.4 Inertial Navigation Systems

As mentioned in the previous sections, with the current and future rise of autonomous and overactuated vehicles, the importance of accurately measuring and estimating vehicle states becomes ever more significant. To acquire accurate and reliable data INS units are often used in basically all vehicle types - road, air, space, and nautical. The function of INS can be split into 3 main parts. The internal measurement unit (IMU - described in 2.2) measures linear acceleration, angular rates and object orientation. GNSS provides absolute positional data from its measurements. The data processing phase (often represented by some form of Kalman filter) is responsible for fusing the measurements. The main benefit of using INS is **dead reckoning**, which is a way of estimating an object's state when no absolute data is available - when GNSS signal is lost for longer periods, the dead reckoning can compute its location from the relative data provided by IMU. In combination with the Kalman filter, the INS can provide robust state estimation in difficult conditions[34]. The only downside is that professional INS units able to provide high-precision data can be expensive when considering the usage for smaller projects with limited budgets.

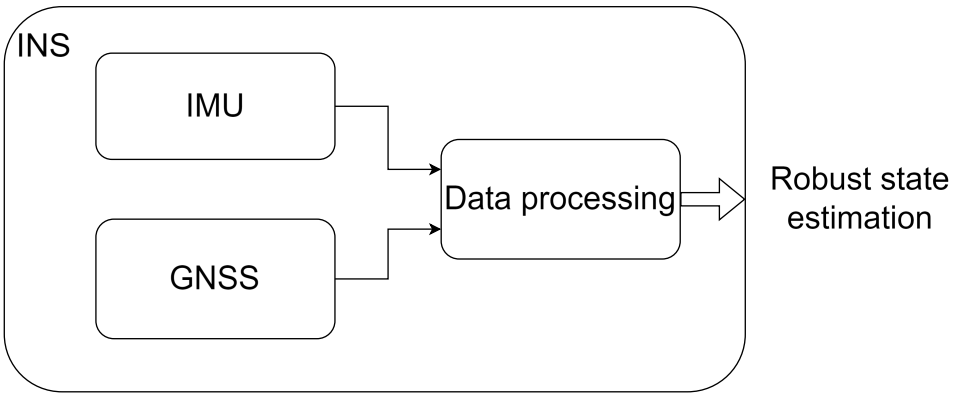


Figure 1.10: Diagram of major functional blocks in INS

The only downside is that professional INS units able to provide high-precision data can be expensive when considering the usage for smaller projects with limited budgets.

VBox 3iS

The VBox Automotive company specializes in the development of positioning systems for land vehicles. The VBox 3iS[23] is the range-topping INS unit. Apart from traditional INS which use IMU and GNSS data, the VBox can be connected to the wheel odometry data via the CAN interface. The integration of wheel odometry into the INS computation significantly reduces the drift of IMU measurements when no GNSS data is available. The manufacturer promises a 100 Hz GNSS receiver, and high-precision IMU, in the dual antenna variant, the VBox has RTK capabilities leading to centimeter precision positioning. The retail price for this unit is in the hundreds of thousands of Kč range.

Figure 1.11: The VBox 3iS[10]

1.4 My contribution

As already mentioned in the sections 1.1 and 1.3, the demand for collecting vehicle data is growing. Robust vehicle state estimation is required for all the advanced and autonomous control systems to function properly. Also, with the advent of various forms of overdrive vehicles, the importance of thoroughly controlling and understanding vehicle dynamics, and in particular side-slip angle and lateral speed, is high. In full-size automobiles, the sensor instrumentation possibilities are virtually endless. Multiple cameras, LiDaRs, sets of ultrasonic sensors, and more. On the other hand, when it comes to developing sub-scale platforms the limitations in the form of usable space and power start to appear. My contribution to the problem is the creation of a sensor fusion system for the development vehicle platform created by my colleagues from the Smart Driving Solutions Research Center. Given the size of the platform and the fact, that it is subject to multiple student

projects, the use of a professional-grade INS unit is not considered a viable option. This thesis aims to create a complementary filter as a baseline form of sensor fusion, and an Extended Kalman Filter using the "Golde Tschechische Hande" to combine attainable hardware and custom-made software to match the performance of the professional equipment. The EKF is based on the enviable work of Tomáš Twardzik[35], who has developed it as a part of his master's thesis. My adaptation of said EKF will be written in C++ and deployable in real-time on the car. The goal is to prove, that robust vehicle state estimation is possible on embedded platforms and that a bit of elbow grease and wit can outmatch full pockets of the green ones when it comes to performance.

Chapter 2

Theoretical Part

This chapter aims to shed light on the theoretical basics of sensors and concepts used in this thesis.

2.1 GNSS

GNSS stands for Global Navigation Satellite System and is the leading solution for position tracking in the world. The most famous GNSS system is GPS, a technology developed by the US Army in the 20. century, later other major world powers also recognized the need for their own GNSS tracking system for strategic reasons, which led to the creation of the Russian GLONASS system, Chinese BeiDou, and EU-made Galileo. India and Japan also have their technologies, but they are used only in their respective countries. GPS, GLONASS, and BeiDou are available globally.
The position data is successfully measured and calculated when the ground-level receiver obtains from the satellites orbiting in space. They carry precise atomic clocks, which are crucial for the final solution. These timestamps, in conjunction with other data sent by the satellites, are used to calculate the receiver's position.

2.1.1 Differential GNSS

To improve the precision of the standard GNSS solutions another receiver, called the base, can be added. The base has to be in a known position, usually obtained by averaging the measurement over several days. The difference between known and measured positions serves as a corrcection, which is sent to the receiver (in DGNSS term called the rover) to improve the accuracy of the final solution. Layout and short description of DGNSS can be seen in figure 2.1

2.1.2 Real-Time-Kinematics

In order to achieve centimetre-level accuracy in position measurements, a technology known as Real-Time Kinematics (RTK) has been developed. It is based on the DGNSS configuration of two receivers, rover and base, where

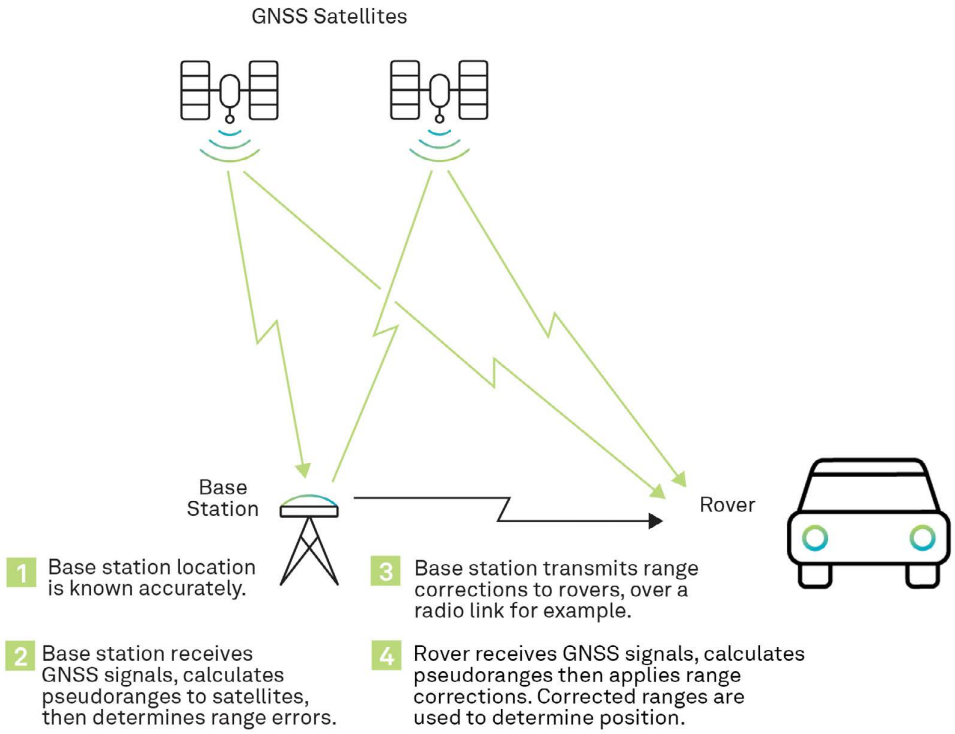


Figure 2.1: Layout and description of the DGNSS system[11].

standard corrections calculated from the difference between the known and received base positions are combined with corrections obtained by **carrier phase measurement**, which counts the number of wavelengths the signal sent by the satellite has, and **pseudo-range measurement**, which estimates the distance the signal had to travel to get from the satellite to the receiver. The RTK algorithm combines this data and outputs the new corrections, now called Real Time Correction Measurements (RTCM), which are then sent to the rover, where they are used to achieve the desired centimeter-level precision.

2.2 IMU

The inertial measurement unit measures orientation, linear and angular acceleration of an object. It houses accelerometers, gyroscopes, and in some cases also magnetometers.

2.2.1 Accelerometer

In most modern IMUs, MEMS (micro-electromechanical systems) accelerometers are used. As depicted in figure 2.2, the acceleration acting on the proof mass creates its displacement, which in turn changes the distance between the fixed plates and plates of the proof mass. This change in distance leads to a

change in electrical capacity between the plates which is converted into voltage change at the output and further in the resulting acceleration measurement.

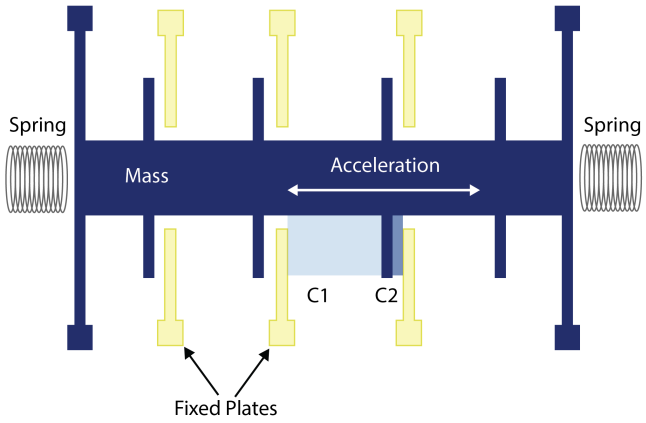


Figure 2.2: A schematic representation of MEMS accelerometer construction[12].

2.2.2 Gyroscope

The gyroscope measures the angular rate of an object. Like the accelerometers used in IMUs, most gyroscopes today are also MEMS. The gyroscope leverages the Coriolis effect to measure the angular rate. As can be seen in figure 2.3, the drive combs make the whole orange casing vibrate, hence when a torque is applied to the object, due to the mentioned Coriolis effect a change of capacity between the sense combs occurs, which is then converter to voltage change at the output and then into the resulting angular rate measurement[13].

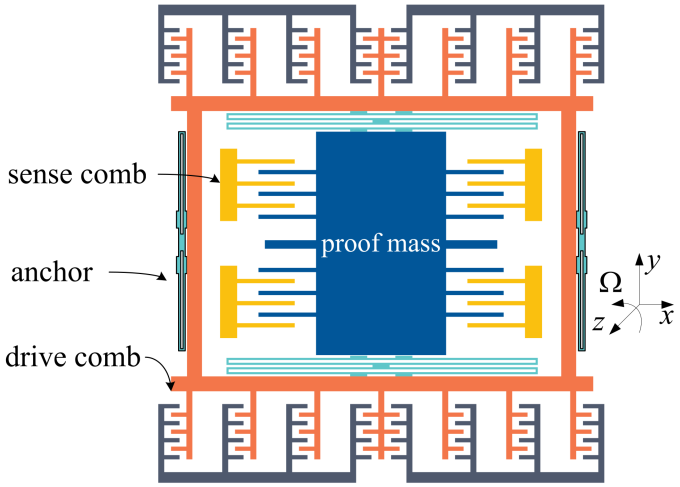


Figure 2.3: Structure of MEMS gyroscope[13].

2.2.3 Magnetometers

Magnetometers measure the object orientation in the earth's magnetic field. The ones mainly used in IMU work based on the Hall effect. As can be seen in figure 2.4, the magnetic field introduced to the conductor offsets the flow of electrons, thus a voltage can be measured across the conductor, which is then converted into magnetic field measurement. One hall sensor can measure the field only on one axis. To measure the magnetic field in all three directions standard magnetometer houses three hall sensors each measuring one axis[14].

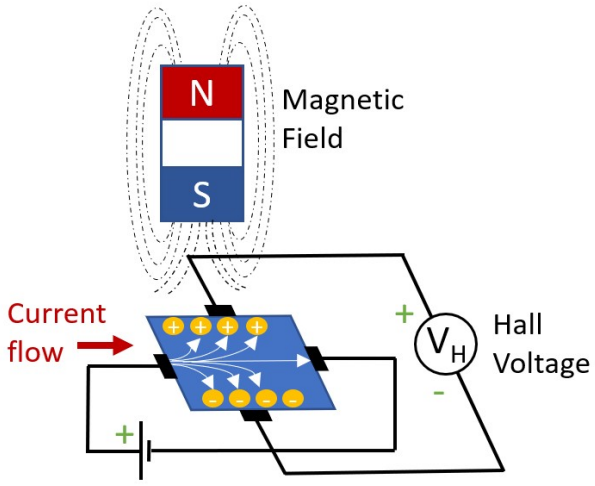


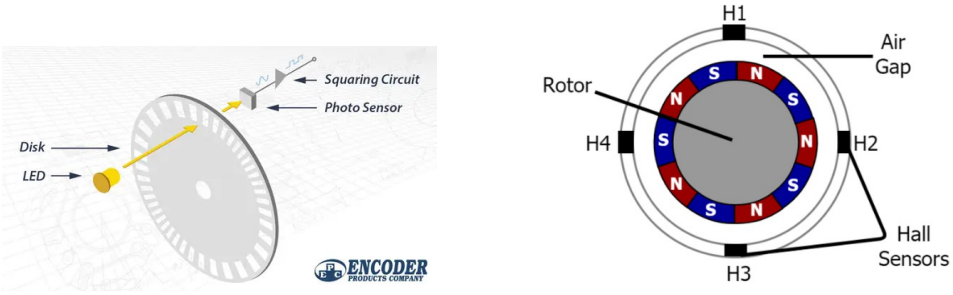
Figure 2.4: Visual representation of Hall effect magnetometer[14].

