POLITECNICO DI TORINO

Department of Mechanical and Aerospace Engineering

Master of Science in Mechatronic Engineering

Master's Degree Thesis

Robust Localization for an Autonomous Racing Vehicle

Supervisor: Prof. Andrea TONOLI

Co-supervisors: Prof. Nicola AMATI

Eng. Stefano FERACO

Candidate:

STEFANO FAVELLI

265499

Academic Year 2020 - 2021

Dedicata a...

Una nuova fine e un nuovo inizio...

Questa tesi rappresenta la fine di un ciclo bellissimo, un'esperienza che non sarebbe stata la stessa senza Diana, a cui devo l'ispirazione, la forza e il coraggio di questi anni. E' stato difficile, ma ci siamo riusciti, ancora una volta, insieme.

Ringrazio la mia famiglia e i loro sacrifici, impalpabili ma costanti, che mi hanno permesso di essere qui oggi.

Un grazie anche a tutti gli amici, a quelli di sempre e a quelli incrociati, persi e ritrovati durante il percorso, siete stati degli ottimi compagni di viaggio.

Infine, un ringraziamento speciale va a Stefano, per avermi supportato e sopportato in questi mesi di lavoro e di stesura della tesi, e al Professor Andrea Tonoli, per la fiducia garantita dall'inizio dell'avventura in Squadra Corse e il rispetto reciproco che ne è derivato.

Ad meliora et maiora semper,

Abstract

In the framework of autonomous driving, one of the most challenging problems to be solved is represented by the localization task. Considering the self-driving vehicle as a robotic system, the localization problem arises with motion in an unknown environment and relates to the capability of the system to know at each instant its own positioning with respect to a fixed reference. Knowledge of position and orientation is referred to as pose and it is related to the state estimation capabilities of the control system. In autonomous navigation and driving, state estimation is a key feature to achieve robustness, repeatability and high-end performances of control actions. In this scenario, autonomous racing represents the most challenging ground for testing the effectiveness of autonomous algorithms, thanks to the wide range of dynamical and physical conditions presented by the racetrack.

The aim of this thesis work is the deployment of a robust and cost-effective localization architecture for the first autonomous racing prototype of Politecnico di Torino, participating to Formula Student Driverless (FSD) competition. Starting from the localization problem statement, the thesis focuses on the investigation of the main methods to achieve robust positioning and effective state estimation. The solution is guaranteed by well-known filtering techniques, exploiting both self-contained information and global measurements of vehicle's states. The investigated method proposes and compares the application of Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) to the sensor fusion of on-board data streaming from an Inertial Navigation System (INS) and a Global Navigation Satellite System (GNSS) sensor.

The design of the hardware layout and the software architecture are both presented, discussed and experimentally validated in real time, using a properly instrumented all-wheel drive electric racing vehicle, developed by Squadra Corse PoliTo. The proposed algorithm deployed on an on-board high-performance computing platform, has proven to achieve sub-meter accuracy in positioning tasks, also for short outages of GPS signal. Besides, the whole architecture provides precise state estimation for the retained single-track vehicle model, exploited for further control strategies. The experimental results show a substantial equivalence of the application of the two filters considered. Nevertheless, the UKF-based method is characterized by a lower estimation variance in the localization task, providing more robust results and thus is chosen for the final implementation on the vehicle.

Contents

Chapter 1

Introduction

Motorized transportation has shaped the world since its introduction, with dramatic evolution during the last couples of centuries. Autonomous Vehicles (AVs) are about to further revolute the way we live and how we perceive transportation, because of their technological and socio-economic impact. Individual transportation is not the only area in which AVs will be deployed and a significant change of perspective is already experienced in public transportation, delivery and cargo operations, as well as in the sector of speciality vehicles for farming and mining activities.

People's needs and demands are evolving fast together with the development of new technological innovations in the field of transportation and the public's experience of the benefits introduced by partially automated or autonomous systems, is increasing the expectations for the future developments. The improvement of life's quality promised by the new paradigms of the automotive industry in terms of safety and environmental impact of the technologies adopted, is the main drive of a market aware of the topics, but also worried about the possible outcomes. For this reason, major concerns on the widespread diffusion of autonomous technologies in transportation remain unchanged, both from the technical point of view as well as from the legal and ethical perspective. The global effort of engineering academic institutions and private companies worldwide on autonomous driving, is focused on the technological advancement required in different fields, to allow the increase in currently adopted automation levels, and to let this breakthrough to happen.

1.1 Background

As AVs are experiencing this increasing interest worldwide, a huge research effort to continuously address design challenges related to safety and performances has been put in place both by traditional automotive OEMs as well as a great number of start-ups and newcomers [1] [2]. Many Advanced Driving Assistance Systems (ADAS) are already present in the majority of the vehicles of the recent mass productions, thanks to recent developments of engineering methods in the real-time assessment of vehicle's dynamics and passengers' comfort, as well as in the development of robust safety systems [3] [4] [5]. The ADAS are one of the first steps towards vehicle automation introduced in the automotive field and represent for the general public a reliable and appealing feature for a new vehicle purchase, mainly thanks of the increased level of safety perceived.

However, fully autonomous commercial vehicles are still far from being introduced in the automotive market, although they are retained to change the whole mobility panorama in the next decades, thanks to the contribution of Artificial Intelligence (AI), as stated in [6] and [7]. The technological limitations of the solutions available at the moment, overlap with the limited trust of the customers towards the technology and its adoption at daily basis. In this scenario, research efforts are focused on increasing the reliability of the systems available and allowing the development of affordable safety-oriented solutions to impact the general perception of the technology. The transportation habits on the planet are known to be source of an alarmingly high number of deaths every year related to transportation. Road accidents and pollution are the main causes of this trend, so the transition towards a safer and more sustainable future in the way we move ourselves and our goods, is not only needed but somehow inevitable. Being ready for the technological challenges ahead, will be of primary importance for any player in the automotive and transportation industry in the years to come, to take the most effective actions.

When talking about AVs from a more technology-oriented perspective, every vehicle can have a range of autonomous capabilities allowed by its on-board equipment. To enable a common classification of autonomous systems, the Society of Automotive

Engineers (SAE) International established in the early 2014, its SAE J3016 "Levels of Automated Driving" standards [8]. The guidelines, adopted both by the United Nations and the US Department of Transportation, have been updated in 2018 and are now considered as the industry standard for autonomous driving technologies.

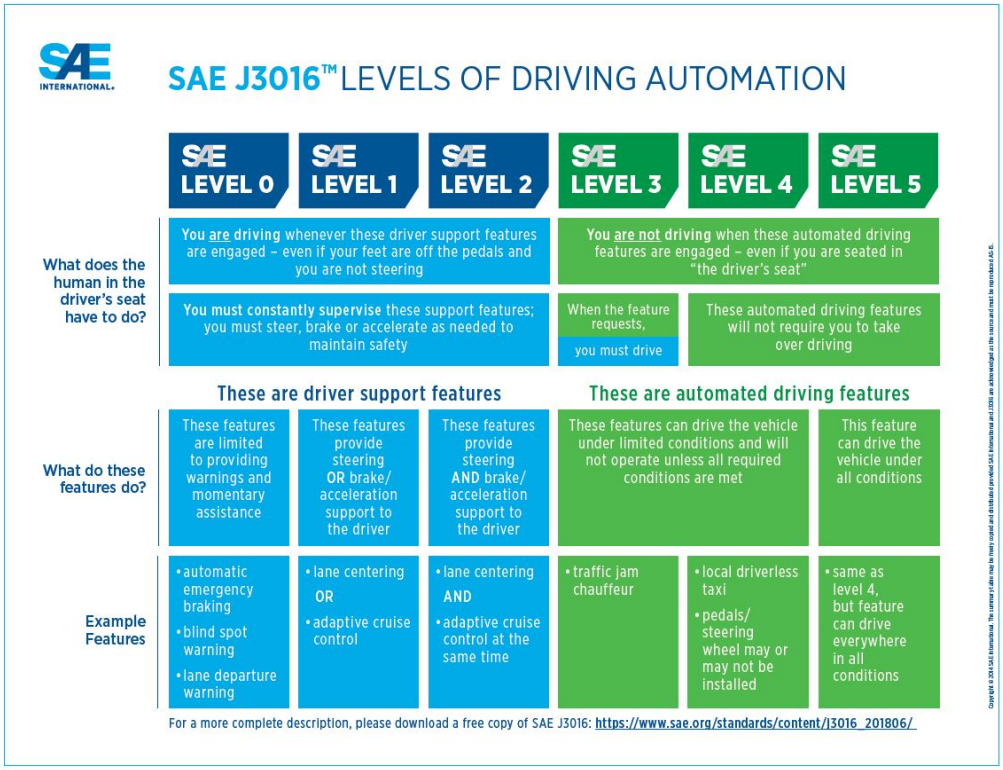


Figure 1.1: *SAE Levels of Driving Automation.*

The classification, reported in Figure 1.1, is made on a 0-5 level, in which an increasing number represents the greater autonomy of the system from a human supervisor. The distinction of driving-assistance systems into the scale is based on the level of responsibility and attentiveness required to the human driver, which is completely unnecessary at Level 5. The most automated individual transportation systems available on the market today, only achieve Level 2 automation, when the driver still needs to monitor the system and decide when take over control of the vehicle, as for example with Tesla's Autopilot. This is mainly imposed by current legal restrictions in the road usage of the technology; nevertheless, Level 3 and 4 of automation have been achieved and successfully tested in different scenarios, where the environment conditions allow them to be applicable.

On the other side, no technology is yet capable of achieving full automation at Level 5 and some experts claim that this level will never be achieved either. This is due to the dependence of the level of autonomy on the environment in which the AV is operating, including road rules, culture, atmospheric conditions and all the contouring conditions which a human driver is capable to naturally consider. Anyway, SAE's levels are more focused on the technical aspects rather than the legal or the cultural ones, thus they are helpful to clarify the role of the Automated Driving Systems (ADS) which are progressively developed and adopted.

The six resulting levels of classification are defined as follows:

- *Level 0* - No automation: steering or speed control may be momentarily assisted by the vehicle, but the human driver is in charge of all the aspects of driving;
- *Level 1* - Driver assistance: longitudinal or lateral support under well-defined driving scenarios (e.g. highway) are guaranteed, because the vehicle takes over either the speed or the steering control on a sustained basis;
- *Level 2* - Partial automation: both speed and steering control are taken over by the vehicle, therefore continuous longitudinal and lateral support under well-defined driving scenarios are guaranteed. Nowadays, available Level 2 vehicles are equipped with a wide set of ADAS;
- *Level 3* - Conditional automation: the vehicle becomes capable of taking full control under well-defined driving scenarios, but the driver must be always in the condition of suddenly taking back control when required by the system;
- *Level 4* - High automation: human interaction is not needed any more, the vehicle takes full control and complete a journey in full autonomy, under limited driving scenarios and environment conditions. Pedals and steering wheel are likely to be still present in the vehicle, to guarantee the possibility to drive in scenarios that go beyond the defined uses cases;
- *Level 5* - Full automation: the vehicle takes full control under all driving scenarios, no more provisions for human control are present, so only the concept of passenger exist.

The definition of the Levels of Driving Automation is considered also in this thesis as the background for the development of an ADS, thus the problem will be considered only from a technological point of view. In particular, this thesis work is focused on one of the software stacks required to compose a complete ADS, with the scope of solving the "localization problem" to allow autonomous navigation. The ADS considered is developed at Politecnico di Torino by the research group of Prof. Andrea Tonoli and Prof. Nicola Amati at *DIMEAS - LIM Mechatronics Lab* (`http://www.lim.polito.it/`). The system is devoted to allow the autonomous conversion of an electric racing prototype designed and developed by the student's Team Squadra Corse PoliTo participating to the Formula Student Driverless Competition (FSD).

1.2 Motivation

The result of the progressive change of perspective in transportation, is pushing the development of new competences in the field of autonomous driving and many opportunities of research under different topics are available at academic level. The main motivation for this thesis work, has been to implement a software component participating in the deployment of an ADS, given a set of performance requirements and a development hardware platform.

Tackling the challenge of designing and deploying an ADS is complex and requires multi-fields competences, which is justified by the number of tasks which the system is asked to simultaneously consider and control. The vehicle is considered as a mobile robot which must be able to perceive its surroundings and its internal states to have a complete and reliable knowledge of its operational environment. A first set of requirements comes from the perception task, which mainly involves the research of the optimal trade-off between the computational effort required by the pipeline and the accuracy of the measurements and the states sensed or estimated. From a software point of view, perception is normally considered as the first pillar of driving automation, on which the planning and control pipelines rely to take decisions and modify accordingly the states of the mobile robot.

The autonomous system design usually involves a "sense-plan-act" strategy, supported by the following hardware components:

- *Sensors*: Multiple sensors are used to perceive internal states of the system and acquire information on the immediate surroundings (external conditions, landmarks, other objects, etc.);
- *Logic Processing Unit*: On-board embedded computer to process the inputs and plan the needed control actions;
- *Actuators*: Mechanical servomotors, electronic relays and control systems, dedicated to the realization of the planned actions.

The fields involved in each step of the strategy are multiple and the aim of this thesis is to cover the design of a valid method to solve the localization task to allow autonomous navigation. The discussion presented in the following chapters, addresses in detail only the perception pipeline, which involves mainly the fields of sensing and processing. Starting from the requirements definition, the design of the method is tackled both from an hardware and software point of view, because of the strong integration required to guarantee the expected performances. The results obtained by the adopted solution are presented and discussed with the aim of verifying the requirements and complete the study on the solution of the localization problem.

Localization has been recognized as a key enabler for the development of any technology devoted to self-driving cars [9] [10], representing one of the fundamental components of an effective perception pipeline. Considering the AV as a traditional robotic system, the localization problem arises with the vehicle's motion in an unknown environment and becomes more critical when the effectiveness of path planning or vehicle dynamics control algorithms is tested in the limit conditions presented by challenging driving scenarios, such as the racetracks or urban areas, which are considered in this work.

Strictly linked to localization, the precise real-time vehicle's state estimation is another crucial topic to be addressed by any ADS, because it directly impacts the performance of every control algorithm devoted to the driverless system. For this reason, also the accurate and robust estimation of the vehicle's pose (position and

orientation) and states is a goal of the method presented, which has been motivated by the need of an accurate and robust real-time positioning system to deploy an effective local path planner [11] [12]. In the retained vehicles, the choice of a local path planner for autonomous driving is given by the unavailability of an a-priori knowledge of the driving environments [13] [14].

1.3 Formula Student Driverless (FSD)

The opportunity of development of this thesis project has been given in the context of the Formula Student Competition by the Formula Student Driverless (FSD) category. The software components devoted to localization presented in the following, are part of the ADS of the first autonomous prototype of Politecnico di Torino, participating to the FSD events. In Figure 1.2 it is presented the first ever European FSD event, the Formula Student Germany Driverless 2017, taking place every year in Hockenheimring since then.