2.3 RPM Sensors

RPM can be measured in a number of ways the main types of the sensor are optical and Hall effect sensors. **Optical sensors** use the difference in the reflected light of contrasting parts of the encoder, as can be seen in figure 2.5a. **The Hall effect sensors** use a similar principle but instead of reflected light, they leverage the voltage change caused by the changing magnetic field of the magnet fixed to the rotor. This can be seen in figure 2.5b.

2.4 Camera

In general terms, a camera is a visual sensor capturing color images via photosensitive chips, which turn light into a signal, that is processed to form the final image. To preserve the color information to the digital world, the photosensitive pixels are aligned in what is called the Bayer filter and the construction can be seen in figure 2.6. Since the pixels can only detect light intensity and not color, the Bayer filter sets one of red, blue or green to each



(a) : Visual representation of the optical RPM sensor[36].

(b) : Visual representation of the Hall RPM sensor

Figure 2.5: Visual representation of different RPM sensors[15].

pixel on the grid. These positions are firmly set, and because of this, the final image looks natural to the human eye. In the present-day automotive industry, cameras are all the rage. They can be utilized in computer vision-related tasks like driving lanes, traffic signs and lights detection. More advanced cameras with multiple lenses, called stereo cameras, can be used to get a depth sense of the captured environment. These stereo cameras will be used later in this thesis to obtain visual odometry data with the help of certain computer vision algorithms[16].

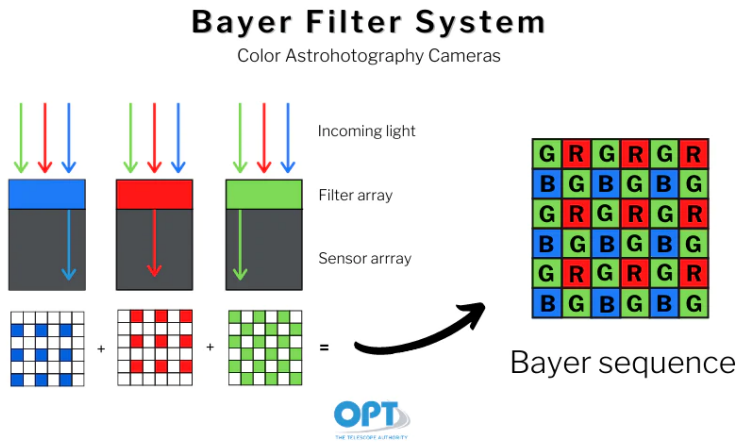


Figure 2.6: Pattern of the Bayer filter, each pixel on the grid has set RGB color, which is represented in the final processed image[16].

2.4.1 Visual Odometry

Visual odometry can be defined as a process of estimating vehicle states from given camera images. The raw image data do not offer much to work with and for that reason it has to be fed into a visual odometry algorithm. The algorithm typically consists of several phases. **Image processing** pahse, where tasks like adjusting for lens distortion are executed.**Feature detection**

phase, where objects of interest are tracked in the images. **Data processing** phase, where the extracted information is used to compute wanted results. Visual odometry can be aided by IMU measurements, often built into the camera body.

Chapter 3

Vehicle Platform

Since the developed sensor fusion algorithms are designed to be modular and used on multiple platforms created by the SDS Research Center, this chapter shows which vehicle platforms currently belong to the SDS Research Center, their architecture, attributes, and design. The modularity of the devised sensor fusion solutions is fueled by having the right vehicle model for each platform and quality sensor instrumentation on each platform to supply needed data to the models and algorithms.

3.1 ToMi2

The ToMi2 platform was developed[37] by my colleagues from the Smart Driving Solutions Team in collaboration with the Toyota Research Institute. It was created to provide a testing and verification platform for various research projects in the field of overactuated and autonomous vehicle control.

3.1.1 Physical Design

ToMi2 is built on top of the Losi Desert Buggy E-XL 2 RC platform, which in its unchanged form is a 1:5 RC model with a BLDC motor powering all four wheels. The main changes revolve around steering. To achieve independent steering on all wheels the front axle mechanism had to be altered and the rear axle completely changed for another front axle with the same modifications as the first one. In the final version, ToMi2 is only rear-wheel driven due to the noisiness of the revolution speed measurements on driven axes. Because of this, only the rear wheels are driven, so the data from the front axle can be used for reliable measurements. The independent steering of all four wheels is enabled by adding four servo motors, one to each wheel, which are responsible for steering. Other minor changes to the original car include a redesigned differential mount, repositioned batteries, and casing for all the microcontrollers that provide the control for the car.

Figure 3.1: The ToMi2 platform developed by the SDS Research Center in collabration with TRI.

3.1.2 ECUs Architecture

Raspberry Pi 4 is used as a head unit. It uses data from other microcontrollers that provide interfaces with corresponding sensors. To manage the communication with other mentioned controllers and data processing, ROS2 is used.
Navio2 board in combination with Raspberry Pi 3 is used to provide data from the GNSS module and IMU[38]. It also reads radio protocol SBUS data to send inputs for dynamic simulations and generates PWM inputs for steering servos and the traction motor.
The **STM432** measures the angular rates of the traction motor and all four wheels.
Arduino Micro reads accelerometer data from wheel axes and serves as the handler of radio communication with the handheld controller, which is used to manually control the car if desired.
Nvidia Xavier is in charge of processing camera images. It handles computer vision tasks, path planning and provides visual odometry data[39] .
All external boards communicate with the main Raspberry via UART, except the Nvidia Xavier, which uses Ethernet communication. A visual representation of this configuration can be seen in 3.2.

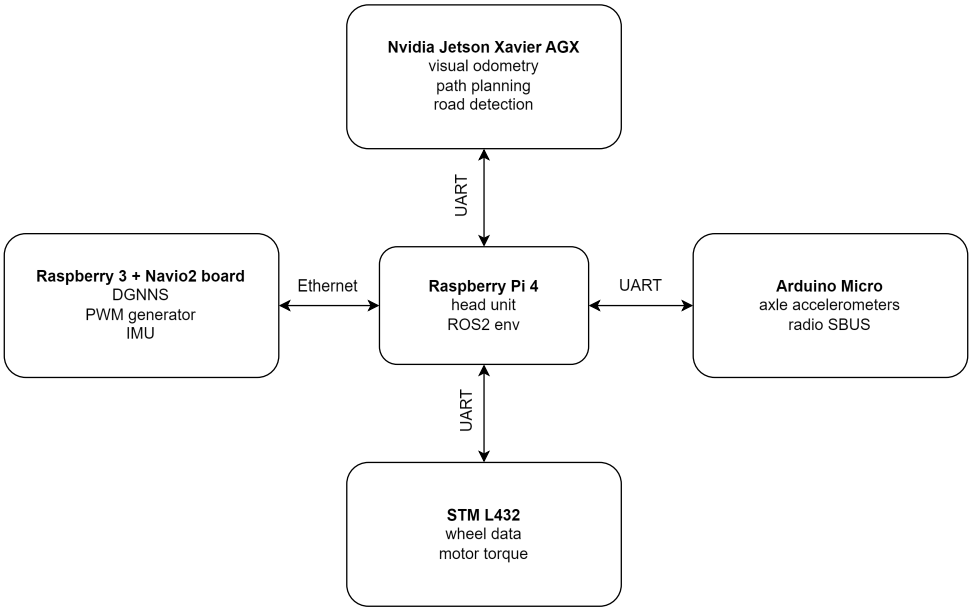


Figure 3.2: Diagram of ToMi2 system architecture

3.1.3 Sensors Used

DGNSS

As a part of the thesis of Tomáš Twardzik, a differential GNSS system added to the ToMi2 platform. The Stationary Base (SB) is placed in a known location and sends RTK corrections to the car carrying two receivers - Moving Base (MB) and Heading Module (HM), which enable precision body heading measurement instead of the movement heading measurable when only one receiver is on the car. All three GNSS modules used are U-Blox Zed F9P able to receive multiband GNSS signal, measuring rate of 20 Hz, and horizontal position accuracy of 1.5 or with RTK enabled up to 0.01 meters.

IMU

As described in section 3.1.2 the Navio2 board has built-in dual 9DOF IMU, which provides accelerometer, gyroscope, and magnetometer data. The location of the dual IMU on the Navio board can be seen in figure 3.3.

RPM Sensors

ToMi2 has RPM sensor for the main traction motor and for each wheel. These sensors are Hall effect RPM sensors (their function is described in section 2.3) and the were implemented by former SDS team member Tomáš Rutrle[37].

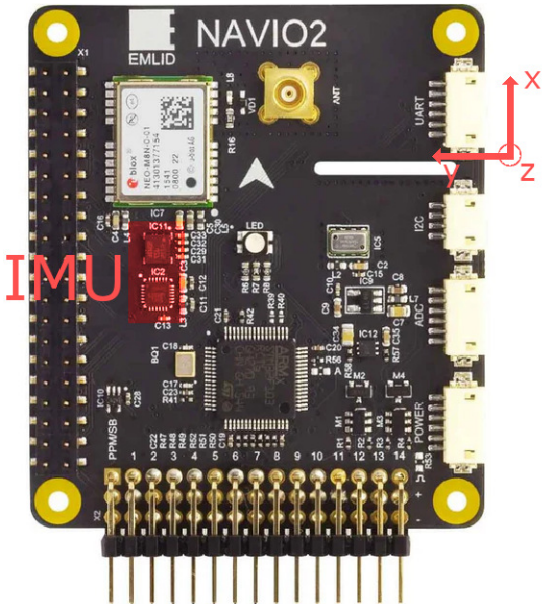


Figure 3.3: Navio board layout, the IMU placement highlighted in red.[11].

Figure 3.4: StereoLabs Zed2 camera with 2 sensors visible[17].

Camera

The StereoLabs Zed2 camera is also an addition by Tomáš Twardzik[35]. It was selected for its ability to provide data for deep learning, motion tracking, and depth sensing tasks. In this thesis, the camera in combination with the Nvidia Isaac Elbrus algorithm will be used to acquire data relevant for visual odometry. This algorithm was selected due to its high performance and the fact, that it is specially implemented for the Nvidia Xavier platform, which is used to process all the image data. The camera also has a built-in IMU to help increase measurement precision in combination with the Elbrus algorithm. The main yield of the Elbrus algorithm is linear and angular velocity estimation, which are important data for localization and vehicle state estimation.

3.1.4 Vehicle Model Parameters

This section lists the physical parameters of the ToMi2.

Table 3.1: Table of ToMi2 vehicle parameters

ToMi2 paramaters			
Name	**Parameter**	**Value**	**Units**
Cornering stiffness front	C_f	1232.4	N/rad
Cornering stiffness rear	C_r	1232.4	N/rad
Front distance to COG	l_f	0.3	m
Rear distance to COG	l_r	0.3	m
Wheel diameter	r	0.085	m
Weight	m	21	kg
Moment of inertia	I_z	245.5	$kg \cdot m^2$

3.2 Smart Sub-Scale Verification Platform

When the demand for research of new control algorithms started to outgrow the potential of ToMi2, an idea to create a new, bigger and better vehicle platform was born. The main upgrades sought after are all four wheels independently powered and steered, bigger space for housing various sensors that might be needed, and more computing power. This platform is still under construction, but all the work done in this thesis for ToMi2 is developed to be modular with easy transferability to this platform.

3.2.1 Physical Design

The vehicle is based on a 100x60 cm rectangular aluminum frame. The body frame is fitted with a motor at each corner. The motors are driven by SOLO motor controllers and can be controlled independently. The steering is provided by servo motors connected to each wheel making them independently steered with full 360° of rotation.

3.2.2 ECU Architecture

Two main components in the electronics layout are the **BeagleBone AI-64** responsible for the low-level controls and data logging from sensors, and the **Zotac E-series Zbox** thanks to its GPU handling camera tasks and computationally expensive tasks such as path planning and image processing.

BeagleBone AI-64

The BeagleBone AI-64 is an embedded Linux-based computer utilizing the 64-bit ARM architecture with a huge range of fast interfaces. The combination

Figure 3.5: Smart Sub-scale Verification Platform developed by SDS Research Center.

of its size and ability makes the BeagleBone an ideal building block of custom-made vehicle control architecture. In this case, the board has 3 peripheries connected communicating via 100 Hz UART and CAN[40].

Aceinna IMU - the Aceinna OpenRTK330 INS unit has both IMU and GNSS capabilities but this application has higher requirements for GNSS, so only its 100 Hz triple-redundant IMU is used[41].

Dual GPS - the dual GNSS receivers in moving base + heading module configuration used are the U-Blox ZED-F9P-00B-02, combined with static base station sending RTK correction to the car the system is capable of centimeter-level precision[42].

SOLO motor controllers - the SOLO UNO V2 controllers are used to drive the traction motors, capable of generating PWM output signal of frequency up to 80 kHz and CAN communication[43].

Zotac E-series Zbox

The Zotac is a mini-PC with Intel Core i7 CPU and NVIDIA GeForce RTX 4070 GPU[44]. These parameters mean, that on this platform Zotac is used for Image processing, Visual odometry computation, path planning, and all tasks requiring GPU in general. It is used to handle 2 peripheries, a stereo

camera and a Navio2 board, which is used only to receive signals from the remote controller used to control the motors.

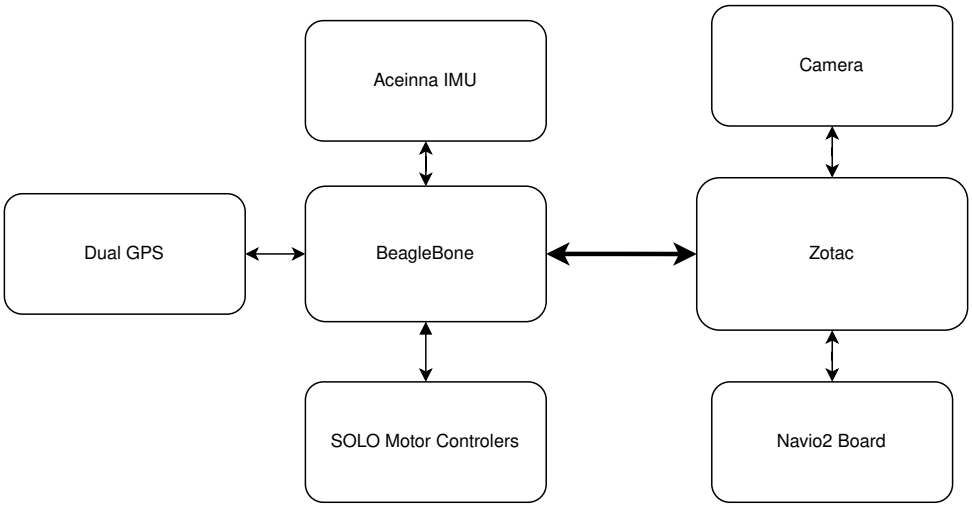


Figure 3.6: Block diagram of control architecture of the Smart Sub-scale Verification Platform.

Chapter 4

Implementation

4.1 Complementary Filter

The complementary filter is a sensor fusion method that combines different measurements to acquire more reliable data. It uses two filters complementing each other to improve sensor readings. When one source of measurement contains high-frequency noise, it gets sent through a low-pass filter to filter the noise out. The second source of measurements can contain drift as a form of low-frequency noise. That gets sent through a high-pass filter, that complements the low-pass filter, to filter out the drift. The filtered measurements get summed forming the output of a complementary filter, more accurate than the two inputs combined[24].

4.1.1 Implementation

The complementary filter in this thesis is used to filter three vehicle states, longitudinal and lateral velocity of the vehicle and its heading. To obtain data for the filter, pure sensor measurements are used along with the kinematic vehicle model to compute desired filter inputs. The derivation of the complementary filter is as follows:

$$1 = G(s) = H(s) + L(s), \tag{4.1}$$

$G(s)$ represents the complementary filter, $H(s)$ the high-pass filter, and $L(s)$ the low-pass filter. The equation 4.1 can be reformulated as,

$$1 = \frac{\tau s + 1}{\tau s + 1} = \frac{\tau s}{\tau s + 1} + \frac{1}{\tau s + 1}, \tag{4.2}$$

Meaning that the filters $H(s)$ and $L(s)$ are,

$$H(s) = \frac{\tau s}{\tau s + 1} \qquad L(s) = \frac{1}{\tau s + 1}. \tag{4.3}$$

Given the measurement $y(t)$, measurement rate $\dot{y}(t)$ and their respective laplace transformations $Y(s)$ and $sY(s)$ and an assumption that the filtered value $\hat{y}(t)$ equals to the sum of the measurement and the measurement rate,

$$\hat{Y}(s) = Y(s) + \frac{1}{s} sY(s), \tag{4.4}$$

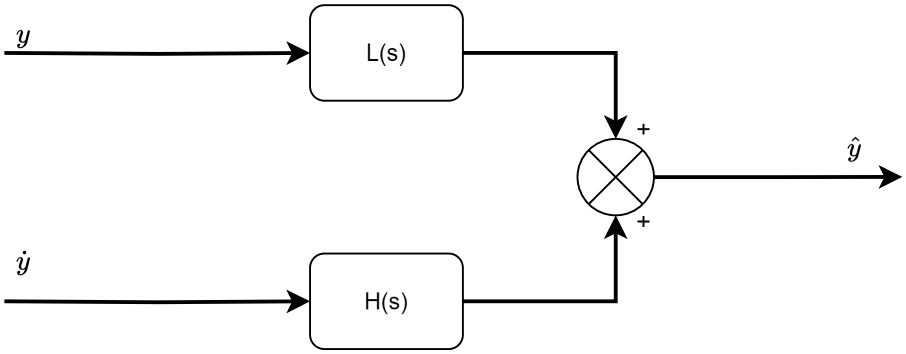


Figure 4.1: Block representation of a complementary filter. The input x is with high-frequency noise and the $L(s)$ is a low-pass filter. The input y contains drift and $H(s)$ is a high-pass filter.

The final equation for the complementary filter is,

$$\hat{Y}(s) = \frac{\tau s}{\tau s + 1}\frac{1}{s}sY(s) + \frac{1}{\tau s + 1}Y(s), \tag{4.5}$$

and can be shortened to,

$$\hat{Y}(s) = \frac{\tau}{\tau s + 1}sY(s) + \frac{1}{\tau s + 1}Y(s), \tag{4.6}$$

where the $sY(s)$ corresponds to $\dot{y}(t)$ and $Y(s)$ to the $y(t)$. Visual representation can be seen in figure 4.1. For real implementation in the vehicle, the filter has to be converted to the discrete domain using the zero-order hold method.

4.1.2 Kinematic Model

The kinematic model is a way to mathematically describe the motion of a vehicle without considering the forces that affect the motion. Several assumptions have to be made for the model to work[45]. They are:

Bicycle model - the vehicle is represented by two wheels, front and rear. The respective wheels are considered to be in the center of the original wheel axis.

Front and rear steering - the model is derived assuming both wheels are steered. The front steering angle is represented by δ_f and the rear angle by δ_r.

Center of gravity - the weight of the vehicle is assumed to be at one point. The distance from the COG to wheel axes is l_f for the front axle and l_r for the rear and the wheelbase corresponds to $L = l_f + l_r$.

Planar motion - the assumption of planar motion means that the Z coordinate and pitch and roll angles are neglected.

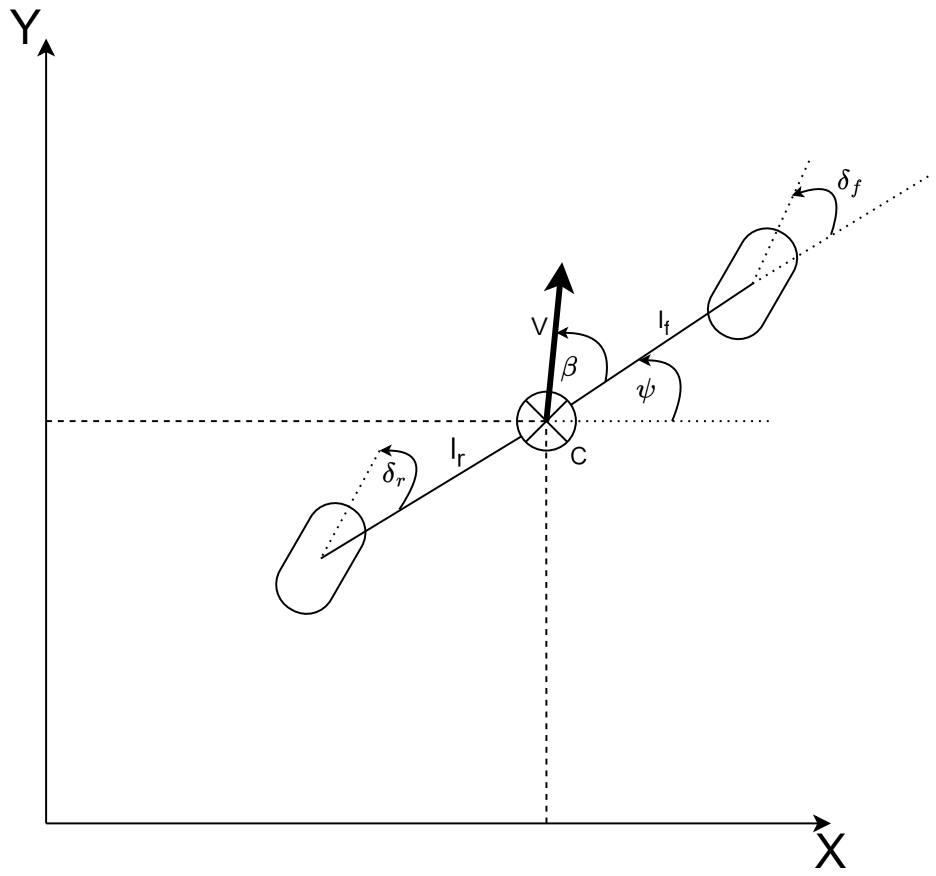


Figure 4.2: Image of representation of the kinematic model

Three coordinates are needed to describe the motion of the vehicle.

$$\begin{bmatrix} X \\ Y \\ \psi \end{bmatrix}, \tag{4.7}$$

$[X, Y]$ are the inertial coordinates of the location and ψ is heading describing the orientation of the vehicle. The velocity at the COG is denoted by V and makes and angle β with the vehicle's longitudinal axis. This angle β is called the slip angle of the vehicle. The kinematic model can be described by four equations.

$$\dot{X} = V\cos(\psi + \beta), \tag{4.8}$$

$$\dot{Y} = V\sin(\psi + \beta), \tag{4.9}$$

$$\dot{\psi} = \frac{V\cos(\beta)}{L}(\tan(\delta_f) - \tan(\delta_r)), \tag{4.10}$$

$$\beta = \arctan(\frac{l_f \tan(\delta_r) + l_r \tan(\delta_f)}{L}). \tag{4.11}$$

4.1.3 Longitudinal Velocity

Two data sources are used to obtain accurate longitudinal velocity measurements. First is an acceleration in the x-axis measured by the accelerometer in IMU, that must be compensated for the centrifugal force effect affecting MEMS accelerometers.

$$a_x^c = a_x - \dot{\psi} v_y, \tag{4.12}$$

a_x^c is the compensated accelerometer measurement, a_x is the raw accelerometer measurement and the v_y is lateral velocity computed from the kinematic model. The second source of data is v_x computed from the kinematic model, concretely from this equation.

$$v_x = V\cos(\beta) \tag{4.13}$$

Which is an adapted version of the equation 4.8 the slip angle β is used to compute v_x respective to the vehicle. The β is computed by 4.11. Since the integration introduced accumulated drift to the measurement, a high-pass filter is applied to the v_{xacc}. The v_x contains high-frequency noise from the steering angle measurements used to compute β and a low-pass filter is applied.

Implementation

As described in the section 4.1.1 the filer is first designed in a continuous domain and then discretized for better implementation in the car. Thus in the equation,

$$\hat{V}_x(s) = \frac{\tau s}{\tau s + 1} A_x^c(s) + \frac{1}{\tau s + 1} V_x(s), \tag{4.14}$$

the $Y_r(s)$ corresponds to the measurement rate - the accelerometer measurement and $Y(s)$ represents the velocity computed from the kinematic model. During tuning of the filter, $\tau = 0.05$ was found to get the most accurate results. Using the zero-order hold method, the filter is discretized to,

$$\hat{V}_x(z) = \frac{z - 1}{z - 0.8187} A_x^c(z) + \frac{0.1813}{z - 0.8187} V_x(z). \tag{4.15}$$

The Simulink implementation can be seen in figure 4.3. To deploy the lateral velocity complementary filter on the ToMi2 platform, C++ code was generated from the Simulink model and integrated into the vehicle control systems.

4.1.4 Lateral Velocity

Similar to the longitudinal velocity filter, for the lateral velocity filter, accelerometer measurement in the y-axis is compensated for the effect of the centrifugal force and then integrated over time,

$$a_y^c = a_y - \dot{\psi} v_x, \tag{4.16}$$

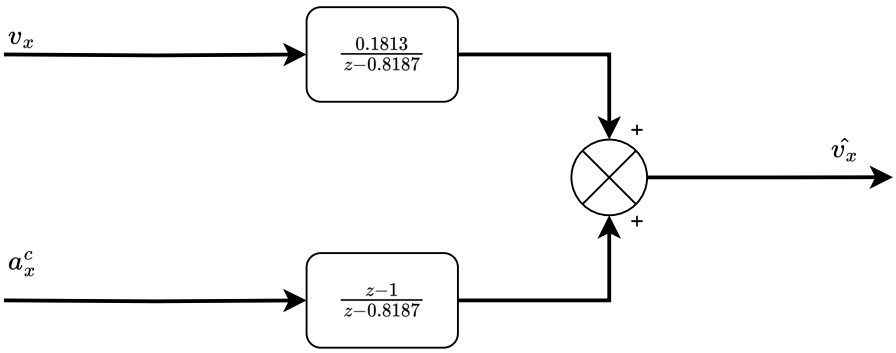


Figure 4.3: Simulink implementation for the discrete complementary filter for v_x

where a_y^c is the compensated accelerometer measurement, a_y is the pure accelerometer measurement and v_x is the longitudinal velocity from the kinematic model. The lateral velocity from the kinematic model v_y computation is based on 4.9, but to acquire only the velocity respective to the car, the equation is:

$$v_y = V\sin(\beta) \tag{4.17}$$

where V is the velocity of the car and β is vehicle slip angle. As the longitudinal case, the v_{yacc} contains drift due to the integration, so a high-pass filter is applied. The v_{ykin} is subject to noise from the steering angles used in β computations requiring the use of a low-pass filter.