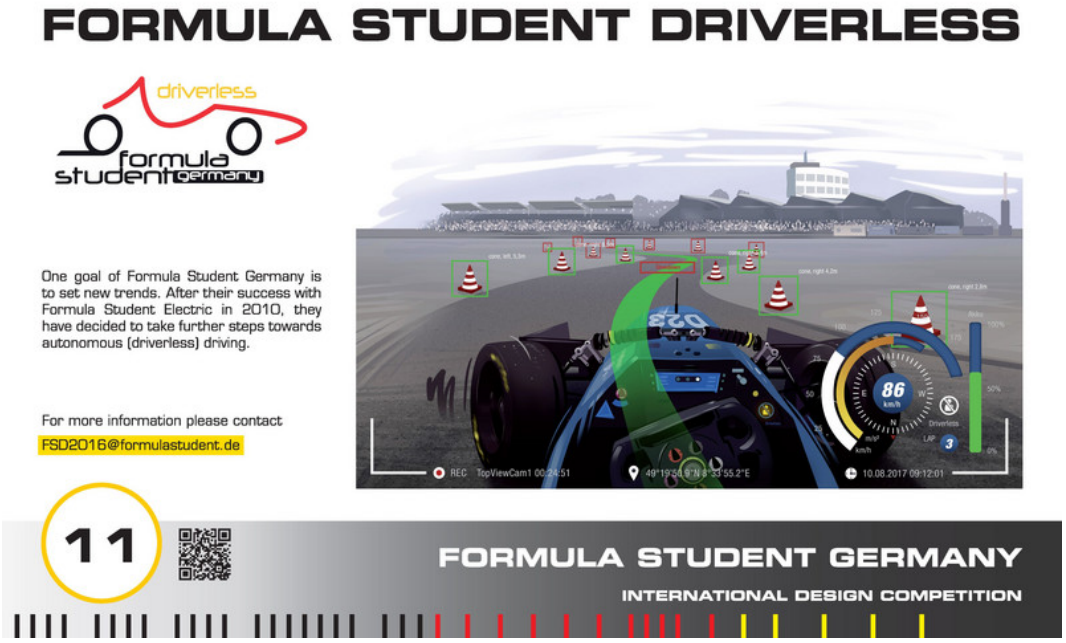


Figure 1.2: *Presentation of the first ever Driverless event in Formula Student Germany 2017.*

Formula Student or, with its original name, Formula SAE, is a student design competition organized by SAE International (previously known as the Society of Au-

tomotive Engineers, SAE). Started in the USA in the 80s, Formula SAE challenges students' teams from all around the world to design, build, test, and race a small-scale formula-style racing car. Each students' team presents a prototype based on a series of rules [13], whose purpose is both to ensure on-track safety, given that the cars are built and driven by students, and to promote competence's growth and clever problem solving.

The prototype race-car is judged in a number of different events. In the Driverless category, the points schedule for the Formula Student events is presented in Figure 1.3, following the official rules of the competition [13].

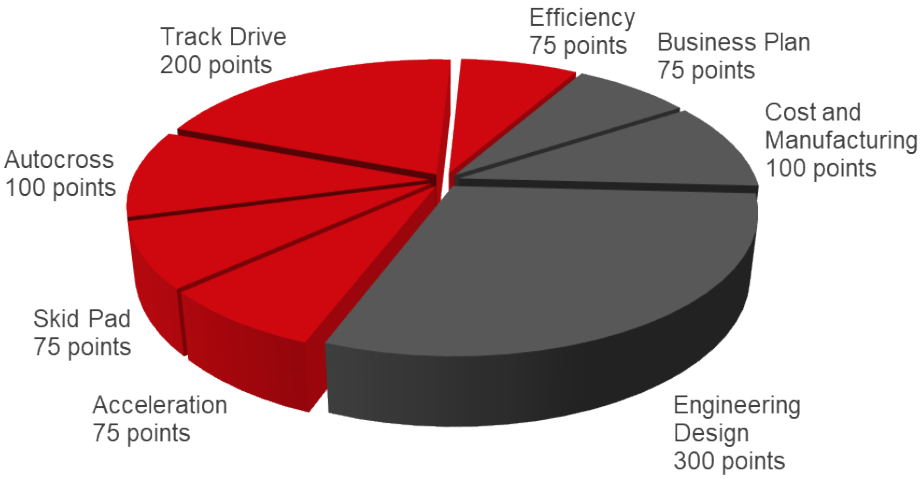


Figure 1.3: *Scoring of Formula Student Driverless Competition. The red part represents Dynamic Events scores while the grey one the Static Events scores, giving a total of 1000 points.*

Formula Student challenges the team members to go beyond their educational background, by incorporating intensive practical experience in building and manufacturing as well as considering the economic and management aspects of the automotive industry. The challenge is to compose a complete package consisting of a well-constructed race-car and a sales plan that best matches these given criteria. The decision is made by a jury of experts from the motor-sport, automotive and supplier industries. The jury judge every team's car and sales plan based on engineering innovation, construction quality, and cost planning. The rest of the judging will be done out on the track, where the students demonstrate in the dynamic disciplines how well their self-built race-cars behave in their true environment.

The Driverless category of Formula Student has been introduced in 2017 to allow the students to tackle the challenge of designing, testing and racing an autonomous prototype. The spirit of the competition which is more engineering than motor-sport oriented, allows the development and deployment of innovative technological solutions to autonomous driving problems in a controlled and safe environment. The background of the competition is an important stimulus for the research of new methods and many contributions by different universities worldwide is coming directly from the competences acquired during FSD events.

Politecnico di Torino has been one of the first Italian universities to participate in Formula SAE events, strongly believing in the opportunities for the students' competences growth enabled by the competition. The Formula SAE team from Politecnico di Torino is Squadra Corse PoliTo, founded in 2006 under the Department of Mechanical and Aerospace Engineering (DIMEAS) and participating to the electric category since its introduction in 2012. In 2021, the team will present its first driverless prototype, based on the 2019 season electric racing prototype, winner of the 2019 edition of the Formula SAE Italy event in the electric category: the *SC19D* presented in Figure 1.4.

Figure 1.4: *Squadra Corse PoliTo SC19 prototype, winner of Formula SAE Italy Electric 2019 at Varano de' Melegari, Italy.*

1.4 State-of-the-Art

Formula Student Driverless (FSD) gives the unique opportunity to engineering students from all over the world to develop and test innovations in the field of autonomous systems design. As we have introduced so far, this thesis work tackles one of the many challenges proposed by this design and an important background of researches and previous works can be found in dedicated literature.

The investigated method is addressed to the properly instrumented all-wheel drive electric racing vehicle, developed and produced by Squadra Corse PoliTo, and a commercial compact SUV. The racing vehicle is retained as a development platform for the proposed algorithm that is validated on a commercial compact SUV to prove responsiveness, accuracy and scalability of the proposed algorithms. The problem of finding the best trade-off between those features is a common design challenge when addressing robust positioning and localization for ADSs and has been tackled with different approaches in literature, as investigated by [15] [16] [17] [18]. Nevertheless, all the aforementioned research works solve the localization problem together with the mapping problem by means of the well-studied Simultaneous Localization and Mapping (SLAM) algorithms, embedding different state estimation techniques and hardware setups.

In general, SLAM algorithms are a well-documented solution in literature [19] and have proven to be a robust and reliable source of state estimation also for racing applications. However, the main drawback of the SLAM approach when dealing with unknown racetracks or urban maps, is the need of a software stack able to run a global mapping routine during a first low-pace lap of the course to achieve a good level of details of the map. Then, the vehicle can navigate for the remaining laps in a perfectly known environment in a pure autonomous roaming condition. This process is poorly applicable to commercial vehicles in urban areas: accurate digital mapping is one of the main concerns of the autonomous technology providers at the moment, but the limitation of the approach is the impossibility of tackling for the first time an unknown environment, with an high level of confidence. Moreover, in the autonomous racing scenario, it means to sacrifice the dynamic performance of the vehicle during

the initial lap, while building the environment for a global path planner. Conversely, the localization method investigated and proposed in this thesis, aims to provide an accurate real-time localization to enable a local path planner that partially emulates the behaviour of a human driver. The goal is to ensure the needed short-term and high-end performance, even in the case of facing the racetrack for the first time, which is the main challenge proposed by the concept of FSD dynamic events. Therefore, the localization algorithm has a pivotal role for assuring the responsiveness of the entire autonomous system, because it enables the ability of the vehicle to be self-aware of its position on the track and its dynamic conditions at any time with the highest level of accuracy.

In this work, the performance of two renowned filters for non-linear models, namely the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF), are compared in estimating the position and orientation of the retained vehicles. Furthermore, the vehicle's states are also estimated for the real-time assessment of the retained single-track vehicle model that is exploited for further control strategies. The two investigated filters fuse the measurements taken by a Global Navigation Satellite System (GNSS) sensor and an Inertial Navigation System (INS) mounted onboard the retained vehicles. The application of the EKF and UKF with sensor fusion of GPS and INS sensors have been individually validated in recent literature works for the assessment of vehicle's localization in several navigation algorithms [20] [21] [22], while the goal of this thesis has been to provide an effective comparison of the two filters in action on the considered vehicles in order to deploy the most effective solution for the considered application.

The proposed solution is experimentally tested on the autonomous racing prototype and then validated on the commercial compact SUV. Furthermore, a localization algorithm should guarantee a low computational effort for the devoted computational platform, while assessing the estimation of the vehicle's position and orientation and the estimation of the vehicle dynamics states for the local path planner. In this work, it is also shown how to partially decentralize the sensor fusion routine at hardware-level using an industrial-grade navigation sensor, combined with the aforementioned filtering techniques deployed on-board for redundancy. The advantage of this approach is to

increase the robustness of the localization pipeline that can be reached with the hardware components present on-board. Eventually, localization robustness can be further refined and improved by adding measurements from other sensors if available, such as vision-based or odometry sensors. However, this procedure can cause a reasonable degradation of responsiveness and an increased level of complexity, thus being beyond of the scope of this thesis.

1.5 Thesis Outline

This thesis work is structured as follows:

1. *Chapter 2* presents the theoretical background of the topics presented, with a particular focus on filtering techniques and vehicle modelling.

2. *Chapter 3* is dedicated to the design and the implementation of the proposed method. Starting from the requirements definition, the hardware setup is presented first and then the software architecture design is discussed, with particular focus on implementation and tuning of the filters used for sensor fusion.

3. *Chapter 4* presents the experimental validation process setup and the obtained results. The analysis is concentrated on two different datasets acquired on-board the considered vehicles, representing the different driving scenarios of interest. The reported results include estimated vehicle's states, such as vehicle's position and orientation, RPY angles, velocity and acceleration measurements.

4. *Chapter 5* is the final chapter, where conclusions and future works are reported.

Chapter 2

Theoretical Background

Before diving into the discussion of the proposed method, this chapter is dedicated to the presentation of the theoretical background considered during this work. The solution of the localization problem to enable autonomous navigation deals with a number of different engineering topics which are at the base of the final pipeline design.

Localization is part of the perception task of the ADS, so a pivotal role is played by the sensing and processing techniques used. The central topic of this chapter is the introduction of the sensors used to acquire the external measurements from the environment and the internal states of the vehicle. The first part of the analysis focuses on the conventions used in navigation theory to acquire global positioning information and on the data needed for localization purposes. Then, the presentation of the processing methods used, which belong to the family of the sensor fusion techniques, is carried out. Sensor fusion is used to process the data acquired by sensors both at hardware level, if low computational effort is required, and at central level, where more complex implementations accounting a larger number of inputs are possible. Among the possible sensor fusion algorithms, this work considers only the well-documented class of the filtering techniques, which involves the probabilistic approach to sensor modelling and measurements acquisition.

As a conclusion of this chapter, the dynamical system's modelling is presented. Modelling covers a fundamental role in all robotics applications, because allows to introduce a custom level of approximation of the problem and account for strong non-

linearities in an effective way, both from a theoretical and an implementation point of view. In the following, mainly vehicle modelling is presented and analysed in detail; whilst sensor modelling represents an equally important part of the discussion for sensor fusion, it is out of the scopes of this work, and thus a deeper insight is omitted.

2.1 General Concepts

In Robotics, localization denotes the robot's ability to determine its own position and orientation within a specific frame of reference. Path planning is considered as an extension of localization, because both the robot's current position and the position of the goal location are required and should be determined within the same frame of reference or coordinates' system. The process of robot navigation is also related to the map building, which can be in the shape of a metric map or any notation describing visited locations in a common frame of reference. In this work, the map building problem is not solved by the localization pipeline proposed and it is treated separately by the whole perception algorithm, discussed in other thesis works.

The basics to be analysed to solve localization, refer to two main aspects related to the conventions used to represent the robotic system in the environment: navigation techniques and coordinate systems. Navigation is the first concept introduced: the different techniques that can be used to effectively track the position and orientation of a mobile robot in space are briefly presented and discussed. On the other hand, when dealing with GNSS sensors as in the case of this thesis work, the relations between coordinate systems for position representation, must be well-known to effectively study navigation. For this reason, the conventions used to determine the reference frames for the path planning and positioning tasks are here presented and analysed.

2.2 Navigation

Navigation or, more specifically in our case of interest, land navigation, is the field of study that focuses on the process of monitoring and controlling the movement of a terrestrial vehicle from one place to another on Earth. The problem arising with

navigation, can be solved by means of different techniques, which are based on the knowledge of the conventions used to describe positions and orientations of objects on the Earth's surface or immediate surroundings.

The conventions related to positioning are imposed by the studies of Geodesy, the science of measuring and understanding Earth's geometric shape, orientation in space and gravitational field. In Geodesy, the Earth's physical surface (land or sea in contact with the atmosphere), is modelled as a Geoid. The Geoid is a model of Earth corresponding to the equipotential gravitational field surface that is best fitted to the global mean sea level. It is an imaginary surface beneath the land surface, which corresponds to the shape that the oceans would take if there were no solid land, but only the Earth's gravity field applied to them. However, the Geoid as a geometrical surface is too complex to be used in navigation tasks.

The commonly used approximation in the Reference Ellipsoid, with minor axis coinciding with the Earth's rotational axis and centre coinciding with the centre of mass of the Earth. The ellipsoid model obtained is suitable for navigational purposes due to its simple analytical expressions, which are discussed in the following. Within the definition of this model, also the concepts of Meridians and Parallels of Latitudes are presented. A Meridian is an arc that stretches between the north pole and the south pole and is perpendicular to the equator. The Prime Meridian passes through Greenwich, England, and is set to be the longitude reference. On the other hand, the Parallels are imaginary circles, parallel to the equator, which is set to be 0 degrees of latitude as fixed convention; this makes the poles the points with 90 degrees of latitude.

The problem of navigation is nowadays easily solved considering these conventions and different techniques are commonly available to track positions of objects within this common frame. Navigation can be generally self-contained or can exploit absolute measurements of external objects; most of the times, the two techniques are used in combination, and measurements are fused to give an accurate estimation of positions and velocities. In general, self-contained methods are referred to as dead-reckoning methods, while absolute measurements fall under the category of global positioning techniques.

2.2.1 Dead-Reckoning Techniques

When exploiting a self-contained method, by starting at a known position and measuring heading and velocity at known time intervals, the position can be estimated though kinematics relations. In dead-reckoning no external measurements are used, so any error at any stage of the measuring process can propagate and accumulate during the computation of the following instants. This can eventually cause unbounded growth of errors, usually referred to as drift.

A common source of error propagation in dead-reckoning is integration drift. As an example, following the basic schematic illustration of Figure 2.1, accelerometer measurements (external specific forces) and gyroscope measurements (angular velocities) are normally integrated by embedded signal processing routines to obtain respectively positions and orientations. If the initial pose is known, and if perfect models for the inertial measurements are used, the process illustrated would lead to perfect pose estimates. However, in practice, inertial measurements are noisy and biased, and the integration steps from angular velocities to rotations and from accelerations to positions, introduce integration drift [23].