Implementation

Similar to section 4.1.3, the filter is designed first in Laplace domain and then discretized. The Laplace representation is,

$$\hat{V_y}(s) = \frac{\tau s}{\tau s + 1} A_y^c(s) + \frac{1}{\tau s + 1} V_y(s), \tag{4.18}$$

where $\hat{V_y}(s)$ is the lateral velocity estimate in the Laplace domain, $A_y^c(s)$ is the compensated accelerometer measurement in the Laplace domain, and $V_y(s)$ is the lateral velocity from the kinematic model in the Laplace domain. By the use of cyclic constant optimization, the best-performing value if $\tau = 0.2$ was found. The discretized filter equation is,

$$\hat{V_y}(z) = \frac{z-1}{z-0.9512} A_y^c(z) + \frac{0.04877}{z-0.9512} V_y(z), \tag{4.19}$$

The Simulink implementation can be seen in figure 4.4. To deploy the lateral velocity complementary filter on the ToMi2 platform, C++ code was generated from the Simulink model and integrated into the vehicle control systems.

4.1.5 Heading

The heading complementary filter takes the body heading from the differential GNSS as the measurement with high-frequency noise and the yaw rate from

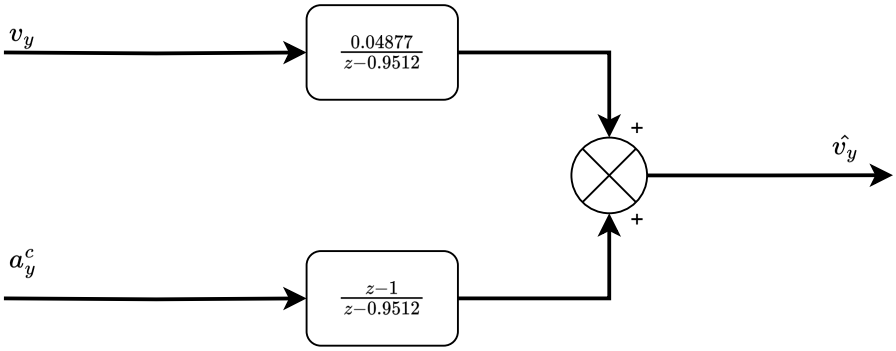


Figure 4.4: Simulink implementation for the discrete complementary filter for v_y

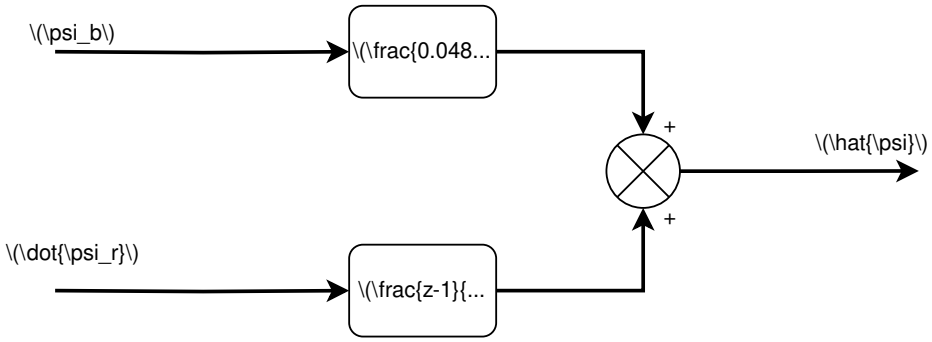


Figure 4.5: Simulink implementation for the discrete complementary filter for ψ

IMU measurements as the measurement with low-frequency noise. The body heading gets passed through a low-pass filter and the yaw rate is passed through a high-pass filter.

Implementation

Like the two previous complementary filters, this filter is also first designed in the Laplace domain and the discretized. The Laplace representation is,

$$\hat{\Psi}(s) = \frac{\tau s}{\tau s + 1}\Psi_r(s) + \frac{1}{\tau s + 1}\Psi_b(s), \tag{4.20}$$

where $\hat{\Psi}(s)$ corresponds to the filtered heading, $\Psi_r(s)$ to the yaw rate, and the $\Psi_b(s)$ to the body heading. The filter is tuned for the best performance, τ was set by cyclic optimization as $\tau = 0.1$. Using the zero-order hold method the Laplace domain filter is discretized to,

$$\hat{\Psi}(z) = \frac{z-1}{z-0.9512}\Psi_r(z) + \frac{0.04877}{z-0.9512}\Psi_b(z). \tag{4.21}$$

The Simulink implementation can be seen in figure 4.5. To deploy the heading complementary filter on the ToMi2 platform, C++ code was generated from the Simulink model and integrated into the vehicle control systems.

4.2 Extended Kalman Filter

This section encompasses the Kalman and Extended Kalman filter definition, implementation description, and verification.

4.2.1 Kalman Filter

As already briefly described in section 1.2 the Kalman filter is a state estimator for linear systems, utilizing model dynamics to enhance the precision of its outputs. Kalman represents its estimates in states and covariances. It consists of two main parts the state prediction and the state update[25].

State Prediction

Also called the time step uses the system dynamics to compute state predictions and its covariances for the next iteration. To the prediction phase belong 2 equations. The **State Propagation equation**,

$$\hat{\mathbf{x}}_{n+1,n} = \mathbf{F}\hat{\mathbf{x}}_{n,n} + \mathbf{G}\mathbf{u}_n, \tag{4.22}$$

where $\hat{\mathbf{x}}_{n+1,n}$ is the predicted state estimate, $\mathbf{F}$ is the state transition matrix, describing the system dynamics, $\hat{\mathbf{x}}_{n,n}$ is the current estimate, $\mathbf{G}$ is the control matrix mapping inputs to the system, and $\mathbf{u}_n$ is the current input vector. As mentioned above the Kalman filter works with state estimates and their covariances, so the second equation of the prediction phase is the **Covariance Propagation equation**,

$$\mathbf{P}_{n+1,n} = \mathbf{F}\mathbf{P}_{n,n}\mathbf{F}^T + \mathbf{Q}, \tag{4.23}$$

where $\mathbf{P}_{n+1,n}$ is the predicted estimate covariance, $\mathbf{P}_{n,n}$ is the current estimate covariance, and $\mathbf{Q}$ is the process noise covariance. After the state prediction phase has finished, the predicted estimates and their covariances are sent to the state update phase, where they no longer have the $n+1, n$ stamp but since they are already predicted the sample signature is n, n for the outputs of the prediction phase.

State Update

The state update phase takes the predicted values and physical measurements of the tracked states and combines them in a correction for the Kalman filter, yielding more accurate and precise estimates. This phase consists of the **Kalman gain equation**,

$$\mathbf{K}_n = \mathbf{P}_{n,n-1}\mathbf{H}^T(\mathbf{H}\mathbf{P}_{n,n-1}\mathbf{H}^T + \mathbf{R}_n)^{-1}, \tag{4.24}$$

where $\mathbf{K}_n$ is the Kalman gain, $\mathbf{R_n}$ is the measurement covariance, and $\mathbf{H}$ is the observation matrix. The Kalman gain indicates how trustworthy the measurements are and how much the prediction from the prediction phase

should be corrected. To compute the updated state estimate, the **State Update equation** is used,

$$\hat{\mathbf{x}}_{n,n} = \hat{\mathbf{x}}_{n,n-1} + \mathbf{K}_n(\mathbf{z}_n - \mathbf{H}\hat{\mathbf{x}}_{n,n-1}), \tag{4.25}$$

where $\hat{\mathbf{x}}_{n,n}$ is the updated state estimate, and $\mathbf{z}_n$ is the physical measurement. The output of the state update equation is considered the most precise and accurate current state estimate. The third equation of the update phase is the **Convariance update equation**,

$$\mathbf{P}_{n,n} = (\mathbf{I} - \mathbf{K}_n\mathbf{H})\mathbf{P}_{n,n-1}(\mathbf{I} - \mathbf{K}_n\mathbf{H})^T + \mathbf{K}_n\mathbf{R}_n\mathbf{K}_n^T, \tag{4.26}$$

where $\mathbf{P}_{n,n}$ is the current covariance estimate, $\mathbf{I}$ is identity matrix, and $\mathbf{P}_{n,n-1}$ is the last covariance estimate. The equation computes the correction of the predicted covariance matrix.
The state update phase provides reliable state estimates, which can be further used in subsequent applications. The Kalman filter is accepted as the optimal state estimator, but only for liner systems. Since most of the real-world systems in need of state estimation are nonlinear a variation of the Kalman filter was devised - the Extended Kalman filter.

4.2.2 Extended Kalman filter

The Extended Kalman filter was developed by NASA in the '60s. It builds on the capabilities of the Kalman filter but with the ability for state estimation of nonlinear systems. The system equations in the EKF do not have to be linear but have to be differentiable. Instead of the linear state transition and observation matrices, the EKF works with jacobians of the respective differentiable functions. In other words, the EKF does linearization in the current state and covariance estimates as the operating points and then proceeds to function as standard Kalman filter[46][46].

Prediction Phase

In the same way as the KF the EKF can be split into two phases, prediction and update. The prediction phase contains the **State Propagation equation**, where the linear system transition matrix $\mathbf{F}$ is replaced by,

$$\mathbf{F} \Rightarrow f(\mathbf{x}_n), \tag{4.27}$$

where $f(\mathbf{x}_n)$ represents the nonlinear equations of the system, so the EKF version of the State Propagation equation is,

$$\hat{\mathbf{x}}_{n+1,n} = f(\mathbf{x}_{n,n}) + \mathbf{G}\mathbf{u}_n, \tag{4.28}$$

where $\hat{\mathbf{x}}_{n+1,n}$ is the new state prediction, $\mathbf{G}$ is the control matrix and $\mathbf{u}_n$ is the vector of external system inputs. The **Covariance Propagation equation** has also undergone some changes the system transition matrix is now the jacobian of the nonlinear transiton matrix,

$$\mathbf{P}_{n+1,n} = \mathbf{F}\mathbf{P}_{n,n}\left(\mathbf{F}\right)^T + \mathbf{Q}, \tag{4.29}$$

where $\mathbf{P}_{n+1,n}$ is the new state covariance prediction, $\frac{\partial f}{\partial \mathbf{x}}$ is the jacobian of the state transition matrix, $\mathbf{P}_{n,n}$ is the current covariance estimate and $\mathbf{Q}$ is the process noise covariance matrix.

Update Phase

Similar to the prediction phase, the update phase of the EKF also introduces changes to the equations known from the KF. In the **Kalman Gain equation** is the observation matrix replaced by the jacobian of the nonlinear observation matrix,

$$\mathbf{K}_n = \mathbf{P}_{n,n-1}\left(\frac{\partial h}{\partial \mathbf{x}}\right)^T \left(\frac{\partial h}{\partial \mathbf{x}}\mathbf{P}_{n,n-1}\left(\frac{\partial h}{\partial \mathbf{x}}\right)^T + \mathbf{R}_n\right)^{-1}, \tag{4.30}$$

where $\mathbf{K}_n$ is the Kalman gain, $\frac{\partial h}{\partial \mathbf{x}}$ is the observation matrix jacobian, $\mathbf{P}_{n,n-1}$ is the covariance estimate, and $\mathbf{R_n}$ is the measurement covariance. The **State Update equation** for the EKF is,

$$\hat{\mathbf{x}}_{n,n} = \hat{\mathbf{x}}_{n,n-1} + \mathbf{K}_n(\mathbf{z}_n - h(\hat{\mathbf{x}}_{n,n-1})), \tag{4.31}$$

where $\hat{\mathbf{x}}_{n,n-1}$ is the updated state estimate, $\mathbf{z}_n$ is the measurement, and $h(\hat{\mathbf{x}}_{n,n-1})$ represents the nonlinear observation matrix. The **Covariance Update equation** of the EKF is,

$$\mathbf{P}_{n,n} = \left(\mathbf{I} - \mathbf{K}_n\frac{\partial h}{\partial \mathbf{x}}\right)\mathbf{P}_{n,n-1}\left(\mathbf{I} - \mathbf{K}_n\frac{\partial h}{\partial \mathbf{x}}\right)^T + \mathbf{K}_n\mathbf{R}_n\mathbf{K}_n^T, \tag{4.32}$$

where $\mathbf{P}_{n,n}$ is the updated covariance estimate, and $\mathbf{I}$ is the identity matrix.

4.2.3 Single-Track Model

The nonlinear dynamic model used in the EKF is the single-track vehicle model. This section provides insights into the function of said model. The implementation of the by Ing. Denis Efremov[18] is used. The single-track model has 3 degrees of freedom, representing the planar and yaw motion of the vehicle. In order for the model to work a number of assumptions have to be taken into account.