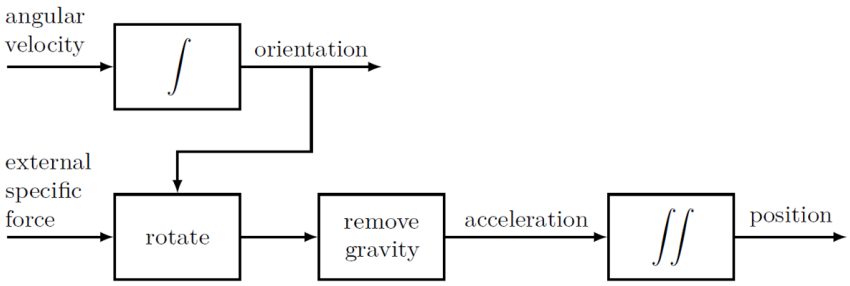


Figure 2.1: *Schematic illustration of dead-reckoning technique: the accelerometer measurements (external specific force) and the gyroscope measurements (angular velocity) are integrated to obtain position and orientation.*

Odometry

The most common type of dead-reckoning technique is odometry, in which the distance travelled by a wheeled vehicle can be estimated by counting the rotations of its wheels. The measurement of the angular displacement of the wheels can be acquired by using

rotary encoders and then it is translated into linear displacement by integration. The main problem of odometry techniques is the propagation of any error on the rotation measures which are integrated to obtain angles and consequently positions. The most common source of integration drift is introduced by wheel slippage, where the angular rotation does not correspond to any linear motion. However, the advantages of odometry sources are the very low costs, the high sampling times possible and the accuracy achieved in the short-range.

Inertial Navigation

Inertial navigation methods use inertial sensors such as accelerometers and gyroscopes to achieve self-contained measurements of the states. The most common type of inertial sensor is represented by Inertial Measurement Units (IMUs), which provide linear accelerations using one or more accelerometers, and rotational rates using one or more gyroscopes. Eventually, integrating the information from the gyroscope with a magnetometer is a common method to obtain a heading reference (referred to as attitude in Robotics). Typical configurations contain one accelerometer, one gyroscope, and one magnetometer per axis for each of the three principal axes: roll, pitch and yaw. Inertial sensors have a very wide field of application, which reflects on the accuracy of the measurements and on the cost of the hardware. A deeper discussion on inertial sensors and IMUs is carried out in Section 2.4.1.

2.2.2 Position-fixed Techniques

Absolute positioning methods are based on the measurement of the distances or angles from multiple stationary objects whose position is exactly known, thanks to which the position of the moving object can be estimated using trilateration or triangulation techniques.

Electronic Compass

Electronic compasses use only magnetometers to sense the local magnetic field and they can provide heading measurements relative to Earth's magnetic north. The main

disadvantage of electronic compasses is that the local magnetic field is easily distorted by nearby power lines or metal structures, making their heading measurements particularly unreliable for land vehicles.

Global Navigation Satellite Systems (GNSS)

GNSS use satellites to estimate the position of moving objects on the surface. Satellites broadcast coded radio frequency signals that can be divided into three main components: the carrier wave, a ranging code and the effective navigation data. The carrier wave is a sinusoidal radio frequency signal in the L-band. The ranging code allows the receiver to determine the travel-time of the signal to estimate distance. The navigation data includes the satellites position and velocity. The receiver can then use the information contained in GNSS signals to estimate its own position by using trilateration techniques. Accuracy for low-cost commercial receiver is a few meters, while for industrial-grade sensors can arrive to centimetres.

Celestial Navigation

Celestial navigation is the oldest position-fixed technique known used for navigation. It is based on the usage of angular measurements between the visible horizon and a celestial body to determine the position of the moving object. The instrument used since ancient times for celestial navigation is the sextant; nowadays, this method for navigation is not used any more.

2.3 Coordinate Systems

All the navigation techniques introduced, can be effectively used in combination (for example in fusion techniques), if the measurements acquisitions from different sources follow common rules regarding the reference frames used. In the following, the main coordinate systems used for navigation purposes are introduced.

In general, the position of a point in space can be described with a vector originating from the origin of a reference frame; in this way, the position is always considered as relative to the specific frame chosen. Inertial reference frames have null net sum of

the forces acting on them, resulting in zero velocity or a constant velocity motion. As convention, inertial sensors provide measurements relative to inertial frames, and every non-inertial reference frame can be exposed to forces causing accelerations relative to the inertial frame; in this way, the specific force can be measured.

In the following, a brief introduction of the notation used is presented before the definition of the coordinates systems relevant for navigation purposes. The main coordinate systems considered are listed below, with reference to [24]:

- *ECI* - Earth Centred Inertial
- *ECEF* - Earth Centred Earth Fixed
- *WGS* - World Geodetic System
- *LTP* - Local Tangent Plane
- *Body fixed*

2.3.1 Vector Notation

In this work as a convention, vectors are expressed with bold lowercase letters and vector components are expressed with non-bold lowercase letters. Both are expressed with a superscript that refers to the reference frame in which the vector or the component is represented. As example, a three-dimensional vector $\mathbf{r^a}$ represented from the origin in an arbitrary reference frame $\mathbf{R_a}$ is expressed as:

$$\mathbf{r^a} = [x^a, y^a, z^a]^T \tag{2.1}$$

2.3.2 Coordinate Transformation

For navigation-related computations, coordinate vectors transformations between different reference frames are needed. A transformation of coordinates is a particular case of linear mapping, because rigid transformations induce linear operators applied to coordinate vectors. Transformations can either preserve orientation (translations) or

change orientation of the frame (rotations). In the case of the reference frames considered in the analysis, only rotations are considered and thus coordinates transformations can be studied through a particular case of transformation matrix: the rotation matrix.

Rotation matrices are expressed with bold uppercase letters, completed with a subscript representing the reference frame to whom the original vector belonged before the transformation, and a superscript that expresses the frame in which the vector will be represented after the transformation. As example, the rotation matrix transforming vectors represented in the reference frame $\mathbf{R_a}$ into an arbitrary frame $\mathbf{R_b}$ is expressed as:

$$\mathbf{R}_{\mathbf{a}}^{\mathbf{b}} \tag{2.2}$$

The transformation of the vector $\mathbf{r^a}$ belonging to the frame $\mathbf{R_a}$ into the frame $\mathbf{R_b}$ is given by the vector $\mathbf{r^b}$, expressed by the following relation:

$$\mathbf{r^b} = \mathbf{R}_{\mathbf{a}}^{\mathbf{b}} \cdot \mathbf{r^a} \tag{2.3}$$

If both the reference frames considered as orthonormal, which is always true in our cases of interest, the rotation matrix is also orthonormal. The representation matrix of one orthonormal frame into another orthonormal frame, is proper (it holds that $det(\mathbf{R}_{\mathbf{a}}^{\mathbf{b}}) = 1$) if and only if both the frames are right-handed (or left-handed). In this case, the inverse of the rotation matrix is also its transpose. As example, using the matrix notation introduced so far, it holds the following:

$$\mathbf{R}_{\mathbf{a}}^{\mathbf{b}} = (\mathbf{R}_{\mathbf{a}}^{\mathbf{b}})^{-1} = (\mathbf{R}_{\mathbf{a}}^{\mathbf{b}})^{T} \tag{2.4}$$

The inverse transformation, used to rotate the vector represented in frame $\mathbf{R_b}$ back to the frame $\mathbf{R_a}$, is given by the vector $\mathbf{r^a}$, expressed equivalently by the following relations:

$$\mathbf{r^a} = \mathbf{R}_{\mathbf{b}}^{\mathbf{a}} \cdot \mathbf{r^b} = (\mathbf{R}_{\mathbf{a}}^{\mathbf{b}})^{T} \cdot \mathbf{r^b} \tag{2.5}$$

Table 2.1: *ECI Reference Frame.*

Origin	Fixed at Earth's mass center.
X-axis	Points through the equatorial plane towards the vernal equinox.
Y-axis	Points through the equatorial plane completing the right-hand triplet.
Z-axis	Points along Earth's rotational axis.

2.3.3 Earth-Centered Inertial (ECI) Frame

The Earth-centered inertial (ECI) frame is a global reference frame that has its origin at the centre of Earth. This reference frame does not rotate with the Earth, and serves as an inertial reference frame for objects in the near-Earth environment, such as satellites. Objects' positions in the ECI frame can be described with both Cartesian and spherical coordinates. Using spherical coordinates, a position can be described in terms of right ascension and declination (ref. to Table 2.1): the right ascension is the angular distance from the vernal equinox going eastward along the equatorial plane, while the declination is the angle between the equatorial plane and Earth's rotational axis, positive on the north hemisphere.

2.3.4 Earth-Centered, Earth-Fixed (ECEF) Frame

The Earth-centered, Earth-fixed (ECEF) frame is also a global reference frame with its origin at the centre of Earth, and three orthogonal axes fixed to the Earth. The Z-axis points through the North Pole, the X-axis points through the intersection of the IERS Reference Meridian (IRM) and the equator, and the Y-axis completes the right-handed frame, as reported in Table 2.2. This reference frame rotates with Earth at an angular rate of approximately 15 °/hour (360 ° over 24 hours). Within the ECEF reference frame, there are two primary coordinate systems used when describing the position of an object: Cartesian and Geodetic. They are usually referred to respectively as ECEF rectangular (ECEF-r), and ECEF geodetic (ECEF-g) coordinate systems. ECEF-r is the more direct way to express coordinates because leverages the (X, Y, Z) axis definition introduced and can be easily related to the Local Tangent Plane (LTP) coordinates commonly used for their ease of interpretation. On the other hand, the ECEF-g is more convenient to express positions in Earth's immediate surroundings, so it has been chosen as standard reference for satellites positioning systems.

Table 2.2: *ECEF Reference Frame.*

Origin	Fixed at Earth's mass center.
X-axis	Points through the equatorial plane at the prime meridian.
Y-axis	Points through the equatorial plane completing the right-hand triplet.
Z-axis	Points along Earth's rotational axis.

Table 2.3: *WGS Coordinate System parameters.*

a	Semi-major Axis	6378137.0 m
1/f	Flattening Factor	298.257223563 –
ω_E	Nominal Mean Angular Velocity	$7292115.0 \times 10^{-11}$ rad/s
μ	Geocentric Gravitational Constant	3986004.418×10^8 m^3/s^2

2.3.5 Geodetic Coordinate System

The Geodetic coordinate system is related to the definition of the ECEF-g frame and it is a set of polar coordinates, that accounts for the first-order effect of Earth being an ellipsoid rather than a sphere. The ECEF-g position of an object is described in terms latitude (ϕ), longitude (λ), and altitude or elevation (h). The longitude of a point is the angle between the prime meridian and another meridian passing through that point. The latitude is the angle between the equatorial plane and a latitude parallel to it, that passes through the point of interest. The World Geodetic System (WGS84) is the standard Geodetic coordinate system used by the Global Positioning System (GPS) for navigation. By definition, it uses the Reference Ellipsoid model of Earth's surface, defined by the parameters reported in Table 2.3.

2.3.6 Local Tangent Plane (LTP) Coordinate System

Local Tangent Plane (LTP) coordinates are the most relatable type of coordinates to express the position of a moving object on a map on Earth's surface. Among LTP coordinates, the focus of will be on the East-North-Up (ENU) frame, in which the results of this work are expressed.

The ENU frame is a local reference frame defined by its ECEF coordinates as shown in Figure 2.2. As a common convention in many applications, this frame is fixed to the vehicle or the moving platform, and moves with the body frame. The ENU frame is defined such that the East-North directions span a plane tangent to the Earth's surface

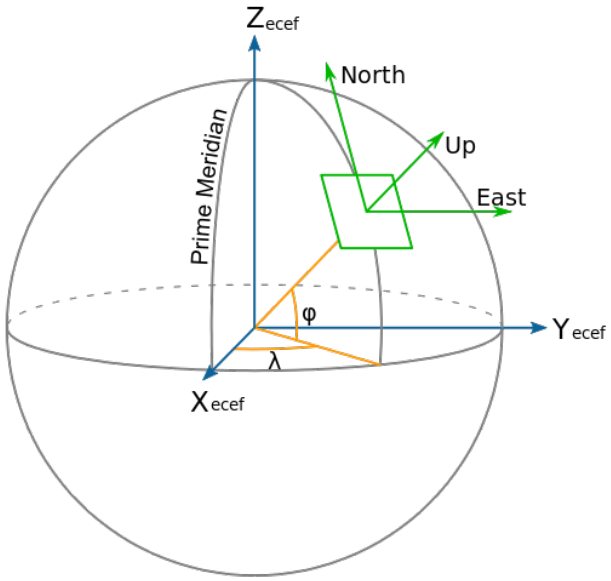


Figure 2.2: *ENU reference frame in the ECEF coordinate system.*

Table 2.4: *Local ENU Reference Frame.*

Origin	Arbitrary point on Earth's surface.
X-axis	Points East.
Y-axis	Points North.
Z-axis	Points Up.

at its present position, assuming a WGS84 ellipsoid model of the Earth, while the Up direction points away from Earth, perpendicular to the East-North plane. The defined reference is called Local ENU reference frame (ref. to Table 2.4).

Similar to the ENU frame, it can be defined also the North-East-Down (NED) frame, that can be placed locally on a vehicle or a platform, and moves around with it. The NED differs from the ENU in the direction of the three orthogonal axes, being defined by the X-axis pointing North, the Z-axis pointing towards the interior of the Earth and the Y-axis, completing the right-handed frame, pointing East.

2.3.7 Body-fixed Coordinate System

In most applications, the position, velocity, and orientation of a system are found using sensors mounted on its body. The platform can have its own reference frame known as body frame, also called vehicle frame in the following. This type of reference frame consists of an origin that is typically placed at the platform's centre of gravity, and

Table 2.5: *Body fixed ENU Reference Frame.*

Origin	Fixed at the vehicle's center of gravity.
X-axis	Points in the Longitudinal direction.
Y-axis	Points in the Lateral direction.
Z-axis	Points Up.

three orthogonal axes that compose a right-handed system. These axes are usually configured to the body in such a way that the X-axis is pointing forward, the Y-axis is pointing to the transversal direction, and the Z-axis is pointing up, composing the so-called Body Fixed ENU frame (ref. to Table2.5), for the similarities with the previously defined reference. Usually, the orientation of this frame can be described as relative to the respective Local ENU frame, by using RPY angles or quaternions.

2.4 Sensors

Sensors are the fundamental building elements of a perception pipeline for autonomous navigation purposes, hence cover a critical role for autonomous driving technologies development, both from a technical and an economical point of view. A sensor is a mechatronic system which can detect specific properties of the physical environment to which is exposed, and can directly process the information or forward it for processing. Sensors commonly consist of two components, a sensitive element, and a transducer. The sensitive element directly or indirectly interacts with the external input, while the transducer translates this input into an output signal that can be read by a data acquisition system.

The theory on the construction of sensors is a very wide field of research itself, and the extensive description of working principles, limitations and performances, is out of the scopes of this thesis work. In this section, the main sensors used in the ADS proposed by Squadra Corse PoliTo Driverless are presented, with a specific focus on the ones used by the Localization pipeline proposed, namely a Global Navigation Satellite System (GNSS) receiver and an Inertial Navigation System (INS) which embeds an Inertial Measurement Unit (IMU).

The common definitions of the sensors' characteristics that can be found in this

thesis work are reported below. For extensive information on sensors' characteristics, the reader is invited to refer to dedicated literature works, such as [25] and [26].