- Lift, roll, and pitch motion are neglected.
- Vehicle mass is centered at the center of gravity.
- The front and rear tires are assumed in the center of their respective axes.
- Mass distribution on the axles is assumed to be constant.
- Vehicle dynamics is controlled by the angular velocities of the wheels and steering angles.

- Aligning torque and the pneumatic trail resulting from the side-slip angle of a tire are neglected.

The model can be split into a number of functional blocks each representing different part of vehicle dynamics.

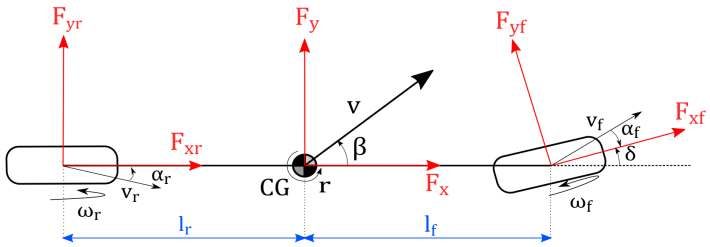


Figure 4.6: Single track model representation[18].

Rigid Body Dynamics

The rigid body dynamics block represents the behavior of the chassis. It has three degrees of freedom represented by the velocity of the COG ($v = |\mathbf{v}|$), side-slip angle β, and the rotational motion represented by yaw rate $\dot{\psi}$. The equations are as follows,

$$\dot{\beta} = \frac{1}{mv}(\cos(\beta)F_y - \sin(\beta)F_x) - \dot{\psi},$$
$$\dot{v} = \frac{1}{m}(\cos(\beta)F_x + \sin(\beta)F_y),$$
$$\ddot{\psi} = \frac{1}{I_z}M_z,$$

where $[\beta, v, \psi]$ are the model states, m is the weight of the car, F_x is the longitudinal force on the COG of the car, F_y is the lateral force on the COG of the car, I_z is the yaw moment of inertia and the M_z is rotational momentum.

Steering Angles Projection

The steering angles projection transforms the forces acting on tires in the wheel coordinate system to the forces and rotational momentum action on the COG. The transformation matrix can be written as,

$$\begin{pmatrix} F_x \\ F_y \\ M_z \end{pmatrix} = \begin{pmatrix} \cos\delta_f & -\sin\delta_f & \cos\delta_r & -\sin\delta_r \\ \sin\delta_f & \cos\delta_f & \sin\delta_r & \cos\delta_r \\ l_f\sin\delta_f & l_f\cos\delta_f & -l_r\sin\delta_r & -l_r\cos\delta_r \end{pmatrix} \begin{pmatrix} F_{xf} \\ F_{yf} \\ F_{xr} \\ F_{yr} \end{pmatrix}, \quad (4.33)$$

where δ_f is the steering angle of the front wheels and δ_r is the steering angle of the rear wheel.

Tire Models

The tire model has two stages, a raw calculation of the lateral and longitudinal forces on the wheels via the Simplified Pacejka Magic Formula[47] and a scaling stage called the traction ellipse. The Simplified Pacejka equations:

$$F_{xf}^{raw}(\lambda_f) = c_{D,x}F_{zf}\sin(c_{C,x}\tan^{-1}(c_{B,x}\lambda_f - c_{E,x}(c_{B,x}\lambda_f - \tan^{-1}(c_{B,x}\lambda_f)))),$$
$$F_{xr}^{raw}(\lambda_r) = c_{D,x}F_{zr}\sin(c_{C,x}\tan^{-1}(c_{B,x}\lambda_r - c_{E,x}(c_{B,x}\lambda_r - \tan^{-1}(c_{B,x}\lambda_r)))),$$
$$F_{yf}^{raw}(\alpha_f) = c_{D,y}F_{zf}\sin(c_{C,y}\tan^{-1}(c_{B,y}\alpha_f - c_{E,y}(c_{B,y}\alpha_f - \tan^{-1}(c_{B,y}\alpha_f)))),$$
$$F_{yr}^{raw}(\alpha_r) = c_{D,y}F_{zr}\sin(c_{C,y}\tan^{-1}(c_{B,y}\alpha_r - c_{E,y}(c_{B,y}\alpha_r - \tan^{-1}(c_{B,y}\alpha_r)))),$$

where λ_f, λ_r are front and rear longitudinal slip ratios, α_f, α_r are front and rear lateral slip ratios. The c_B, c_C, c_D, c_E are Pacejka magic constants, and the F_{zf}, F_{zr} are the load forces applied on the front and rear wheels. The load forces on the car are calculated as follows,

$$F_{zf} = gm\frac{l_r}{L}, \qquad F_{zr} = gm\frac{l_f}{L}, \tag{4.34}$$

where F_{zf} is the load force on the front wheel, g is the gravity coefficient of the Earth, F_{zr} is the load force on the rear wheel, l_f is the distance from the front wheel to the COG, l_r is the distance of the rear wheel to the COG, and $L = l_f + l_r$. The raw forces have to be scaled with respect to the tire. For that purpose, a traction ellipse model is used. The sum of the lateral and longitudinal forces on the tire cannot be greater than the load force acting on the wheel. This restriction is guaranteed by the use of the traction ellipse equaling,

$$F_tot = \sqrt{\frac{F_x^2}{c_{D,x}^2} + \frac{F_y^2}{c_{D,y}^2}} \leq \mu F_z \tag{4.35}$$

where μ is the friction coefficient of the road. The scaling of the forces to comply with the limitations of the traction ellipse is performed as follows,

$$\beta^* = \arccos\left(\frac{|\lambda|}{\sqrt{\lambda^2 + \sin^2(\alpha)}}\right) \tag{4.36}$$

$$\mu_{x,act} = \frac{F_x^{raw}}{F_z}, \qquad \mu_{y,act} = \frac{F_y^{raw}}{F_z}, \tag{4.37}$$

$$\mu_{x,max} = c_{D,x}, \qquad \mu_{y,max} = c_{D,y}, \tag{4.38}$$

$$\mu_x = \frac{1}{\sqrt{(\frac{1}{\mu_{x,act}})^2 + (\frac{\tan\beta^*}{\mu_{y,max}})^2}}, \qquad F_x = |\frac{\mu_x}{\mu_{x,act}}|F_x^{raw}, \tag{4.39}$$

$$\mu_y = \frac{\tan\beta^*}{\sqrt{(\frac{1}{\mu_{x,max}})^2 + (\frac{\tan\beta^*}{\mu_{y,act}})^2}}, \qquad F_y = |\frac{\mu_y}{\mu_{y,act}}|F_y^{raw}, \tag{4.40}$$

where F_x, F_y are maximum forces applicable in the longitudinal and lateral direction, and the μ are traction coefficients.

Wheel Kinematics

Wheel kinematics equations calculate the lateral and longitudinal velocity of each wheel (v_x, v_y), and the side-slip angle of the respective tires. The equations are,

$$\begin{pmatrix} v_{xf} \\ v_{yf} \end{pmatrix} = \begin{pmatrix} \cos\delta_f & \sin\delta_f \\ -\sin\delta_f & \cos\delta_f \end{pmatrix} \begin{pmatrix} v\cos\beta \\ v\sin\beta + l_f\dot{\psi} \end{pmatrix}, \tag{4.41}$$

$$\begin{pmatrix} v_{xr} \\ v_{yr} \end{pmatrix} = \begin{pmatrix} \cos\delta_r & \sin\delta_r \\ -\sin\delta_r & \cos\delta_r \end{pmatrix} \begin{pmatrix} v\cos\beta \\ v\sin\beta + l_f\dot{\psi} \end{pmatrix}. \tag{4.42}$$

The side-slip angles (α_f, α_r) can be calculated as,

$$\alpha_f = -\arctan\frac{v_{xf}}{|v_{xr}|}, \tag{4.43}$$

$$\alpha_r = -\arctan\frac{v_{xr}}{|v_{xr}|}. \tag{4.44}$$

Slip Ratios

The slip ratios of the wheels can be calculated using,

$$\lambda_f = \frac{\omega_f r - v_{xf}}{max(|\omega_f r|, |v_{xf}|)}, \qquad \lambda_r = \frac{\omega_r r - v_{xr}}{max(|\omega_r r|, |v_{xr}|)}, \tag{4.45}$$

where ω is the angular velocity of the wheel and r is the wheel diameter.

4.2.4 Implementation

As mentioned, this thesis builds on top of the already near-perfect EKF implementation by Tomáš Twardzik. The original is written in Matlab as an offline tool for post-processing logged data from CSV files. The program takes data from two main sources - GNSS, IMU, odometry measurements, and visual odometry data from images captured by a stereo camera. This thesis adds and changes several things about the original implementation:

- Online usability - the EKF outputs should be disposable in real-time to provide data for control algorithms and others.
- Modularity for deployment on other platforms - obtain a C++ library with the EKF functionality for easy integration to multiple platforms.

The changes were first made in Matlab, then Matlab Coder utility was used to generate the demanded C++ library. These two main objectives can be further divided into smaller subtasks and the goal of this section is to shed light on the modification made to the original program to meet the goals of this thesis.

Structure

To ease the cognitive strain needed to understand the changes made, the EKF can be divided into sections, on which can be better shown what modifications were made and why.

Inputs - handling input data to the filter.

Data sampling - parse the input data into set messages for the EKF.

Initial data gathering - get data to initialize the EKF.

Time step - update the estimates and covariances based on the system dynamics.

Data step - correct the predictions with the real-world measurements.

Inputs

Original - the original implementation takes two CSV files as inputs. Data logged from GNSS, IMU, odometry, and various miscellaneous information about the vehicle. The second CSV file contains visual odometry data processed by the Nvidia ISAAC-Elbrus algorithm.
Modified - the modified version adopts all the data types from the original, even though not all of it gets used, but is left in the program for possible future utilization. The data is only in form of one dimensional arrays, since the modified version works in real-time and only one data sample at the time is needed.

Data Sampling

Original - the offline version has a data sampling function, that parses all the data from the CSV files and sorts them into four message types - GNSS, IMU, VO (visual odometry), CTRL (wheel odometry). These messages get stored in a Matlab data object, and all have corresponding time stamps based on which they get sent to the EKF. This data sampling method utilizes Matlab's ability to handle variable-sized objects to store messages for a given time in a structure.
Modified - keeping in mind that the modified version has to check all the boxes for Matlab C++ code generation, the use of variable-sized objects is unacceptable. Luckily the real-time nature of the modification means that only one time sample at once has to be processed. The data is still sorted into the mentioned message types to ensure compatibility with the rest of the code, but the implementation has to be "dumbed down" to meet the C++ requirements. That means fixed-size variables, preallocated arrays and etc.

EKF initialization

Before the EKF itself can start, it needs data about its initial position. The data include *lattitude, longitude, attitude, body heading, accelerometer*

measurements, and yaw rate from gyroscope.
Original - since the offline version gets the data from a CSV file containing records of the whole session, the initial data is just the mean values of the first 1000 samples.
Modified - the initial data gathering is limited by time, not the number of samples. The sequence is set to last 30 seconds and the mean of the gathered samples is then used as the initial data needed for the EKF initialization.

Time Step

When data measurements is received, the EKF performs a prediction step, the theory is described in section 4.2.2.
Original - during the prediction step implementation the inputs are parsed to fit the model. The system matrices are computed and the Matlab *lsim* function is used to compute the system prediction. **Modified** - since the **lsim** function is not compatible with Matlab Coder C++ generation, a fourth-order Runge-Kutta solver was implemented instead.

Data Step

The update step is performed when sensor measurement data is received to compute the state and covariance estimate update. First, the observation matrix, measurement covariance matrix and their linearizations are computed. With these matrices the rest of the EKF update phase, as described in 4.2.2 is performed.
Original - the original version utilizes Matlab's ability to handle variable-sized objects to compose the measurement matrices.
Modified - since the C++ generation must use fixed-size variables, the beauty of the original function had to be unfortunately destroyed to be replaced by rigid if statements that cover all possible combinations of incoming data messages.

Chapter 5

Experiments

This chapter aims to show the promised functionality of the modified EKF and the Complementary filter. Given that the Sub-Scale Verification platform is still under construction, all tests and experiments will be conducted on the ToMi2 platform. Due to the size limitation of the car, carrying additional verification instrumentation directly on the vehicle, like the Vbox 3iS, is out of the question. The direct comparison between the Vbox and the developed algorithms would be ideal, but the second best thing and comparable method in terms of validating this thesis's outcomes is to set a validation track as a grand truth. Both the EKF and the complementary filter run simultaneously, to get plausible data for further validation and cross-comparison.

5.1 Validation track

As mentioned in 3.1.3, the ToMi2 platform is equipped with RTK-capable GNSS receivers, which can be leveraged to obtain precise measurements required to set the validation track. The individual points were obtained by averaging stationary position measurements along the desired track. Having individual points is only half of the work. To obtain a real validation track, Centripetal Catmull-ROM spline is used[48]. The spline generates a trajectory between the measured points as can be seen in figure 5.1. Since the validation track has only three states, which can be checked - $[x, y, \psi]$, where ψ is calculated by,

$$\psi_t = \arctan \frac{y_t - y_{t-1}}{x_t - x_{t-1}}, \tag{5.1}$$

in other words, *atan* of the difference between the current and the last position. Thanks to the attributes of the single-track model (4.2.3) these three states are enough to guarantee the accuracy of the other three states - $[\beta, \dot{\psi}, v]$, estimated by the EKF.

5.2 Complementrary filter

The implemented complementary filters handle lateral and longitudinal velocity and heading. The validation is conducted on the same data as the EKF

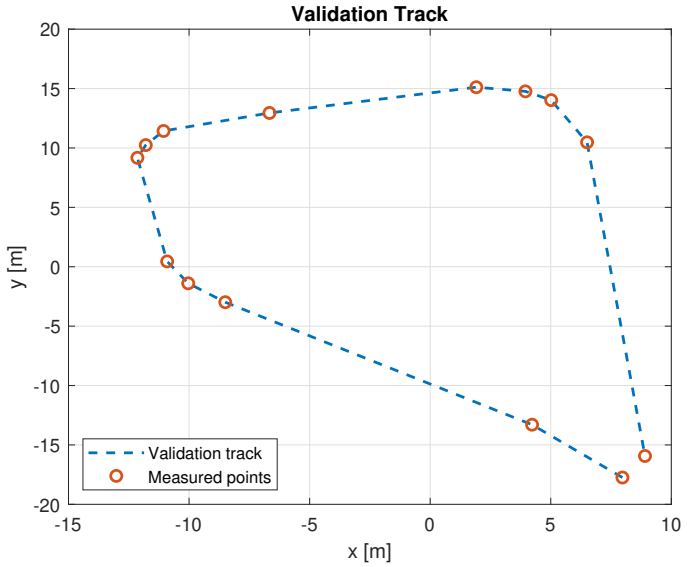


Figure 5.1: Validation track created from measured points using Catmull-ROM spline.

validation - a test ride around the validation track with speed change.

5.2.1 Longitudinal Velocity

The implementation of the filter is described in 4.1.3. The filter output is compared to the longitudinal velocity of the car's COG obtained from the kinematic model (also described in section 4.1.3). As can be seen in figure 5.3 the filtered measurement still has quite a bit of high-frequency noise and has an offset. Since this version of the complementary filter is tuned to minimize the difference, these remaining imperfections in the filtered signal can be tracked to the limitations of the vehicle platform, the sensor measurements, and the computed data from the kinematic model, which assumes non-existent slip at the wheels and works best at low-speed maneuvers - conditions hard to meet in real-world testing.

5.2.2 Lateral Velocity

The filter implementation is described in 4.1.4. The output is compared to the lateral velocity of the car's COG obtained from the kinematic model (also described in section 4.1.4). At first glance, the figure 5.4 could look like the filter in not properly tuned. Unfortunately, this output is the best performance, that could be achieved, due to similar reasons as mentioned in 5.2.1. In addition to that, the lateral dynamics of the overactuated vehicle are considerably faster than the longitudinal, so the limitations of the kinematic model show even more.

Figure 5.2: The validation track points displayed on satellite images for context[19].

5.2.3 Heading

The implementation is described in 4.1.5. Filter output is compared to the EKF heading, because the vehicle platform does not measure another independent source of body heading other than used as one of the inputs. As can be seen in figure 5.5 the headings are almost identical, apart from some offset. The heading complementary filter is the only filter shown, that can compare its performance to the EKF output of the respective state.

5.3 Extended Kalman filter

In section 4.2.2 it is stated that the Extended Kalman filter is prepared to fuse GNSS, IMU, odometry, and visual odometry data. Before any experiments are shown, it needs to be stated that at the time of the deadline of this thesis, I have not managed to successfully launch the Nvidia Isaac Elbrus[49] algorithm in combination with the Zed 2 STK camera. With the help of my smarter and more experienced colleagues, we only managed to locate the possible source of troubles as a non-compatible current Isaac algorithm version with our Zed camera. However, this is only a minor complication, as the EKF is more than capable even without the visual odometry data, and its modular architecture means, that when the Isaac algorithm is up and running, nothing has to change. The EKF is ready to process the additional data.

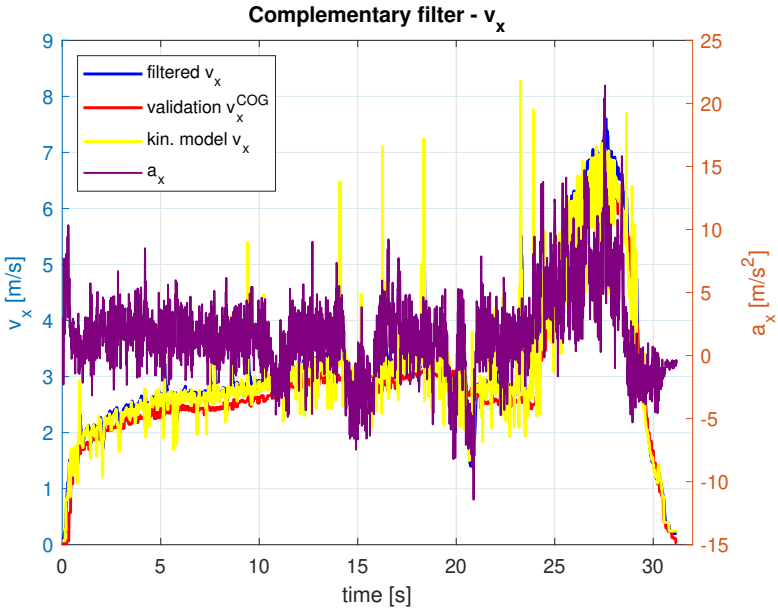


Figure 5.3: Filtered longitudinal velocity v_x compared to the longitudinal velocity of the car's COG v_x^{COG}.

5.3.1 Methodology

The experiments can be split into three categories,

General experiments - under ideal conditions to prove the baseline functionality

Offset experiments - these tests are meant to prove that the EKF is capable of withstanding the effects of offset in the GNSS measurements - in the real world experienced, when the receiver is surrounded by high obstacles and the signal has rebounded off the obstacles.

Sensor error - this category tests that the EKF is able to stay on course when a signal outage for a short period of time occurs.

Because of the lack of any trajectory-tracking technology, the author is left to rely on his vehicle-controlling skills via the wireless transmitter, meaning that the final accuracy of the validation trajectory tracking reflects human imperfections. Anyhow, this fact has no impact on the quality of the results, because, with the EKF, there are more ways how to show its abilities.

5.3.2 General

In term of the general functionality of the EKF, it should be first demonstrated, that the modified real-time C++ implementation has the same outputs as the original Matlab version. Figures 5.6 and 5.7 show that the conversion of the offline Matlab version to the real-time C++ version is valid - the logged data is the same.

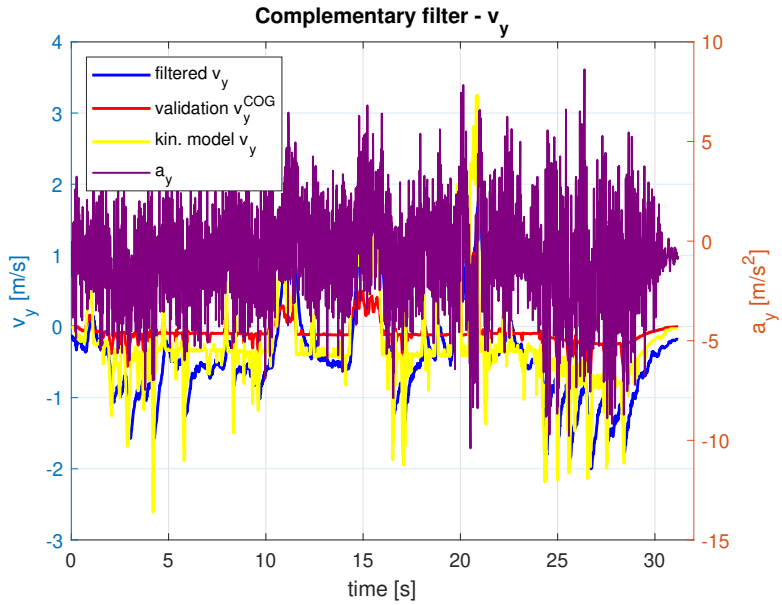


Figure 5.4: Filtered lateral velocity v_y compared to the lateral velocity of the car's COG v_y^{COG}

5.3.3 Offset

In these experiments, the ability of the EKF to react on permanent offset in GNSS data - 0.5 meters in both directions, is tested. The desired behavior of the EKF is that due to the offset, the covariance of the GNSS data is high and the filter will rely more on the model prediction, and other sources of data, therefore the drift due to the GNSS offset should be minimal. In figures 5.8a and zoomed figure, the 5 seconds of GNSS offset are marked with yellow circles.

The average distance between the EKF with offset and the EKF without offset over the 5 seconds of the skewed GNSS data can be computed as the average of,

$$d = \sqrt{(x - x_o)^2 + (y - y_o)^2}, \tag{5.2}$$

where d is the distance between the offset EKF and normal EKF position, (x_o, x) are their respective x positions and (y_o, y) are the y positions. The offset distance is $d_o = 0.7071$ m, and the average difference between the offset EKF and the normal is $d_avg = 0.6296$ m. From this data, it is easy to see that the EKF reduces the offset in GNSS data by 7.75 cm, which is 10.96% of the GNSS offset. The heading data of this experiment can be seen in figure 5.8b. During the offset data, the heading follows the same curve as the default data.

5.3.4 Sensor Outages

The sensor outages are simulated for 5 seconds, during which the EKF has to rely on other measurements to compensate for the outage.

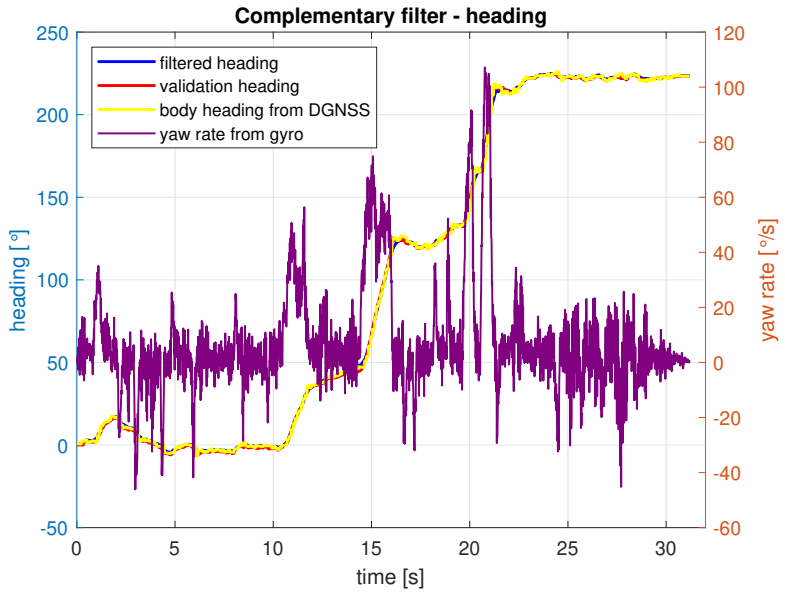


Figure 5.5: Filtered heading compared to the EKF heading

GNSS outage

In 5.9a the orange circles mark the start and end of the GNSS outage, as can be seen from the yellow line - EKF position without the GNSS data, when the signal drops out, the EKF is relying on the relative measurements from the IMU and the prediction data from the model. As can be seen, the EKF positon without GNNS data slowly starts to drift from the desired position (violet line - fully functional EKF) - the longer the outage lasts, the bigger error the positional data accumulates. After the outage, the EKF position regains accuracy almost immediately - the sharp turn in the position graph 5.11c is caused by the regained GNSS connection, the drifted position was correted by the absolute measurement from the GNSS. The average error (difference between normal EKF position and EKF position with outage) can be computed via the equation 5.2. The average error is $d_{gnss} = 61.4$ cm. The error behavior over the time of the outage can be seen in figure 5.10.

In figure 5.9b can be seen, that during the outage the heading difference grew. After the outage, the heading regained accuracy with the original EKF data. The error in relation to the original EKF heading curve during the outage is described in figure 5.9a. The seemingly large discrepancy in the slopes of the curve (heading changes during steering) between the EKF heading and the validation heading occurs due to the absence of autonomous steering - the human inputs can never precisely follow the desired path, mainly during steering and because of that the difference between the curves is created. Since the heading curves do not have increasing offset over time, the heading after finishing the turn is the same - indicating that the prevailing source of error is the author's inability to follow a turn, the EKF heading output can be considered valid.

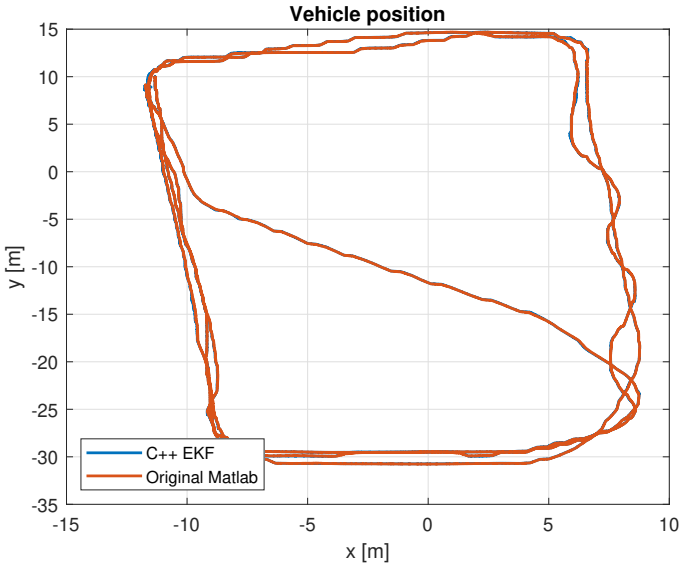


Figure 5.6: Positional data comparison of the EKF versions.