Sensor Error

The absolute sensor error is the difference between the sensor output and the true value. The relative sensor error is the difference divided by the true value.

Noise

Sensor noise is the undesired fluctuation in the sensor output signal when the true value is kept constant. The variance of the noise is an important parameter in sensor characteristics. White noise is a random signal where all frequencies contain equal intensity.

Drift

Sensor drift is an unwanted change in sensor output while the true value is kept constant. In navigation, drift is commonly experienced by sensors leveraging only dead-reckoning techniques.

Resolution

The resolution is the minimal change of the desired physical parameter that the sensor can detect.

2.4.1 Inertial Measurement Unit (IMU)

The first category of sensors to be analysed are the Inertial measurement units (IMUs), which are by far the most common type of inertial sensors. IMUs are used in a wide variety of applications for their good performance, low weight and low cost. Depending on the specific application, different IMUs are nowadays available on the base of the needed resolution, noise tolerance or error admission. In general, all IMUs measure and report the body's specific forces, angular rates, and orientation, using a combination of accelerometers, gyroscopes, and eventually magnetometers. Different technologies for those components are present as shown in Figure 2.3: in the following, the components are presented in a descriptive fashion, with particular focus on the final application in analysis.

Sensor	Gyroscope			Accelerometer		
Physical Effect Used[a]	Conservation of angular momentum	Coriolis effect	Sagnac effect	Gyroscopic precession	Electro-magnetic force	Strain under load
Sensor Implementation Methods	Angular displacement	Vibration	Ring laser	Angular displacement	Drag cup	Piezo-electric
	Torque rebalance	Rotation	Fiber optic	Torque rebalance	Electro-magnetic	Piezo-resistive

[a] All accelerometers use a proof mass. The physical effect is the manner in which acceleration of the proof mass is sensed.

Figure 2.3: *Basic division between Inertial Sensors technologies.*

Accelerometers

Through accelerometers it is possible to measure the relative acceleration with respect to an inertial reference frame. Internal elements of an accelerometer can be modelled as a damped mass on a spring system: when the sensor is exposed to an acceleration, the mass displacement is measured by transducers with capacitive or piezoresistive effects. A capacitive accelerometer exploits the mass as a capacitor, whose capacitance changes with displacement, while a piezoresistive accelerometer leverages the change in material's electrical resistivity with deformation to assess the external acceleration.

Gyroscopes

Gyroscopes are devices used to measure angular rates relative to an inertial reference frame. Early gyroscopes used a spinning mass supported by gimbals: the conservation of angular momentum keeps the spinning mass levelled when the support is tilted, and the angular difference can be measured. Newer types of gyroscopes are the optical and the vibrating gyroscopes, with the latter one representing the main technology adopted by MEMS IMUs (Microelectromechanical Systems IMUs).

Optical gyroscopes use the Sagnac effect: if two pulses of light are sent in opposite directions around a stationary circular loop, their travelled inertial distance is the same. On the contrary, when the loop is rotating, the two pulses travel a different distance, with the one travelling in the same direction of the rotation, covering a smaller inertial distance and thus arriving later with respect to the other. Using interferometry, the

differential phase shift can be measured and translated into angular rate.

Vibrating gyroscopes consist in a vibrating mass mounted on a spring and their working principle leverages the Coriolis effect. When the mass is oscillating in the x-axis and a rotation about the z-axis occurs, an acceleration in the y-axis is measured: this acceleration is the Coriolis acceleration and can be computed as the double product between the speed of oscillation of the mass and the angular rate of rotation. By knowing the first and by measuring the second through the force induced by the Coriolis effect, it is possible to measure the corresponding angular rate.

Magnetometers

Magnetometers are devices used to measure the local magnetic field by using the Hall effect, usually consisting in thin sheet of semi-conducting materials. In a magnetic-free environment, the electrons in the thin sheet are evenly distributed and the potential is null; on the contrary, when a magnetic field is present, the electrons distribute unevenly in the sheet, inducing a potential difference. The potential can be easily measured and then translated into magnetic flux density as a measure of the intensity of magnetic field.

Information on magnetic field is useful to derive the heading of the vehicle by comparing the intensity of the measured field with the local intensity of Earth's magnetic field. Unfortunately, magnetometers have very poor application on mobile robots and vehicles, because of the presence of nearby metallic parts, electrical cabling or even worse, power electronics systems.

2.4.2 Inertial Navigation System (INS)

Inertial navigation Systems (INSs) represent the most common application of IMUs in the field of navigation, in which the terms IMU, inertial instrument, inertial guidance system or integrated navigation system, are often un-correctly used to refer to an INS.

From its definition, an inertial navigation system is a navigation device that uses a computing platform, multiple motion (accelerometers) and rotation sensors (gyroscopes) to continuously calculate by dead-reckoning the position, the orientation, and the velocity (direction and speed of movement) of a moving object without the need of any external reference. Often the inertial sensors are supplemented by a barometric

altimeter and occasionally by magnetic sensors (magnetometers) and/or speed measuring devices (ground truth sensors or odometers). The difference with the class of the IMUs, lays in the presence of an embedded computer which uses the measurements from the inertial sensors and the supplementary ones, in order to assess the vehicle's states. An example of the signal processing performed by an INS is presented in Figure 2.4.

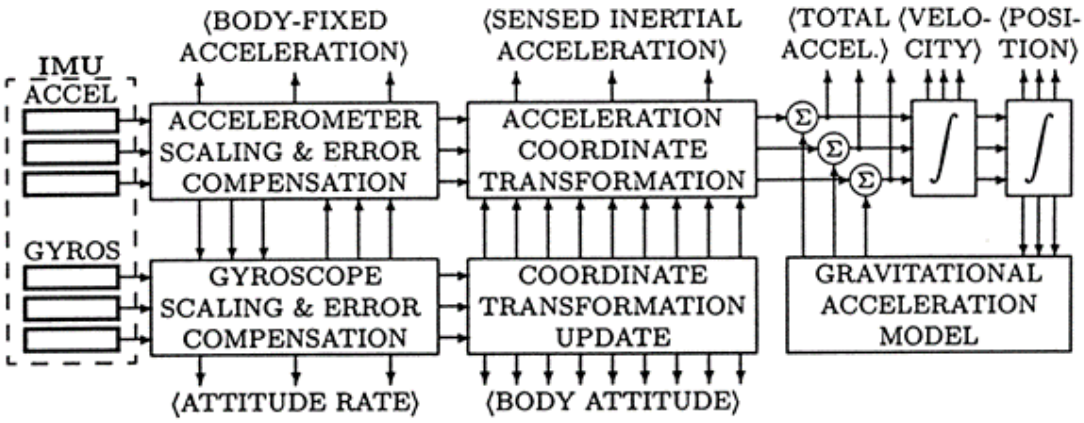


Figure 2.4: *Essential signal processing for strapdown INS.*

INSs are used on mobile robots and on both land and air vehicles, because of their effectiveness in inertial measurements and enhanced accuracy with respect to standalone IMUs. Usually INSs, exploit dead-reckoning techniques to estimate positions starting from the knowledge of fixed points, evaluated by means of absolute positioning devices. To properly operate, an INS should be initialized with an external reference fed to the embedded computing platform, which exploits the information to align the inertial sensors, and begin the navigation phase. The result of having an on-board computer which evaluates real-time positions, is the possibility to increase the accuracy of the sensor's readings and to avoid drift of the position measurements. In more advanced systems, such as the one considered in this work, this is achieved by leveraging sensor fusion algorithms, which are discussed in the following section.

2.4.3 Global Navigation Satellite System (GNSS) Receiver

Conversely to the IMUs and INSs, the satellite navigation systems belong to the category of the absolute positioning devices, because leverage the knowledge of the coordinate systems used in Geodesy to assess objects' position on Earth.

As its name suggests, satellite navigation uses satellites to provide geo-spatial positioning for electronic receivers to determine their own location (longitude, latitude, and altitude/elevation) with high level of precision. The signals also guarantee the electronic receiver to calculate accurately the current local time, which allows time synchronisation; these uses are collectively known as Positioning, Navigation and Timing (PNT). Furthermore, a satellite navigation system with global coverage may be termed as global navigation satellite system (GNSS). Operational systems with global coverage today, are United States' Global Positioning System (GPS), Russia's Global Navigation Satellite System (GLONASS), China's BeiDou Navigation Satellite System (BDS) and the European Union's Galileo. Other countries such as India, Japan and France are also developing their own GNSS network.

Global coverage for each system is generally achieved by a satellite constellation of 20 to 30 medium Earth orbit (MEO) satellites spread between several orbital planes. The actual system layout may vary, but in general they use orbital inclinations greater than 50 degrees and orbital periods of roughly twelve hours (at an altitude of about 20,000 kilometres). The satellites constellations ensure that at least 4 satellites are visible at any place on Earth at any given time, so that trilateration can be used to precisely determine the position of the receiver on Earth's surface with sub-meter accuracy.

The performance of GNSS receivers is assessed using the following four criteria:

1. *Accuracy*: the difference between the receiver's measured and real position, speed or time.

2. *Integrity*: the system's capacity to provide a threshold of confidence and, in the event of an anomaly in the positioning data, an alarm.

3. *Continuity*: the system's ability to function without interruption.

4. *Availability*: the percentage of time a signal fulfils the above accuracy, integrity and continuity criteria.

The usage of a GNSS receiver is a key feature for navigation since the introduction of the first satellite navigation systems in the 90s, which allowed for easier, more accurate, and real-time positioning on Earth. From being exclusively a military technology, satellite navigation is nowadays mainly used for civilian applications, and its world-wide availability is the main reason for which it is considered as a key technology for autonomous driving. However, positioning systems relying only on GNSS receivers do not represent a reliable and robust solution to the localization problem, because of their poor resolution, and the need for a continuous satellite signal reception to operate. In common urban road conditions such as tunnels or urban canyons, signal reception is poor due to shadowing or multipath effects, making pure GNSS navigation ineffective.

2.4.4 Other Sensors

Stereo Cameras

Stereo cameras represent one of the most important sensors available for autonomous navigation, because give the autonomous system the possibility to implement computer vision. Computer-vision tasks include methods for acquiring, processing, analysing, and understanding digital images, in order to extract data from the real world, and produce numerical or symbolic information. Computer-vision routines in autonomous driving leverage machine learning techniques for scene reconstruction, event detection, object recognition and indexing, 3-D pose and motion estimation.

From the hardware point of view, a stereo camera is a type of camera with two or more lenses with a separate image sensor or film frame for each lens. This allows the camera to simulate human binocular vision, and therefore gives it the ability to capture three-dimensional images, a process known as stereo photography. Stereo cameras produce stereo-views and 3-D pictures, which can be used for range imaging other than detection. The distance between the lenses in a typical stereo camera, i.e. the intra-axial distance, is comparable to the intra-ocular distance in human eyes (about 6 centimetres), reproducing quite accurately the 3-D view capability of human vision.

LiDARs

Another fundamental sensor to enable robust autonomous driving, is the emerging category of lidars. LiDAR is the acronym for Light Detection and Ranging or more precisely Laser Imaging Detection and Ranging. The term was originated by the merging of the words light and radar, because of the similarities with the "radio detection and ranging" systems which are commonly used in marine applications.

Lidar is a method for measuring distances (ranging) by illuminating the target with laser light and measuring the time the reflection of the light takes to return to the sensor. Differences in laser waveforms return times and wavelengths are used to make digital 3-D representations of the immediate surroundings of the vehicle. The data acquired by lidar sensors is in the form of a point cloud, which can be processed by machine learning routines to extract information on the environment and perform object recognition and shaping as well as obstacle detection and avoidance.

2.5 Sensor Fusion

From the analysis of the sensors carried out so far, self-contained methods and absolute measurements methods used for navigation purposes, have different advantages and drawbacks in different operating conditions. While dead-reckoning is generally useful for short-distance assessment of positions and vehicle's states, it suffers from error accumulation over time due to numerical integration. On the other hand, absolute measurements are good on a long-distance perspective for the assessment of trajectories and paths, but lack of resolution and strongly depend on environmental conditions. The main references used to carry out the discussion on sensor fusion are [27] [28] [29].

To overcome the issues of the two methods, while fully exploiting the advantages given by the sensors, many sensor fusion techniques have been studied and introduced over the years to deliver the optimal estimate of positions and states of vehicles or mobile robots. Sensor fusion is the merging of numerical data from multiple sensors to achieve an information gain relative to using each sensor individually. Sensory data are combined such that the resulting measurement has less uncertainty than the one associated to the single readings alone. Here, the expression "uncertainty reduction"

refers to a more accurate, more complete, or more dependable result, which is less prone to suffer measurement noise distortion and limitations in the observable range. Furthermore, the process of information fusion can be direct, when the fusion of a set of heterogeneous or homogeneous sensors is performed, or indirect, when the fusion technique leverages a priori knowledge of the environment, mathematical models of the physical process, or history values of sensor data.

2.5.1 State Estimation and Kalman Filtering

Multi-sensor data fusion covers a fundamental role in the design of robust perception pipelines for autonomous driving, which in many scenarios are required to deliver consistent and reliable data even in case of sensor's failures or excessive fluctuations of their measurements. The most common techniques used for sensor fusion derive from the concept of state estimation, exploiting a probabilistic approach in the usage of statistical inference from observations by different sources [28]. In the field of localization, the estimation problem is solved by adopting filtering or smoothing techniques (depending on the time-horizon used), starting from raw sensor measurements. If redundant data coming from different sources representing the same physical property are processed by the estimator (or filter), then the state estimator is also a sensor fusion technique, increasing the certainty of the state.

The most common filtering technique is the Kalman Filter which has been introduced by Rudolf E. Kalman in 1960. The Kalman filter is an optimal recursive data processing algorithm [27] that uses a series of measurements observed over time, corrupted by statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone. This is possible by using some basic knowledge of the system and the measurement device, the statistical description of the system noise, measurement error, and uncertainty in the dynamics, as well as any available information about the initial conditions of the variables of interest. The Kalman filter algorithm is recursive and leverages the estimation in real-time of the joint probability distribution over the variables for each time-frame to deliver the optimal estimate.

The basic principle of working of Kalman filters, is presented in Figure 2.5. The

algorithm runs a first prediction step, based on the prior knowledge of the state to be estimated, and then an update or prediction step, in which the predicted state is corrected based on the innovation, introduced by the current measurement acquired. The fundamental operation performed is the estimation of the probability distribution based on the assumption that the system's state transition is linear and only involves input Gaussian random variables (GRV). For this reason, in estimation theory Kalman Filters are referred to as a special class of Bayes Filters, known as Gaussian Filters.