IMU outage

The start and end of the IMU outage are marked by orange circles in figure 5.11a. From the difference in the EKF position with and without the outage can be seen, that the algorithm can support itself with predictions from the single-track model and data from wheel odometry and absolute position from GNSS. From these findings can be deduced, that short periods of IMU outages are negligible. The average error calculated with 5.2 is $d_{imu} = 3.01$ cm. The behavior of the error over the outage can be seen in 5.12.

The results 5.11b are in a similar fashion as the positional data in 5.11a, the IMU outage is barely noticeable and the EKF filter can compensate for it.

5.4 Comparison

In this section, the comparison of the two sensor fusion methods, the Extended Kalman filter a the complementary filter will take place. The EKF measures six states $(\beta, \psi, \dot{\psi}, x, y, v)$ and the complementary filter only three (v_x, v_y, ψ), that leads to two meaningful ways to compare their performance.

5.4.1 Heading Comparison

The heading comparison was already done in section 5.2.3 because the EKF heading is used for validation of the complementary filter. So from figure 5.5 can be seen, that the two outputs match very closely.

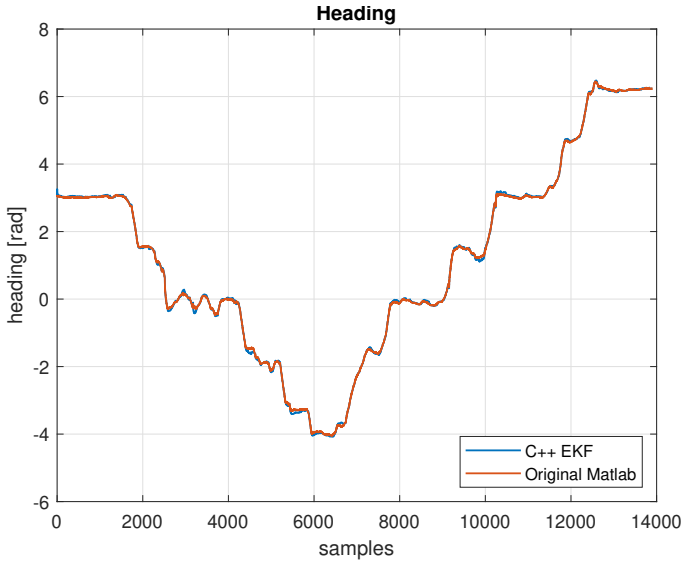


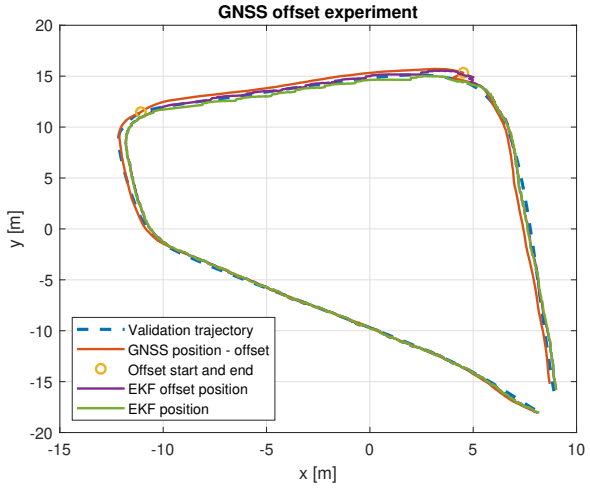
Figure 5.7: Heading data comparison of the EKF versions.

5.4.2 Velocity Comparison

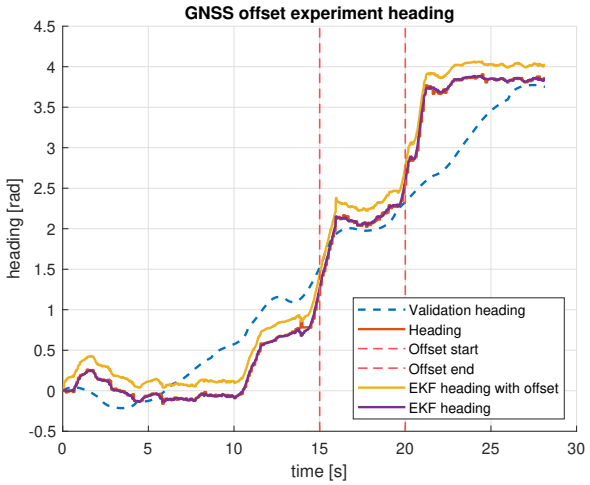
Even though the EKF filters v and complementary filter has v_x and v_y as stated the comparison can be made because of the fact, that v_x and v_y are always perpendicular and can be combined into v using the equation,

$$v_{cf} = \sqrt{v_x^2 + v_y^2}, \tag{5.3}$$

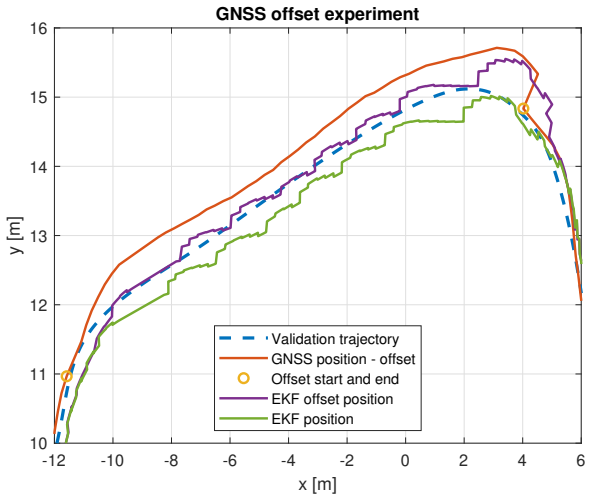
where v_{cf} is the velocity from the complementary filter, which is compared to the EKF velocity in figure 5.13. From the figure is quite clear which one is the better velocity filter. Although both velocities follow the same general curve (which is good news for the complementary filter), the v_{ekf} is clearly less noisy showcasing, that in this application, the EKF is superior to the complementary filter.



(a) : The EKF positional data with the offset in GNSS measurements.

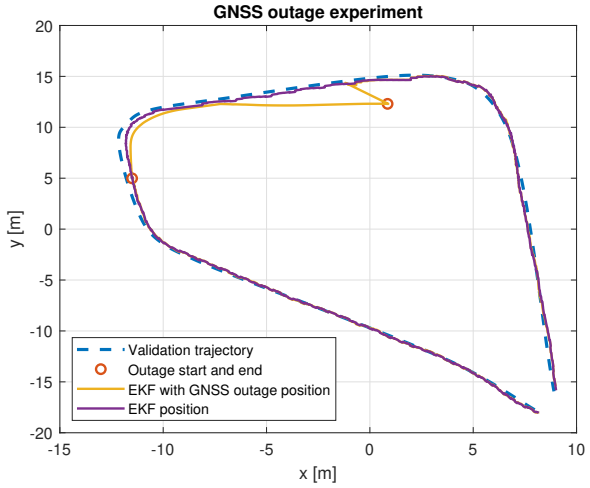


(b) : Heading data from the GNSS offset experiment.

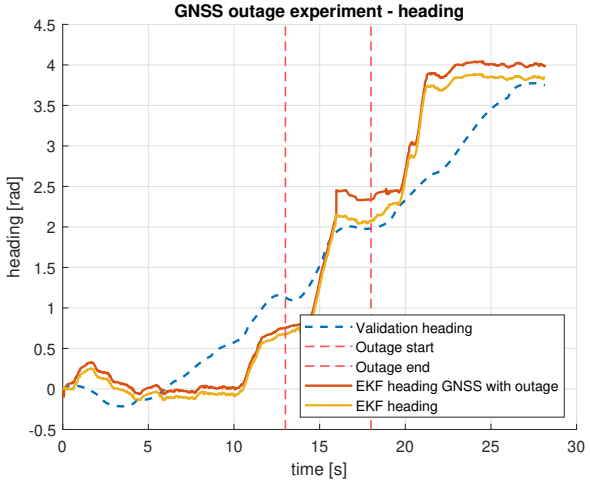


(c) : Heading data comparison of the EKF versions.

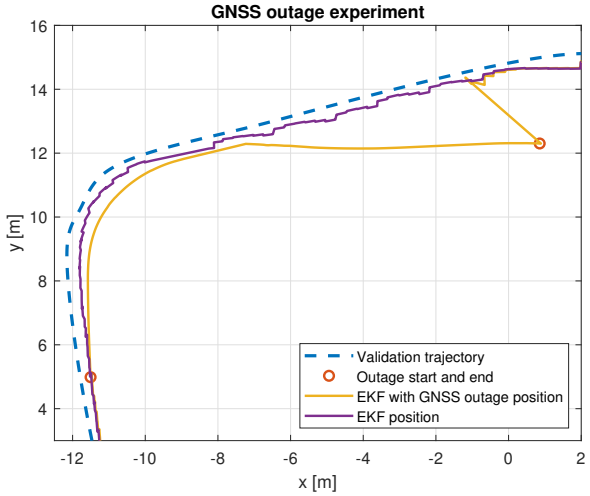
Figure 5.8: Figures for the GNSS offset experiment.



(a) : Graph of vehicle position during an experiment with simulated GNSS signal outage.



(b) : Graph of heading curves from GNSS outage experiment.



(c) : Zoomed position during an experiment with simulated GNSS signal outage.

Figure 5.9: Figures for the GNSS outage experiment.

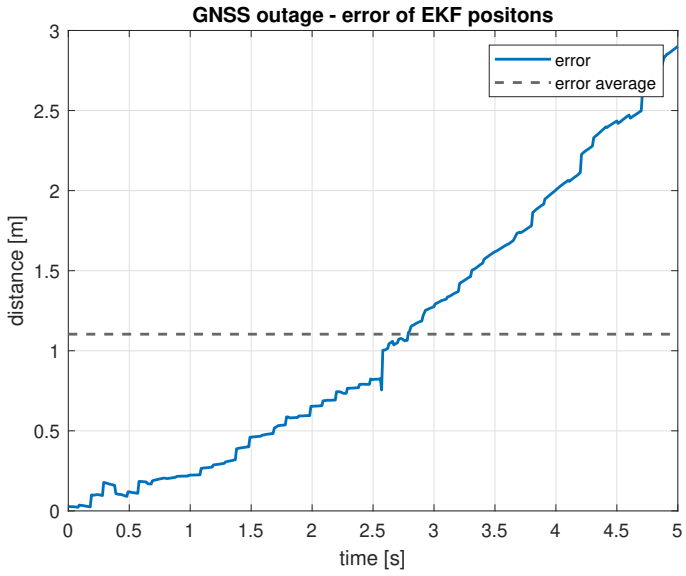
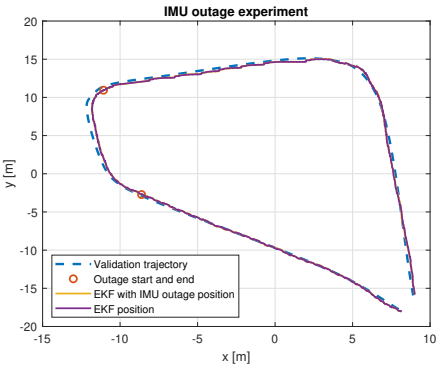
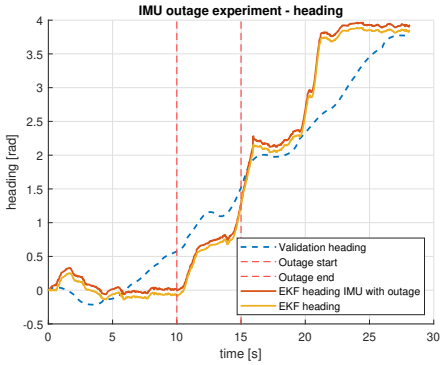


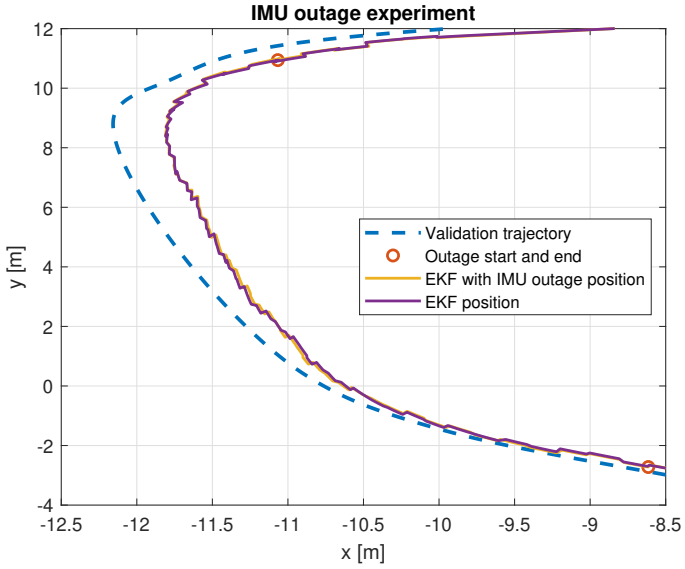
Figure 5.10: Error behavior over the time of GNSS outage



(a) : Graph of vehicle position during an experiment with IMU signal outage.



(b) : Graph of heading curves from IMU outage experiment.



(c) : Zoomed position during an experiment with IMU signal outage.