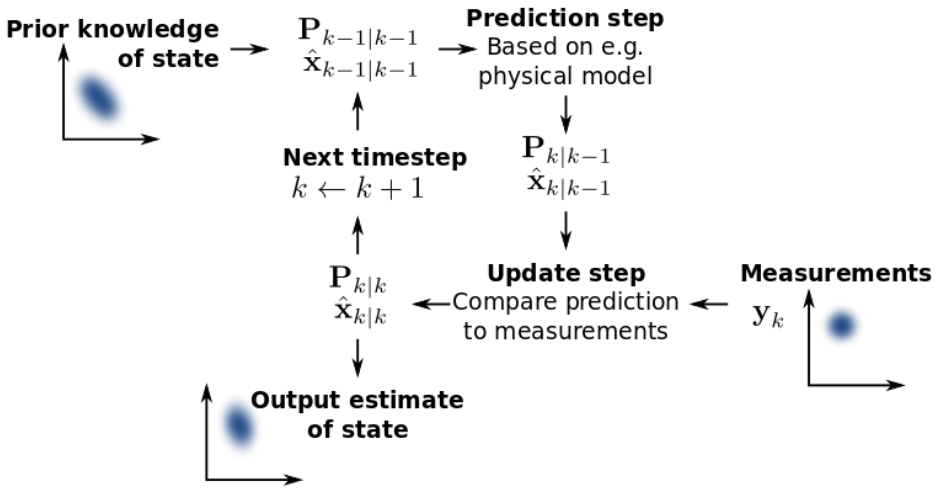


Figure 2.5: *Basics steps performed by Kalman Filtering techniques.*

The Kalman filter addresses the general problem of trying to estimate the state of a discrete-time controlled process, that is governed by the linear stochastic difference equation:

$$x_k = Ax_{k-1} + Bu_k + \varepsilon_k \tag{2.6}$$

with discrete-time measurement acquired at the same time-step that is:

$$z_k = Hx_k + \delta_k \tag{2.7}$$

The Gaussian random variables (GRVs) ε_k and δ_k represent the process and measurement noise, respectively. They are assumed to be independent of each other, white, and with Gaussian probability distribution, whose covariance is expressed by the ma-

trices R_k and Q_k:

$$\begin{aligned} P(\varepsilon_k) &\sim N(0, R_k) \\ P(\delta_k) &\sim N(0, Q_k) \end{aligned} \tag{2.8}$$

The output of the filter is a sequence of state estimates $\hat{x}_{k|k}$ and covariance matrices of the states estimated $P_{k|k}$, recursively obtained from the given initial conditions, namely $x_{0|0}$ and $P_{0|0}$.

The Kalman filter algorithm has proven to be a very powerful tool for state estimation for a long period of time, because of its generally good performance and ease of implementation. Nevertheless, the hypothesis on which it is based, represented a huge limitation for its usage in a variety of applications. For this reason, many other filtering techniques have been developed to widen the field of application of state estimation and filtering techniques. In the following, the most important direct derivations of the Kalman filter in the framework of non-linear systems are presented. The other classes of filters used for state estimation not based on the Kalman filter's structure are not part of the discussion, and thus will be just introduced in the dedicated subsection.

2.5.2 Non-linear Kalman Filtering

This thesis work mainly focuses on the class of non-linear Kalman filtering techniques, which extend the usage of Kalman filters to non-linear systems. Estimation of non-linear systems is extremely important in the implementation of Kalman theory, because almost all practical systems involve non-linearities of some kind. As it will be discussed in the next section dedicated to modelling (2.6), the mathematical model chosen to represent the system under study, is a simplified one, but still is a non-linear one. In a large portion of the field of application of state estimation algorithms, the systems considered are non-linear, thus the extension of the Kalman theory to this class of systems was straightforward.

Starting from the Kalman approach, the optimal Bayesian solution to the problem requires the propagation of the joint probability distribution, a very general approach which incorporates multi-modalities, asymmetries, and discontinuities. For linear systems this operation is quite simple, while in non-linear ones the form of the associated

probability density function (pdf) is not restricted and involves a non-finite number of parameters to be described. Therefore, any practical estimation techniques should involve some approximation to cope with non-linearities. Many types of approximations have been developed, but unfortunately most of them are computationally too demanding or require specific assumptions on the form of the process and observation models, that are not satisfied in practice by any physical system. For this reason, Kalman filters remain the most widely used estimation algorithm and the simple usage of the first two moments of the state (mean and covariance) in its update rule, offers a number of important practical benefits, while being a relatively simple state representation [30].

In general, for a generic discrete-time non-linear system dealing with noisy measurements, the presented framework is represented by the following equations:

$$x_k = g(u_k, x_{k-1}) + \varepsilon_k \tag{2.9}$$

$$z_k = h(x_k) + \delta_k \tag{2.10}$$

where x_k is the state to be estimated, u_k is the control input, z_k is the measured output, ε_k and δ_k are respectively the process and measurements noises. Moreover, an initial state estimate $x_{0|0}$ and an initial estimation error covariance matrix $P_{0|0}$ are given, and the initial estimate is assumed to be uncorrelated with both ε_k and δ_k. Even under the assumption of Gaussian noises affecting the system, the filtering problem rises with complexity when dealing with non-linearities, because non-linear transformations of Gaussian variables, such as g and h in Equations (2.9) and (2.10), do not return Gaussian variables as a result.

The most common application of Kalman theory to non-linear systems is represented by the well-known Extended Kalman filter (EKF). The fundamental operation performed by every EKF algorithm is the linearisation of the non-linear functions (g, h) in Equations (2.9) and (2.10) with a GRV, i.e. the state distribution, which is then propagated through a first order linearisation of the non-linear system via Taylor expansion. Hence, the main assumption that the initial state and measurement noises are Gaussian and uncorrelated with each other still holds. An extensive description of

the EKF algorithms used in this work can be found in Section 3.4.2, when the filters' setup and tuning is presented and analysed.

The central and vital operation performed by the EKF is again the propagation of GRV through the system dynamics, this time considering a first-order linearisation before it. The problem with this kind of propagation through non-linear system's dynamics, is that the operation can introduce large errors in the true posterior mean and covariance of the transformed GRV, which may lead to sub-optimal performance and sometimes cause the divergence of the filter. Those are in fact the main drawbacks of the EKF approach, and as it will be analysed in Section 3.4.2, this is also the case in localization, where GNSS measurements introduce high non-linearities of the system's dynamics, and thus cause the EKF to sub-optimal performances.

Although the EKF maintains the elegant and computationally efficient recursive update of the Kalman filter, its limitations pushed the investigation of alternative methods to the propagation of the first-order linearisation through the non-linear dynamics. One way the problem has been addressed, is by using a deterministic sampling approach, as proposed by [30]: to overcome the EKF limitations, the unscented transformation (UT) was developed as a method to propagate mean and covariance information though the non-linear transformations. The method has proven to be more accurate, easier to implement, and uses the same order of calculations as a linearisation of the same system.

The result is the so called Unscented Kalman filter (UKF). In the UKF, the state distribution is again approximated by a GRV, but is now represented using a minimal set of carefully chosen sample points, called sigma-points. These sample points completely capture the true mean and covariance of the GRV, and when propagated through the true non-linear system, the filter is capable to capture the posterior mean and covariance accurately to the 3rd order Taylor series expansion for any non-linearity. To put it into perspective, the EKF only achieves first-order accuracy and remarkably, the computational complexity of the UKF remain in the same order as that of the EKF. Also, for what concerns the UKF, an extensive discussion on the design and the implementation of the algorithm used in this work is present in Section 3.4.3.

2.5.3 Other Filtering Techniques

The dual of the Kalman Filter (KF) family is represented by the category of the Information Filters (IF), which is applied to non-linear systems as Extended Information Filter (EIF) and Sparse Extended Information Filter (SEIF) [29].

Information Filters exploit the canonical parametrization of GRVs, thus allow for filtering in the so-called information form, which has a slower prediction phase, while having a more efficient correction one if compared to the Kalman Filter. Both the families of filters (KF and IF) retain the same expressiveness, and thus have comparable results both with linear and non-linear systems. The choice of a KF over an IF, is dictated by the easier problem statement using the standard mean and covariance form of the GRVs considered, proposed by the KF.

All the filters considered so far, belong to the macro-category of Gaussian Filters, because their working principles are based on the Gaussian distribution of the errors. Nevertheless, Gaussian Filters represent just one of the three paradigms of state estimation applied to pure localization or combined localization and mapping techniques, as reported by [31]. Most of this theory in fact has been developed to find reliable algorithms for Simultaneous Localization and Mapping (SLAM). As it will be clear in the early stages of Chapter 3, SLAM approach was not feasible for the system under study in this thesis work, but the same kind of filters used in SLAM can be applied to estimation techniques for robust localization tasks. Those techniques can be grouped under the names of Particle Filters and Graph-based methods, which represent together with the Gaussian methods, the three main paradigms of state estimation for localization and SLAM.

Particle Filters and Graph-based methods have proven to guarantee a more stable solution to the state estimation problem with respect to Gaussian techniques, because of the lower level of approximation introduced in measurement noise. Nevertheless, the computational complexity rises with the adoption of non-parametric noise descriptions, especially for non-linear systems [32]. Thus, considering the requirements of this thesis work and the expected outcomes, only Gaussian filters have been considered, because of the best trade-off reachable between computation effort required, robustness, and estimates' accuracy.

2.6 Modelling

As conclusion of this chapter dealing with the theoretical background, this section is dedicated to the modelling approach for both the considered vehicles, along with a brief presentation of sensor modelling. Dealing with state estimation techniques, the approach to localization requires a complete knowledge of the system under control from a mathematical point of view. Filtering techniques must leverage also the a-priori knowledge of the system's model to improve the quality of the state estimation, so it is fundamental to derive a mathematical representation of the system, consistent with the application.

In this section, the modelling of the two retained vehicles is presented together with the presentation of the main parameters of the vehicles considered. Modelling covers a fundamental role in robotics and dynamical system design and control: this discussion aims to present only an overview of the model used in the design, but the complete description of the model derivation can be found in the set of the references used to write this chapter.

Modelling is the mathematical description of a system so that the system can be analysed, optimized, and used for information processing (for example state estimation). A precise and complex model might be closer to reflect the physical reality of the system but will require more computational power to be processed. In the following, the rather simple, but powerful model used in the design of the localization pipeline is presented. The main advantage of the model used is the good abstraction level reached, useful to define a closed set of state variables, while being very efficient from the computational point of view.

2.6.1 Vehicle Modelling

The proposed localization algorithm is tested and validated on two different vehicles. The first one is the electric racing vehicle used as a prototype for the deployment of the final pipeline, while the second is a commercial passenger car (compact SUV class) used for validation. The vehicles are very different one from the other, but the comparison holds when dealing with the testing of the perception capabilities of the ADS. The

testing of the proposed method has been possible on two different vehicles, thanks to the modular hardware architecture chosen, designed to have a minimum impact in terms of packaging and space needed for sensors and embedded computer placement.

For both the vehicles considered, the longitudinal and lateral dynamics can modelled by means of a single-track, linear, three degrees-of-freedom (DoF) model. The model, also referred to in literature as "Bicycle Model" [33], belongs to the class of the dynamic models, and differs from the standard kinematic model of reference for the elimination of the assumption that the velocity vector of each wheel is parallel to the wheel's symmetry plane. The model is used in the ADS of which Localization is part, for the motion control strategies, as described in [34]. For this reason, it has been taken as reference for the design of the sensing and processing phases too, proving to work smoothly also in those fields.

This simplified model has proven to be effective among the different mathematical formulations which are present in literature [35] [36] [33]. The relevant states of the vehicle model are represented in Figure 2.6. The considered formulation features a rigid two-axle vehicle model, and accounts for the linear motion in the xy-plane (namely longitudinal and lateral motions) and the rotation about the z-axis normal to the plane (yaw motion). The two front wheels and the two rear wheels are represented as a single central wheel on the axle: among them, the front wheel is the only steerable one, as the front axle of the nominal vehicle.

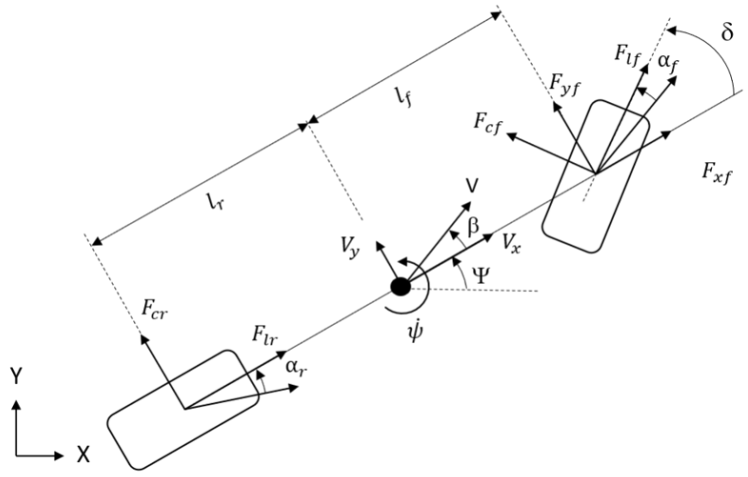


Figure 2.6: *Single track linear 3-DoF vehicle model.*

In Figure 2.6, two different reference frames are considered: the inertial reference frame (XY) and the vehicle reference frame (xy). The two frames considered by the model are chosen consistently with sensors' frames, so that only misalignment should be accounted and no transformations of the frames must be performed. The considered DoFs are the lateral and longitudinal positions of the vehicle, and the yaw angle. A canonical state feedback formalism is used for the modelling definition, as in [5] and [37]. In the state-space model of Equation 2.11, the lower scripts $(\bullet)_f$ and $(\bullet)_r$ denote the variable subscripts at the front and rear wheels, respectively. The state space model is obtained considering a first order transfer function, with time constant $\tau = 0.5\,\mathrm{s}$ for the longitudinal dynamics, and the Euler equation for the lateral and the yaw motion:

$$\begin{bmatrix} \ddot{V}_x \\ \dot{V}_x \\ V_y \\ \dot{V}_y \\ \dot{\psi} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\tau} & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\frac{2C_{\alpha_f}+2C_{\alpha_r}}{mV_x} & 0 & -V_x - \frac{2C_{\alpha_f}l_f - 2C_{\alpha_r}l_r}{mV_x} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\frac{2C_{\alpha_f}l_f - 2C_{\alpha_r}l_r}{I_z V_x} & 0 & -\frac{2C_{\alpha_f}l_f^2 + 2C_{\alpha_r}l_r^2}{I_z V_x} \end{bmatrix} \begin{bmatrix} \dot{V}_x \\ V_x \\ y \\ V_y \\ \psi \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} \frac{1}{\tau} & 0 \\ 0 & 0 \\ 0 & \frac{2C_{\alpha_f}}{m} \\ 0 & \frac{2l_f C_{\alpha_f}}{I_z} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{V}_x \\ \delta \end{bmatrix} \tag{2.11}$$

where x and y are the positions along the x-axis and y-axis, respectively, V_x and V_y are the longitudinal and the lateral vehicle velocities, respectively, ψ is the yaw angle, and δ is the front wheel steering angle.

The equations of motion for the considered vehicle model are developed in the reference frame xy fixed to the Centre-of-Mass (CoM) of the vehicle:

$$ma_x = mV_y\dot{\psi} + F_{xf,w}cos\delta + F_{yf,w}sin\delta + F_(xr, w) \tag{2.12}$$

$$ma_y = -mV_x\dot{\psi} + F_{yf,w}cos\delta + F_{xf,w}sin\delta + F_{yr,w} \tag{2.13}$$

$$I_z z\ddot{\psi} = l_f(F_{yf,w}cos\delta + F_{xf,w}sin\delta) - l_r F_{yr,w} \tag{2.14}$$

where $F_{yf,w}$ and $F_{yr,w}$ are the lateral tires forces applied to front and rear wheels, respectively.

The relation between the inertial reference frame XY and the vehicle-fixed reference

frame xy is then described by the following equations:

$$X = xcos\psi - ysin\psi \tag{2.15}$$

$$Y = xsin\psi + ycos\psi \tag{2.16}$$

$$\Psi = \psi \tag{2.17}$$

The lateral tire forces at the front and rear wheels are considered perpendicular to the rolling direction of the tire, and proportional to the lateral slip angle, α_f and α_r . Considering the assumption of small slip angles, the lateral tire forces are modelled as:

$$\begin{aligned} F_{yf,w} &= 2C_{\alpha_f}\alpha_f \\ F_{yr,w} &= 2C_{\alpha_r}\alpha_r \end{aligned} \tag{2.18}$$

where C_{α_f} and C_{α_r} are the tire stiffness, and α_f and α_r are tires slip angles of the front and rear tires, respectively. Therefore, the tires slip angles are defined as follows:

$$\begin{aligned} \alpha_f &= \arctan\left(\frac{V_y + \dot{\psi} l_f}{V_x}\right) - \delta \\ \alpha_r &= \arctan\left(\frac{V_y - \dot{\psi} l_r}{V_x}\right) \end{aligned} \tag{2.19}$$

where l_f and l_r are distance from the CoM of the front and rear axles, respectively.

The electric racing vehicle is properly instrumented for driving autonomously in the racetrack, and features an integral carbon fibre chassis with honeycomb panels, double wishbone push-rod suspensions, an on-wheel planetary transmission system, and a custom aerodynamic package. The vehicle can reach a maximum speed of 120 km/h with longitudinal acceleration peaks up to 1.6 g. The main vehicle's parameters are listed in Table 2.6.

The commercial passenger car (compact SUV class) has been used in parallel to the racing vehicle for the experimental validation of the proposed localization technique. The main parameters of the retained commercial vehicle are listed in Table 2.7.

Table 2.6: *Main parameters of the retained electric racing vehicle (driver not included).*

Parameter	Symbol	Value	Unit
Mass	m	190	kg
Moment of Inertia about z-axis	I_{zz}	95.81	$\mathrm{N} \cdot \mathrm{m}^2$
Distance from CoM to front/rear axle	$[l_f;l_r]$	[0.839; 0.686]	m
Front/Rear cornering stiffness	$[C_{\alpha_f}, C_{\alpha_r}]$	[44222; 44222]	N/rad

Table 2.7: *Main parameters of the retained commercial vehicle (driver not included).*

Parameter	Symbol	Value	Unit
Mass	m	1270	kg
Moment of Inertia about z-axis	I_{zz}	1550	$\mathrm{N} \cdot \mathrm{m}^2$
Distance from CoM to front/rear axle	$[l_f;l_r]$	[1.02; 1.90]	m
Front/Rear cornering stiffness	$[C_{\alpha_f}, C_{\alpha_r}]$	[65765; 49517]	N/rad

2.6.2 Sensor Modelling

A brief discussion is dedicated also to the sensor modelling considered in this work for data analysis purposes. This section's aim is to introduce basic concepts of error modelling for sensor, giving just an overview of the matter. A deeper insight on the topic is out of the scopes of this work.

Odometry Model

The velocity error introduced by an odometry sensor, i.e. a wheel or motor encoder, can be modelled as:

$$V_{x,o} = \omega_w r_{eff} + n_o \tag{2.20}$$

where ω_o is the angular rate measured by the incremental encoder, n_o is the measurement noise, and r_{eff} is the effective rolling radius. Systematic errors in odometry sources include inequalities in wheel diameters, uncertainties on the wheelbase measure, misalignments of wheels, and finite encoders' resolution. Non-systematic errors on the opposite include motion on inclined planes, wheel-slippage, and external forces. The wheels should ideally be non-compressible and 2-dimensional (knife-edge thin) to minimize errors.

Gyroscope

The principal source of errors on the gyroscope measurements is given by the angular rate measure, because of the integration step needed to indirectly derive orientation angles. The angular rate measure of a gyroscope can be in general modelled as:

$$\dot{\theta}_g = \dot{\theta} + b_g + n_g \tag{2.21}$$

where $\dot{\theta_g}$ is the angular rate measured by the gyroscope, b_g is the sensor bias, and n_g is the measurement noise.

Steering Angle

The last relevant sensor model which should be considered when adopting the 3-DoF model proposed, involves the steering angle δ detection. The steering angle is commonly measured by a potentiometer inside the steering servo mechanism, and the angle measure can be modelled as:

$$\delta_s = \delta + n_s \tag{2.22}$$

where δ_s is the steering angle measured by the potentiometer, and n_s is the measurement noise of the potentiometer.

Chapter 3

Design and Implementation

This chapter deals with the actual design and the final implementation on the vehicles considered of the localization pipeline which is the scope of this work. The theoretical background presented so far, represents the recollection of the material investigated by the authors before the decision-making process behind the chosen hardware and software solution. When the term architecture for localization is used in autonomous system's design, the reference is to both the hardware used (processing and sensing units, such as embedded computers and sensors), and to the software components deployed on them. Furthermore, some theoretical notions will be introduced and discussed also in this chapter, mainly concerning the non-linear filtering techniques introduced in the previous analysis, which have major impacts on the general performance of the pipeline.

The first section of the discussion is dedicated to the requirements definition, which represent the guidelines for the implementation under study. Consequently, a brief recap of the framework of this thesis work is needed to complete the background, with a particular attention to the integration of the localization stack within the complete autonomous system proposed by Squadra Corse PoliTo Driverless. The development of the driverless system has been carried out through the effort of different divisions following a common technical direction. Thus, the design under discussion has been guided and constrained within the common framework and the requirements of the other divisions. In the case of localization, the tasks more affected by the delivered

performance are the Global Mapping and Motion Planning, which require high levels of accuracy and robustness in state estimates.

The second section presents the design choices regarding the hardware architecture: here the trade-off between performance and requirements is solved from the hardware point of view, which is fundamental to guarantee the effectiveness of the pipeline, without adding too many levels of complexity. The final set of sensors chosen and used for localization is presented and analysed from a quantitative point of view, focusing on the functionalities delivered and the mounting on the vehicles. The sensors considered are the industrial-grade INS and the GNSS receiver, optionally aided for validation by the odometry sources and an additional GNSS antenna.

Then, the most important section of the chapter is the one dedicated to the software development. Here, the choice of a partially decentralized system is presented: the adopted INS allows data pre-processing to be performed at sensor-level, while the main filtering algorithms run at central level, increasing the certainty of the estimation, without compromising the computational efficiency. The starting point of the discussion is a brief introduction to the ROS environment, important to understand how the components designed work. The ROS nodes implemented for localization are presented in their characteristics and functionalities; moreover, a theoretical analysis on the filtering techniques used is proposed, considering both the design of the EKF and the UKF solutions to state estimation. Only the theoretical description of the algorithms is reported in this chapter, while the complete code is available from the references cited.

Finally, in the last pages of the chapter, it is presented an overview of the whole pipeline proposed to explain its integration in the complete ADS designed by Squadra Corse PoliTo Driverless.

3.1 Requirements Definition

The first fundamental step in the design process of the pipeline proposed, is represented by the requirements definition. The correct statement of the problem under discussion and the comprehensive description of the design framework, represent a best-practice

of engineering design. Furthermore, when developing and delivering systems or components for a competition, the correct understanding of the regulations is not only preferable, but also mandatory. The challenge proposed by the Formula Student Competition follows a rules book which is at the base of the development of the ADS: this kind of scenario is not different from the common industrial design one, where the objectives are different, and the constraints fixed for other reasons. FSD competition rules mainly deal with safety-oriented guidelines to let the participant Teams express their most advanced competences and level of innovation, without risks.

However, regarding the localization problem proposed in this work, the main requirements do not come directly from the Formula Student rules [13] which leave great freedom from an hardware and software point of view for what concerns the sensing and processing phases of Perception. The requirements presented in this section, are more related to the overall concept of the driverless system proposed by Squadra Corse PoliTo Driverless, and to the availability of the hardware components. More degrees of freedom are obviously available from the software point of view, but as it is analysed in this section, this is the area in which the philosophy of the systems plays a much more important role.

3.1.1 Hardware Resources Management

From the hardware point of view, the architecture of the autonomous system adopted by the first prototype of driverless racing vehicle the *SC19D* by Squadra Corse PoliTo Driverless, is described in Figure 3.1. Overall the system is based on an onboard computing platform responsible for the main autonomous algorithms such as Perception, Localization, Mapping and Path Planning, which communicates with the on-board ECUs of the vehicle, which are in charge of Motion Control.

The main ECU of the SC19D is a dSpace™ MicroAutoBox II which is a real-time system for performing fast function prototyping that can also operate without user intervention as a proper ECU. On the dSpace™ ECU the vehicle's dynamics control algorithm of the car is deployed in real-time. The control algorithm, in normal driving conditions, receives as inputs the driver's requests, performs safety feasibility checks, and delivers a torque request to the motors' inverters controller, consistent with the

vehicle dynamics conditions. In the autonomous driving scenario, the driver's inputs are substituted by the decision-making process developed by the autonomous control system which still runs on the dSpace™ ECU, but needs the computational power of an embedded computer to accomplish the environment perception, to build the map of the track, and position the vehicle accordingly.

The retained on-board computing platform is a NVIDIA™ Jetson AGX Xavier featuring Linux Ubuntu 18.04 and ROS Melodic, which has been chosen for its high computational power, embedded high performance GPU unit (suited for image processing), and compact design. The NVIDIA™ on-board computer hosts the main pipelines of the autonomous stack:

1. *Perception*, which relies on the information captured by a stereo-camera and a LiDAR to provide the readings of the constrained environment of the racetrack.

2. *Localization*, which has to provide to the path planner and the global mapping information on positioning on track and estimation of vehicle's states.

3. *Global Mapping*, which combines the perceived environment and the localization information to provide the path planner with a detailed map.

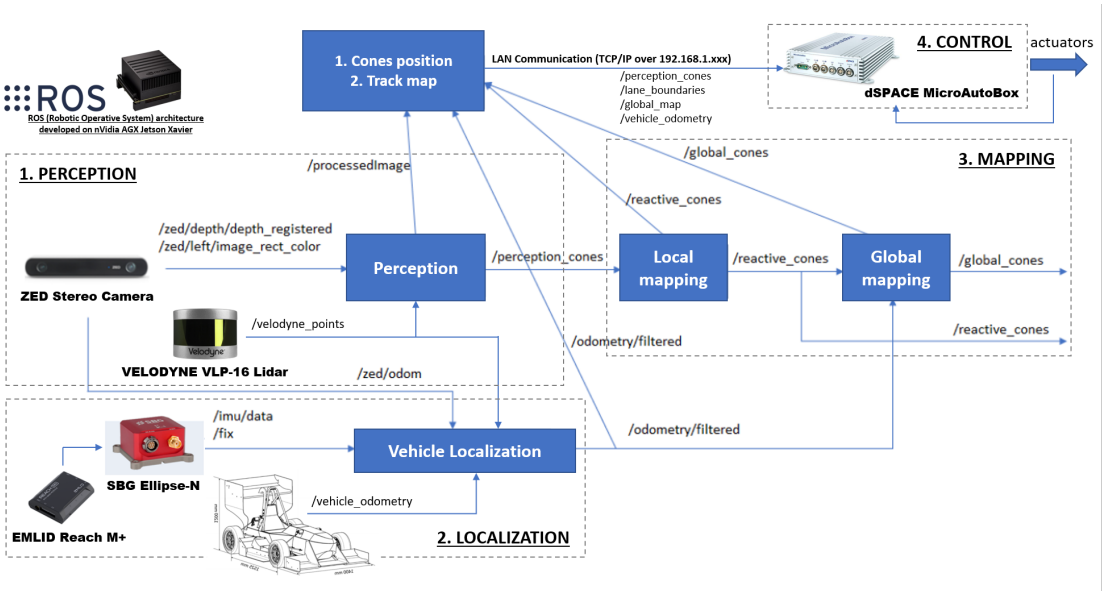


Figure 3.1: *Overall architecture of Squadra Corse Driverless Autonomous Driving System.*

The localization pipeline proposed, relies on the computational power available on the on-board computer, which is divided between the three software stacks of the system. The correct functioning of the pipelines and the distribution of the available memory and CPU resources, are guaranteed by the Robot Operating System (ROS), running on the Linux Ubuntu 18.04 machine. In the following, more information about ROS environment will be given to the reader. For the moment, when dealing with the hardware resources available, the first important constraint on the design of the localization pipeline is given by the adopted computing platform, which imposes to reduce to the minimum the usage of the available resources in order to avoid delays and bottle-neck effects.

3.1.2 Robustness and Racing Challenges

Given the hardware platform chosen for the implementation of the ADS, the challenge of the design of an effective localization pipeline capable of real-time estimation of the complete set of vehicle's states, can be still tackled in different ways. As reported by the research works of other University teams participating in the FSD Competition [15] - [18], the racing challenge and the robustness of the pipeline represent the most important constraints to the design of a driving system of an autonomous race-car. Moreover, guaranteeing the needed performances in terms of responsiveness and thus lap times, with the limited computational resources that can be fitted on the car, require a clever hardware design and a winning hardware/software integration.

The most common solution and also the most effective one, is the choice to decentralize as much as possible the processing effort among the sub-systems composing the architecture. This can be done by adopting sensors equipped with embedded computers or micro-controllers, which process information at sensor-level, and then communicate pre-processed data to the central control unit. This is the most common design solution adopted by many stereo-camera manufacturers for example, which decentralize at camera-level a huge part of image processing effort, and provide as output of the sensor a complete set of information, other than the entire set of raw images. In the choice of the hardware to be used by localization, the same kind of reasoning has been adopted by the choice of an INS, capable of real-time measurement pre-processing.

Finally, dealing with software-related constraints, as we discussed in Chapter 1 of this work, also the philosophy of the racing algorithm adopted, plays a key role in the definition of the design inputs. In the case of localization, as we analysed, the most common and straightforward solution concerns the adoption of Simultaneous Localization and Mapping (SLAM) algorithms, to solve the localization problem together with the mapping one. SLAM is a state-of-the-art solution for many applications on mobile robots and is also adopted in the case of autonomous racing. The SLAM approach is mainly used in combination with global path planners, a type of planning algorithms that rely on the capability of building a complete map of the environment and plan the motion of the robot within the global reference. The information is then exploited to enter in the autonomous navigation phase, in which the system can leverage the global knowledge of the map and enter in the autonomous roaming mode.

However, the ADS designed by Squadra Corse PoliTo Driverless, aim at adopting a local path planner, a class of algorithms devoted to plan the path of the mobile robot within a local map of its immediate surroundings. The challenge of planning the path to be followed by the vehicle is tackled at each time step by the planner on a limited portion of the track [12]. Thus, the algorithm behaves similarly to a human driver which takes the decisions on the trajectory to follow in real-time, turn after turn, based on the perceived environment, without a specific knowledge of the global conditions of the track course. Even in this case, a global knowledge of the complete track is useful to extend the horizon of the driving system, but the fundamental information to be exploited is the one concerning the actual time-step and the local map of the surroundings.

The actual constraint added by the choice of a local path planner on the development of a localization algorithm, is the requirement of very high robustness to sensors' disturbances and failures. The accurate estimation of the vehicle's states in each moment is essential for the performances of the local planner, which should take decisions relying on measurements acquired at the same time instant. In the following, this work analyses how the final pipeline manages to deliver this kind of performance by combining state-of-the-art estimation solutions and industrial-grade sensors.