Figure 5.11: Figures for the IMU outage experiment.

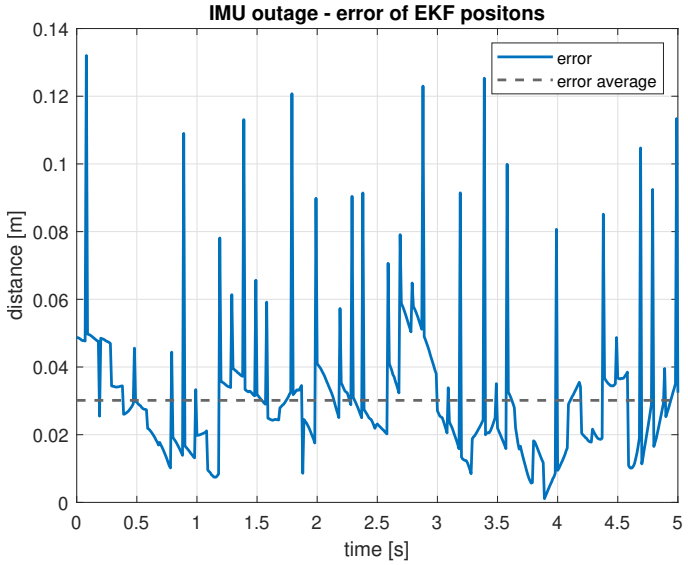


Figure 5.12: Error behavior over time of the IMU outage.

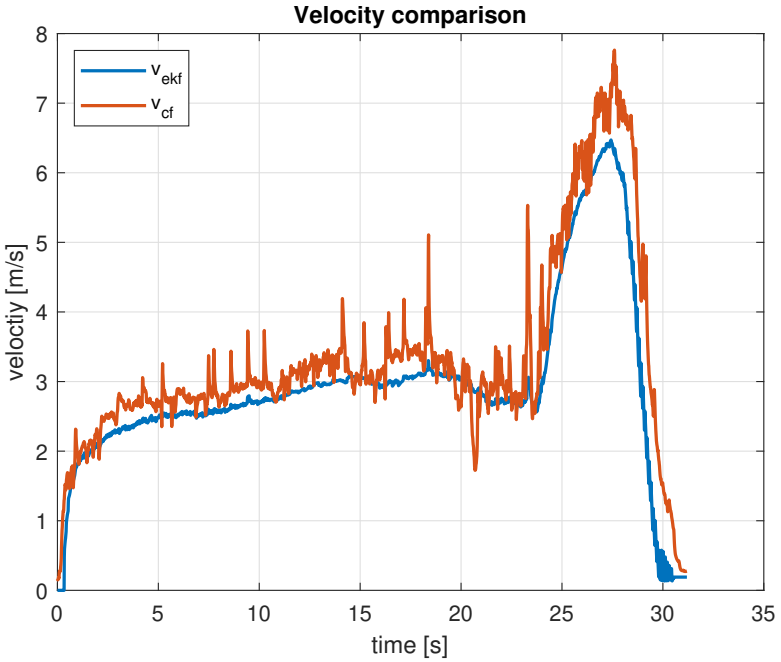


Figure 5.13: Velocity comparison between the EKF velocity v_{ekf} and the complementary filter velocity v_{cf}.

Chapter 6
Conclusions

In this chapter, a summary of the goals of this thesis and their fulfillment will be made, shortcomings found during the process of writing this thesis addressed and the solutions presented. The goal of this thesis was to implement a real-time complementary filter and EKF able to estimate the states of the overactuated vehicle platforms developed by the SDS Research Center - specifically the ToMi2 platform until the Smart Sub-Scale vehicle is completed and ready for testing.

6.1 Complementary Filter

The implementation of the complementary filter was quite straightforward, based on the signals from GNSS, IMU, and wheel odometry accompanied by the kinematic vehicle model. The baseline structure was designed in the continuous domain in Simulink and subsequently discretized. C++ code for deployment on the embedded hardware and the ROS2 environment running on the ToMi2 platform was generated from the Simulink model. The expriments to validate the correct design and tuning of the filter were successful, although the increased noisiness and imperfection due to the limitations of the used kinematic model, can be subject to future improvements.

6.2 EKF

The original implementation of the used EKF was done by Tomáš Twardzik - model-based offline EKF written in Matlab able to fuse GNSS, IMU, wheel odometry, and visual odometry data. The goal of this thesis was to adapt this EKF implementation to work in real-time on embedded hardware in the ROS2 environment used on the ToMi2 platform while making the EKF functions modular for further use on other vehicle platforms. To match these goals, C++ code had to be generated from the original Matlab code. This relatively simple-sounding task proved to be the most painstakingly difficult part of the whole project. Due to the limitations of the Matlab code generator, a vast number of adaptations had to be made to the Matlab code, the major modifications are described in section 4.2.4. After the code was

generated, the resulting C++ class was integrated into the existing ROS2 environment on the ToMi2 platform. Unfortunately, during the integration a problem with the visual odometry algorithm occurred and could not be solved by the time of this writing - the current release of the Nvidia Visual Slam algorithm seems not to work with the Zed 2i Camera present in the car. This means the EKF is currently only handling data from GNSS, IMU, and wheel odometry, with the visual odometry handling ready to use when the camera issues are resolved. Regardless of the absence of visual odometry data, the experiments conducted to validate the performance of the EKF were still successful, only in the GNSS offset experiment was the absence of the positional correction from visual odometry felt - the EKF position drifted a bit too much in the direction of the offset. Otherwise, the main goals were met - successful creation of real-time EKF able to run on embedded hardware.

6.3 Discussion

All in all, the goals set at the beginning were met, apart from the visual odometry in EKF but that does not pose any major setbacks on the functionality. The work on this thesis, although sometimes a bit boring, daunting, and difficult, was an enjoyable experience creating an opportunity to push one's boundaries and to discover new areas of knowledge and abilities, that can be leveraged in future life and work.

6.4 Future work

The main aspects calling for improvement are:

Visual odometry - wait for Nvidia to respond and update its release to support the Zed 2i camera.

New platform - part of the performance limiting factors is the vehicle platform itself. The ToMi2 is at the end of its lifecycle and the limits can be felt, given its based on an RC model car.

Sensors - a good way to improve the EKF performance is to try to equip the vehicle with better and more precise sensors. Some of them arrived during the work on this thesis, but the timeframe did not allow their integration and testing.

Appendix A

Bibliography

[1] Society of Automotive Engineering. Sae levels of driving automation graphic. https://www.sae.org/binaries/content/gallery/cm/content/news/sae-blog/j3016graphic_2021.png, 2021. Accessed: 2024-05-23.

[2] REEcorner. https://ree.auto/wp-content/uploads/2024/03/ree1-2.jpg, 2024.

[3] REE Flat bed image. https://ree.auto/wp-content/uploads/2023/04/SG_PLAFORM_LINEUP_ANGLE_v002-768x410.jpg.webp, 2024.

[4] Hyundai Mobion Concept CES 2024 Photo Gallery. https://www.autoblog.com/photos/hyundai-mobion-concept-ces-2024/.

[5] Hummer EV crabwalk. https://gmauthority.com/blog/2021/08/gmc-hummer-ev-crab-walking-on-woodward-avenue-video/.

[6] Drivepilot. https://www.greencarcongress.com/2023/09/20230928-drivepilot.html, 2023.

[7] Press - Media Resources & Self-Driving Car Images. https://waymo.com/media-resources/.

[8] Introducing the 5th-generation Waymo Driver: Informed by experience, designed for scale, engineered to tackle more environments. https://waymo.com/blog/2020/03/introducing-5th-generation-waymo-driver.

[9] Kalman filter. https://en.wikipedia.org/w/index.php?title=Kalman$_{f}ilteroldid = 1223539406, May2024. PageVersionID : 1223539406.

[10] Ina. VBOX Automotive - VBOX 3iS. https://www.vboxautomotive.co.uk/index.php/en/products/vbox-3is/vb3is.

[11] Navio2 board. https://eu.robotshop.com/cdn/shop/products/navio2-autopilot-kit-raspberry-pi-a-b-8$_{d}0569eca - dd4c - 43d9 - 8a1b - 3d8a6dcdfcf4_{6}00x.jpg?v = 1695136794, 2024.

[12] MEMS accelerometer scheme. https://www.leveldevelopments.com/wp/wp-content/uploads/MEMS-Accelerometer.png.

[13] Tian Han, Guanshi Wang, Changchun Dong, Xiaolin Jiang, Mingyuan Ren, and Zhu Zhang. A Self-Oscillating Driving Circuit for Low-Q MEMS Vibratory Gyroscopes. *Micromachines*, 14(5):1057, May 2023.

[14] SENSOR Test. https://www.hprobe.com/sensor-test/.

[15] Ritik Agarwal, Ghanishtha Bhatti, R. Raja Singh, V. Indragandhi, Vishnu Suresh, Laura Jasinska, and Zbigniew Leonowicz. Intelligent Fault Detection in Hall-Effect Rotary Encoders for Industry 4.0 Applications. *Electronics*, 11(21):3633, January 2022.

[16] O. P. T. Telescopes. Astrophotography 101: The Bayer Filter System. https://optcorp.com/blogs/astrophotography-101/bayer-filter-system, November 2020.

[17] ZED 2 Stereo Camera. https://store.stereolabs.com/products/zed-2.

[18] Denis Efremov. Single track model implementation. https://github.com/SDS-RC-FEE-CTU-in-Prague/SingleTrack.

[19] Google Maps.

[20] Society of Automotive Engineering. Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. `https://www.sae.org/standards/content/j3016_202104/`, 2021. Accessed: 2024-05-23.

[21] De Jong Yeong, Gustavo Velasco-Hernandez, John Barry, and Joseph Walsh. Sensor and sensor fusion technology in autonomous vehicles: A review. *Sensors*, 21(6), 2021.

[22] Mary B. Alatise and Gerhard P. Hancke. A Review on Challenges of Autonomous Mobile Robot and Sensor Fusion Methods. *IEEE Access*, 8:39830–39846, 2020.

[23] Ina. VBOX Automotive - VBOX 3iS. https://www.vboxautomotive.co.uk/index.php/en/products/vbox-3is/vb3is.

[24] Adam Van Gunter. Complementary_filters. https://vanhunteradams.com/Pico/ReactionWheel/Complementary$_F$*ilters.html*.

[25] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[26] REECorner | Modular Electric Vehicle Platform. https://ree.auto/technology/.

[27] Newsroom | Hyundai Mobis. https://www.mobis.com/en/index.do.

[28] Eric Stafford. 2024 GMC Hummer EV Pickup Review, Pricing, and Specs. https://www.caranddriver.com/gmc/hummer-ev, December 2023.

[29] Tesla Autopilot. https://www.tesla.com/support/autopilot.

[30] Tesla transitioning vision. https://www.tesla.com/support/transitioning-tesla-vision.

[31] DRIVE PILOT Automated Driving. https://www.mbusa.com/en/owners/manuals/drive-pilot.

[32] Daniel Golson. We put our blind faith in Mercedes-Benz's first-of-its-kind autonomous Drive Pilot feature. https://www.theverge.com/2023/9/27/23892154/mercedes-benz-drive-pilot-autonomous-level-3-test, September 2023.

[33] Autonomous Driving Technology - Learn more about us. https://waymo.com/about/.

[34] What is an Inertial Navigation System? https://dewesoft.com/blog/what-is-inertial-navigation-system.

[35] Twardzik Tomáš. Fúze senzorickỳch dat polohy pro autonomní vozidla. Master′sthesis, eskvysokuentechnickvPraze.Vypoetnainformancentrum., 2022.

[36] Encoder Products Company. What Is An Optical Encoder? Everything About Optical Encoders. https://www.encoder.com/article-what-is-an-optical-encoder.

[37] Rutrle Tomáš. Vývoj verifikační platformy pro přeaktuovaná vozidla. B.S. thesis, České vysoké učení technické v Praze. Vypočetní a informační centrum., 2020.

[38] Specifications | Navio2.

[39] Nvidia jetson agx xavier. https://www.nvidia.com/en-gb/autonomous-machines/embedded-systems/jetson-agx-xavier/.

[40] BeagleBone® AI-64. https://www.beagleboard.org/boards/beaglebone-ai-64.

[41] Aceinna: Leader in MEMS Sensor Technology. https://www.aceinna.com.

[42] ZED-F9P module. https://www.u-blox.com/en/product/zed-f9p-module, April 2023.

[43] Specifications - SOLO UNO V2 - Solo Motor Controllers. https://www.solomotorcontrollers.com/specifications-solo-uno-v2-slu0722-5832/.

[44] Zotac e-series. https://www.zotac.com/us/product/mini$_p$cs/magnus − en374070c − barebone.

[45] R Rajamani. *Vehicle Dynamics and Control*. Springer, 01 2006.

[46] Simon J. Julier and Jeffrey K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92:401–422, 2004.

[47] H. Pacejka. *Tire and Vehicle Dynamics*. Elsevier Science, 2005.

[48] Wikipedia. Centripetal catmull–rom spline. https://en.wikipedia.org/wiki/Centripetal$_{C}atmullOct$2023.

[49] Nvidia-Isaac-Ros. Nvidia-isaac-ros/isaac-ros-visual-slam: Visual odometry package based on hardware-accelerated nvidia elbrus library with world class quality and performance. https://github.com/NVIDIA-ISAAC-ROS/isaac$_{r}os_{v}isual_{s}lam$.