3.2 Hardware Architecture Design

Given the constraints on the adopted embedded computing platform, the hardware design for localization focuses on the choice of the sensors to be used for the problem solution. As analysed in Section 2.2, two sources of information can be exploited for navigation purposes: self-contained measurements of the states, and absolute measurements with respect to stationary objects. Exploiting the sensor fusion capabilities of commonly available algorithms, it is possible to combine the measurements, and increase their accuracy. As far as state estimation is concerned, the sensor architecture chosen should be able to collect measurements on the states present in the vehicle model considered, guaranteeing that the whole range of these physical properties is covered.

3.2.1 INS: the SBG Systems™ Ellipse-N

The main hardware chosen for localization is an Inertial Navigation System (INS). INS are self-contained, non-radiating, dead-reckoning navigation systems which provide dynamic information through direct measurements, and if integrated with absolute location-sensing mechanisms (such as GNSS receivers), can provide accurate information about the vehicle's position at centimetre level [38]. This capability of INSs can be fully exploited if the sensor has internal computing possibilities: in this case, the fundamental operation of fusing the self-contained measurements and the absolute ones, is performed directly at sensor-level, decentralizing the computational effort and achieving optimal overall results.

The retained INS sensor is the SBG Systems™ Ellipse-N in Figure 3.2, an industrial-grade solution expressly designed for autonomous driving systems. The Ellipse-N embeds a 64-bit microprocessor that is able to process a sensor fusion routine in real-time from measurements of multiple sources: in the case of the Ellipse-N, the sources are a 3-axis IMU, a GNSS receiver, and a barometer. The IMU features industrial grade MEMS components with enhanced vibration rejection and high-end accuracy performances. The main specifications of the 3-axis accelerometer and gyroscope adopted by SBG Systems™ Ellipse-N is reported in Figure 3.3 and 3.4.

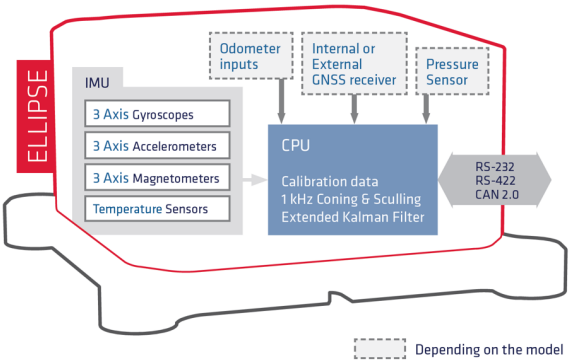


Figure 3.2: *SBG Systems™ Ellipse Series main features.*

	Value	Remarks
Full scale (*g*)	**Marine:** 8 **Land/Air:** 20 **High dynamics:** 40	The Ellipse series are available in three versions depending on the application.
Scale factor stability (ppm)	1000	After one year accelerated aging
Non-Linearity (ppm of FS)	1500	Best Fit Straight Line
One year bias stability (m*g*)	5	After one year accelerated aging
Velocity Random Walk (µ*g*/√hz)	57	
In run bias instability (µ*g*)	14	Allan variance - @ 25°C
Vibration Rectification Error (µ*g*/*g*²)	50	Tested up to 3g RMS for A2 and 10g RMS for A3/A4
Bandwidth (Hz)	390	Internal low pass filters attenuation < 3 dB
Sampling rate (kHz)	4	Advanced anti-aliasing FIR filter
Orthogonality (°)	0.05	

Figure 3.3: *INS accelerometer specifications.*

	Value	Remarks
Full scale (°/s)	**Marine:** 450 **Land/Air:** 450 **High dynamics:** 1000	The Ellipse series are available in three versions depending on the application.
Scale factor stability (ppm)	500	After one year accelerated aging
Non-Linearity (ppm of FS)	50	
One year bias stability (°/s)	0.4	After one year accelerated aging
Angular Random Walk (°/√hr)	0.18	Allan variance - @ 25°C
In run bias instability (°/hr)	8	Allan variance - @ 25°C
Vibration Rectification Error (°/h/g²)	< 1	Tested up to 10g RMS
Bandwidth (Hz)	133	Internal Gyro bandwidth
Sampling rate (kHz)	10	Advanced anti-aliasing FIR filter
Orthogonality (°)	0.05	

Figure 3.4: *INS gyroscope specifications.*

The sensor is small-sized and features an integrated dual-band antenna GNSS receiver, whose specifications are reported in Figure 3.5. The external antenna, mounted on a flat surface of the car, provides orientation angles and accurate GNSS position to the processing algorithm. The usage of this kind of INS, is mostly suited for dynamic environments and harsh GNSS conditions. The INS platform in fact, uses GNSS measurements at a low rate to correct the dead-reckoning position estimated after the initial alignment of the sensor. Nevertheless, it can also operate in lower dynamics applications with a magnetic heading, without the need of initialization, but this is not the case of the automotive application under study.

	Specification		Remark
Signal tracking	**GPS**: L1C/A, L2C **GLONASS**: L1OF, L2OF **GALILEO:** E1, E5b	**Beidou** B1I, B2I **QZSS** L1C/A, L2C **SBAS**	All constellations & signals enabled by default
Horizontal position accuracy	Single point	1.5 m	
	SBAS	1.0 m	
	RTK	1 cm + 1 ppm	
Velocity accuracy	0.05 m/s RMS		
True Heading Accuracy	0,55° 0.4° 0.3°	0.5m baseline 1m baseline 2m baseline	Ellipse-D only.
Velocity limit	500 m/s		
Time to First Fix	Cold start	< 24 s	
	Hot start	< 2 s	
Signal reacquisition		2 s	
Output frequency	5 Hz		
Diff. Corrections	RTCM V3.3		Sent on any serial port

Figure 3.5: *INS GNSS receiver specifications.*

The sensor is fully compatible with the computational platform used on-board the considered vehicles, since it supports both serial communication at high baud rates with the Linux machine running ROS, and the CAN bus communication, with any real-time automotive ECU. Moreover, the retained INS reduces the computational effort required by the embedded on-board computer, because it integrates IMU and GNSS receiver measurements with its embedded 64-bit microprocessor, capable of real-time signal pre-processing and data fusion. The implementation of the sensor fusion routine on the INS platform, based on the adoption of an extended Kalman filter (EKF), will be analysed in the following section (3.3), dedicated to the software architecture design.

3.2.2 Hardware Layout

The hardware layout and the retained racing vehicle dimensions are illustrated in Figure 3.6. The computing platform is connected to a devoted rechargeable lithium battery with a proper custom wiring system, which runs parallel to the main low-voltage system of the car. The INS sensor is directly connected via serial interface to the computing platform, receiving from it the supply voltage and the initialization configurations coming from ROS. The INS is then connected through a standard coaxial cable to the GNSS antenna; the antenna is placed on the main hoop of the vehicle, which has a flat surface and a 20×20 centimetres steel place on which can be placed thanks to its magnetic support (Figure 3.8). An equivalent layout is used on the validation vehicle, the commercial SUV; the layout is not reported in this work, but follows the same implementation steps. For mounting parameters description, refer to Figure 4.2b in the following chapter.

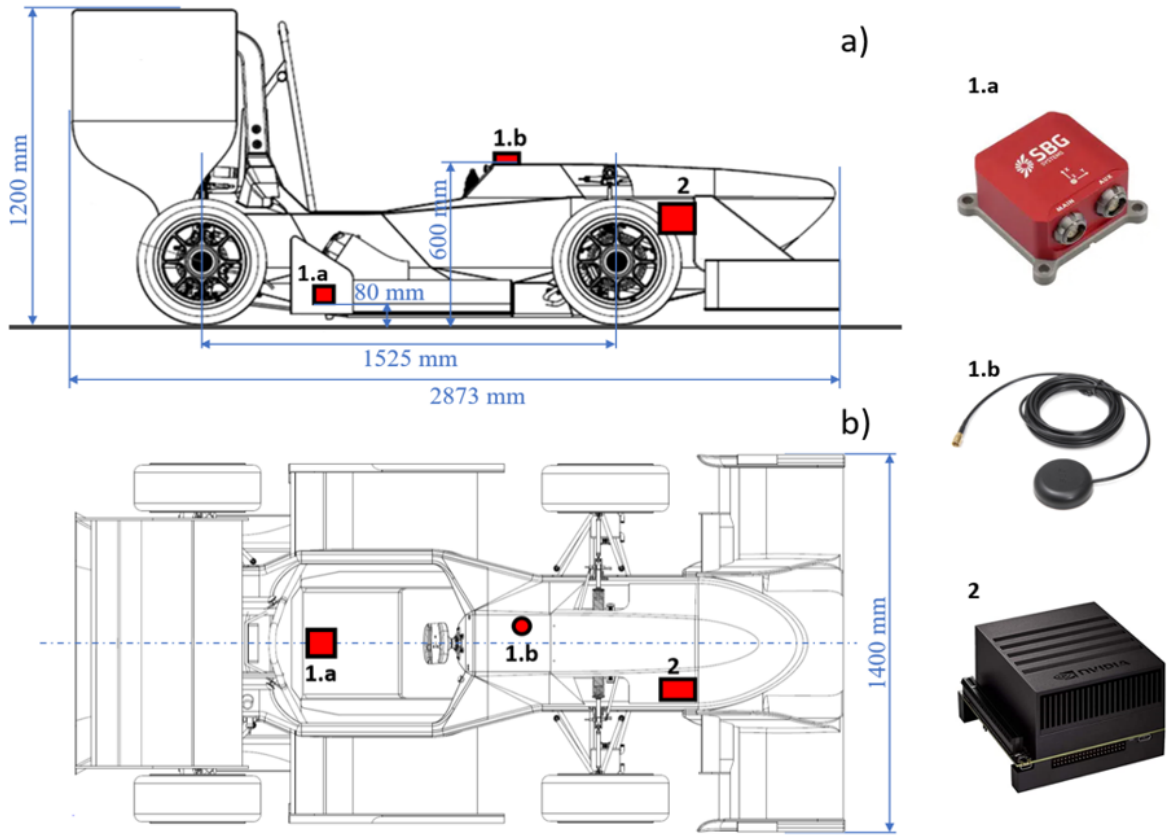


Figure 3.6: *Vehicle layout with hardware positions: a) Side view; b) Top view. 1.a SBG Systems™ INS sensor; 1.b GNSS antenna; 2: NVIDIA™ High-performance computing platform.*

The INS and its placement on the car are represented in Figure 3.7, where also the axis representation of the reference frame associated to the INS is reported. The position of the sensor on the car is close to the CoM of the vehicle: the placement of the unit is irrelevant from the point of view of the measurements acquired, because through the initialization of the sensor performed via software presented in Figure 4.2a, it is possible to account for misalignments, inertial lever arms, and axis orientation. Additional details will be given in the dedicated section (4.1.1).

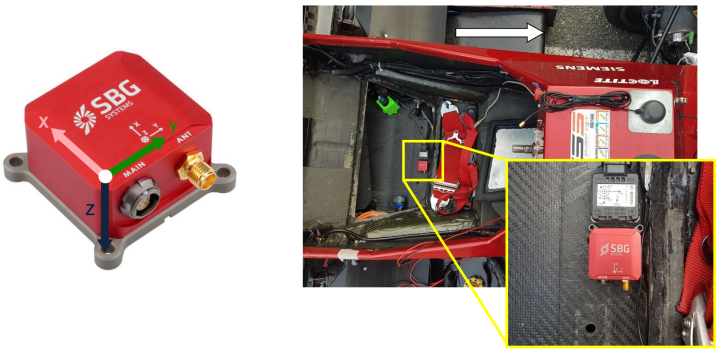


Figure 3.7: *INS coordinate frame representation and actual placement on the SC19D. The white arrow represents the direction of motion.*

Finally, in the Figure 3.8 also the position of the GNSS antenna on the vehicle is showed. The unit needs the clear view of the sky to operate, and a flat surface for the mounting. The mechanical lever arm introduced by the placement of the antenna on the vehicle is also accounted among the initialization parameters of the unit, and will be discussed also in the results presentation.

Figure 3.8: *GNSS antenna and its actual placement on the SC19D. The white arrow represents the direction of motion.*

3.3 Software Architecture Design

In this section, the complete software architecture adopted to solve the localization problem is presented and analysed in detail, with a preliminary discussion dedicated to the framework used. The block scheme of the software components adopted and designed is represented in Figure 3.9. As it is clear from the figure, the main software development framework is the Robot Operating System (ROS), running on the Linux machine of the NVIDIA™ Jetson. All the messages presented in the figure, represent topics of the ROS environment and will be discussed in the following subsection (specifically, with reference to Table 3.1).

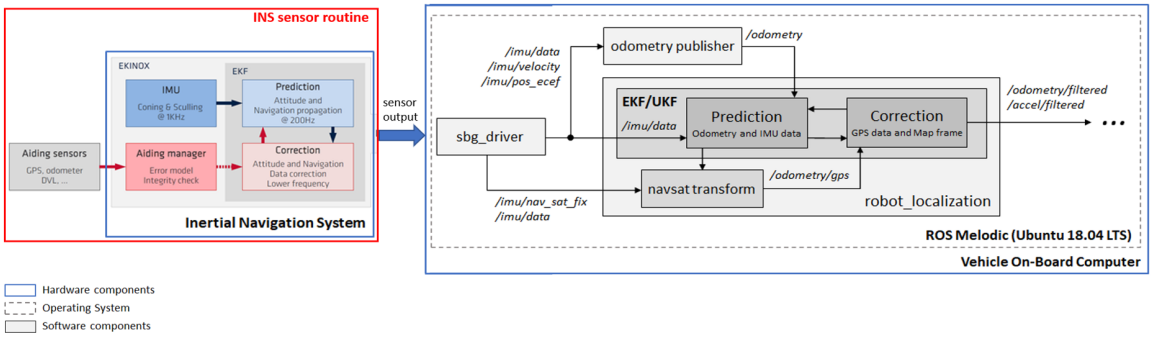


Figure 3.9: *Block diagram of the proposed software architecture.*

The complete architecture used for localization exploits a decentralized filtering architecture, which has been chosen to lighten the computational burden of the on-board computer. In Figure 3.9, the so-called "INS sensor routine", represents the software components running on the micro-processor of the SBG Systems™ Ellipse-N. Specifically, as shown in the block diagram of Figure 3.10, a first stage of sampling and calibration of the IMU via a coning and sculling algorithm, is performed at a frequency of 1 kHz by the sensor itself. The IMU inputs are then processed by an embedded data fusion routine, which integrates GNSS and barometer data by means of an Extended Kalman Filter (EKF), with an output frequency of 200 Hz. The embedded EKF can be properly tuned on the specific application and performs a first pre-processing and fusion stage at sensor-level [39]. The output of the INS sensor routine is the complete set of inertial and navigation measurements, accompanied by sensor's statuses information, and GNSS raw data for post-processing.

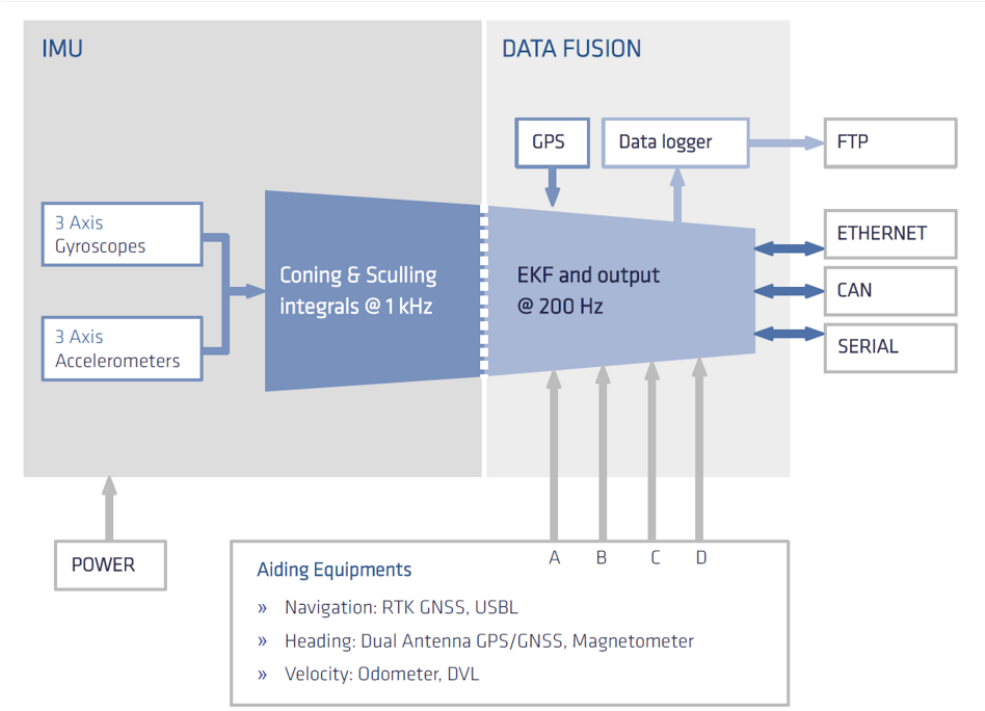


Figure 3.10: *Block diagram of the INS software architecture.*

3.3.1 Robot Operating System (ROS)

The INS sensor routine output is directly fed to the on-board computer via serial communication, where the Linux machine running Ubuntu 18.04 hosts the Robot Operating System (ROS), where the main stages of data processing take place.

ROS is an open-source, meta-operating system for robot software development; consists in a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behaviour, across a wide variety of robotic platforms. It provides the general kind of services expected from a standard operating system, including hardware abstraction, low-level device control, implementation of commonly used functionalities, message-passing between processes, and package management. Differently from a standard operating system it is extremely light-weight, and is oriented to rapid-prototyping of software components and software development. It is considered as a robotic middleware, so a software that connects different software-components and applications [40].

For its flexible characteristics and focus on collaborative robotics software development, ROS has become over the last 10 year a standard for robot programming, and it

Table 3.1: *Retained ROS topics and ROS message types.*

ROS topic	ROS message type	Message content
/imu/data	sensor_msgs/Imu	IMU measurements from INS data
/imu/nav_sat_fix	sensor_msgs/NavSatFix	Raw GPS data from GNSS receiver
/imu/velocity	geometry_msgs/TwistStamped	Linear and angular twist from INS data
/imu/pos_ecef	geometry_msgs/PointStamped	ECEF position from INS data
/odometry	nav_msgs/Odometry	ECEF pose and orientation
/odometry/gps	nav_msgs/Odometry	GPS pose and orientation
/odometry/filtered	nav_msgs/Odometry	Filtered pose and orientation
/accel/filtered	geometry_msgs/AccelWithCovarianceStamped	Filtered acceleration

is very interesting also for autonomous driving software development. The open-source collections of libraries and tools, together with its ecosystem and community, are at the base of the success of ROS as university-level autonomous driving framework and for the same reasons, it has been chosen for the deployment of the autonomous system under investigation.

For what concerns the localization stack, the fundamental data processing and filtering operations are performed by software components developed in ROS, called "ROS nodes". In ROS, the nodes communicate through "topics" on which each node can publish "messages": ROS messages are predefined structures in which the data coming from a node is stored, and different data types can be carried by the same message. ROS messages contain different meta-data and information on their content, depending on the type of message. The ROS messages types considered in this section, are listed in Table 3.1.

When dealing with different sensors' measurements as in the case of localization, ROS environment is also used as source of synchronization. ROS messages carry among their meta-data, the time stamp of the message in ROS time, which is associated to the CPU timing of the Linux machine. This time signal does not correspond to the time at which measurements take place, but it is the time at which the message is processed by the hardware driver. The time stamps of the messages coming from the INS sensor, are directly corrected by the sbg_driver used in ROS, when an external timing signal is used. In the case of the localization pipeline, the external reference is represented by the GNSS timing. Through the GNSS antenna in fact, the sensor receives the time reference from the satellites and the CPU time is synchronized accordingly.

Synchronization is fundamental when dealing with information coming from different sources, and is at the base of the working principle of the transformation conven-

tions used in ROS for reference frames. In the following discussion, the term frame and frame-transformation will be introduced: in ROS, reference frames represent the moving objects at machine-level, so all the information needed to locate the chassis of the vehicle and the position of the IMU, can be retrieved by analysing the "TF tree" or "transformation tree" message.

In the case of localization, the transformation tree is reported in Figure 3.11, and represents the three frames considered. The map and odom frames are created by the filters and represent the fixed frames, originating in the point in which the first measurement is acquired. GNSS positioning information must be transformed from absolute coordinates into the coordinates of the map and odom frames. The base_link frame is the standard frame representing in ROS the chassis of the vehicle: it is obtained by the combining the knowledge of the IMU placement and the centre of motion of the vehicle. The frame associated to the IMU, is one of the information published by the INS routine and is called "UTC | SYNCH", because is the frame associated to the synchronization of the sensor's GNSS with the Coordinated Universal Time (UTC). In our case, this frame is processed by the odometry_publisher node (3.3.2), and is not present in the complete TF tree of the system.

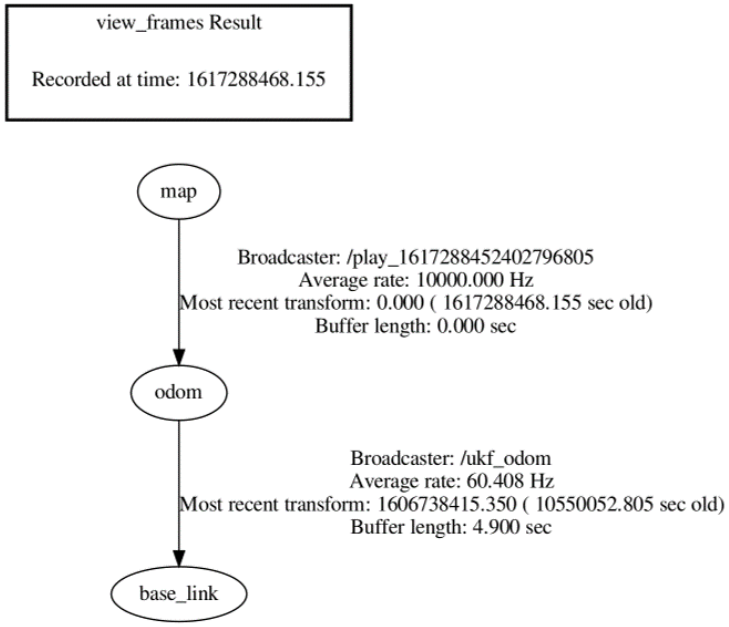


Figure 3.11: *TF tree of the pipeline, from the fixed map frame to the body frame of the INS.*

3.3.2 Custom ROS Packages

The proposed localization pipeline is fully developed by means of properly designed Python and C++ nodes in ROS. The main nodes composing the localization stack are the sbg_driver, the odometry_publisher node, and the robot_localization package. The interaction between the different software components is presented in Figure 3.12, where also the ROS topics used for communication among the nodes are reported.

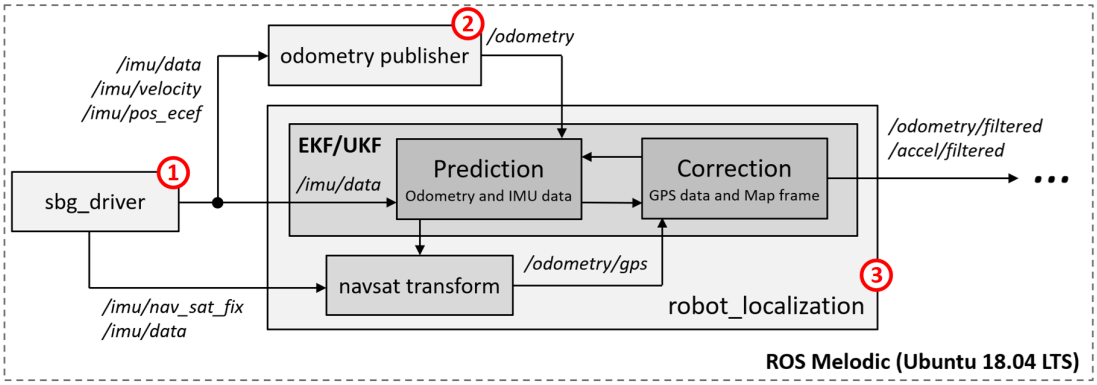


Figure 3.12: *Custom ROS packages block diagram representation with exchanged topics.*

sbg_driver Node

The first node is a SBG Systems™ proprietary C++ node [41], which publishes the filtered sensor data at a custom rate that is decided for each signal on the base of the required robustness. The standard output rate for the Ellipse-N sensor is 200 Hz for each signal, which is the one used in the proposed system. This node publishes proprietary ROS messages, which are used as raw data for post-processing by means of dedicated ROS topics called **/imu/...**, as represented in Figure 3.12.

There are four standard ROS topics published by the node and used by the following nodes of the localization stack: **/imu/data** (sensor_msgs/Imu), **/imu/velocity** (geometry_msgs/TwistStamped), **/imu/pos_ecef** (geometry_msgs/PointStamped) and **/imu/nav_sat_fix** (sensor_msgs/NavSatFix). All the messages refer to the main reference frame of the IMU and are reported in the Earth-Centered-Earth-Fixed (ECEF) world frame, without any transformation between frames published directly by the

node. The node is proposed by SBG Systems™ as a ready-to-use package, and it is mainly used for INS initialisation. A brief discussion on INS initialisation is reported in Chapter 4.

odometry_publisher Node

A custom Python-based ROS node has been designed to properly arrange the information coming from the sbg_driver to be used by the filtering node. The odometry_publisher oversees the filling of a standard ROS topic called **/odometry**, which is part of the navigation messages library /nav_msgs/Odometry. The topic is composed by two geometry messages: **/pose** which refers to the position and orientation of the robot in its frame with respect to a parent frame; **/twist** which contains the information about linear and angular velocities of the linked frame. The resulting odometry message is published in the topic called /odometry and reports position, orientation, and speed of the robot in the East-North-Up (ENU) frame, which is the standard frame in ROS frames conventions. The node performs the transformation between the ECEF frame in which the data gathered by the INS are represented and the ENU frame. In the following, the theory concerning the transformation is reported with reference to [24].

ECEF to ENU transformation

The node implements a code to transform the ECEF coordinates reported by the INS in the /imu/pos_ecef topic to the ENU coordinates required by the /odometry topic to be designed.

The data acquired by GNSS receivers from the satellites, are usually specified in the ECEF geodetic (ECEF-g) coordinate system. The standard convention used by Global Positioning System (GPS) is the World Geodetic System (WGS84), which we have discussed in Section 2.3.5. The coordinates for this frame are usually reported as (λ, ϕ, h) for latitude, longitude, and altitude, respectively. This type of frame is commonly associated with the GPS, which in the case of our application, is used by the INS as main source of satellite navigation information.

To properly transform between a Geodetic coordinate system and a Local Tangent

Table 3.2: *WGS Coordinate System - Reference Ellipsoid parameters.*

μ	WGS-84 Earth's Gravitational Constant	3986004.418×10^8 m^3/s^2
a	WGS-84 Earth Semi-major Axis	6378137.0 m
b	WGS-84 Earth Semi-minor Axis	6356752.3142 m
$f = \frac{a-b}{a}$	Ellipsoid Flatness	3.3528107×10^{-3}
$e = \sqrt{f(2-f)}$	Ellipsoid Eccentricity	8.1819191×10^{-2}

Plane (LTP) one, as the ENU frame which is our target, the following information on the geodetic model of the Earth's surface is assumed. The parameters presented in Table 3.2 are an extension of the parameters introduced in Table 2.3 in Section 2.3.5 dedicated to Geodetic coordinate systems.

The shape assumed is also known as "datum" and the model follows the conventions of the WGS84 used by the GPS. Another parameter to be defined to allow the translation of the coordinates, is the distance from the surface along the ellipsoid normal (N), which is expressed as a function of λ by the following expression:

$$N(\lambda) = \frac{a}{\sqrt{1 - e^2 sin^2(\lambda)}} \tag{3.1}$$

The following step of the transformation is to evaluate the so-called ECEF rectangular (ECEF-r) coordinates from the geodetic ones, which are easier to be related to the LTP coordinates. In general, the ECEF-r (x, y, z) are obtained as function of the ECEF-g (λ, ϕ, h) as follows:

$$\begin{aligned} x &= (h + N) cos\lambda cos\phi \\ y &= (h + N) cos\lambda sin\phi \\ z &= (h + (1 - e^2)N) sin\lambda \end{aligned} \tag{3.2}$$

At this point, the transformation from ECEF-r to LTP can be built up from as a simple sequence of one translation and two rotations. The LTP coordinate system of interest is the ENU frame written as (e, n, u) in the following: it is a right-handed coordinate system, with a strong analogy with the usual (x, y, z) coordinates used to represent a 3-D Cartesian space. The great advantage of the ENU frame, is that the axes coincide with people's common expectation for object positioning on the ground,

and so has been chosen in ROS as the common language for mobile robots' reference frame.

The fundamental passage is to impose the ENU frame to be centred at the WGS84 current Geodetic point (λ_0, ϕ_0, h_0). The coordinates of the origin of the ENU (x_0, y_0, z_0) expressed in ECEF-r are directly obtained with Equation 3.2 by substitution. Now any point represented in the ECEF-r frame can be represented on the surface with the coordinates (x_d, y_d, z_d), accounting the translation from the centre of the Earth (the origin of the ECEF-r frame) to the point of the surface of interest (the origin of the ENU frame).

All the coordinates are now expressed in the same frame, the ECEF-r, so the intermediate coordinate of the considered point can be expressed as a simple translation in the ECEF-r:

$$\mathbf{r^{ECEF}} = \begin{pmatrix} x_d \\ y_d \\ z_d \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} - \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} \tag{3.3}$$

where (x, y, z) is the point of interest expressed in ECEF-r coordinates with the origin already translated into the plane tangent to the surface.

Now to obtain the ENU frame, since it is known its origin's latitude and longitude (λ, ϕ) in ECEF-g coordinates, we should perform two consecutive rotations along the z and x-axis, respectively of magnitude ϕ and λ. The first rotation is introduced by the definition of longitude and is used to align the x-axis along the East direction, while the second is a rotation equal to the latitude in magnitude, performed along the obtained *east*-axis to complete the transformation and make the ENU frame point upwards. A similar procedure is used to find the NED frame, where the latitude angle is used to make the frame point down.

The chain of rotations from the original coordinates (x_d, y_d, z_d) to the final (e, n, u) coordinates is given by the following expression:

$$\mathbf{r^{ENU}} = \begin{pmatrix} e \\ n \\ u \end{pmatrix} = \mathbf{R_1}[\phi] \cdot \mathbf{R_2}[\lambda] \cdot \begin{pmatrix} x_d \\ y_d \\ z_d \end{pmatrix} \tag{3.4}$$

where the rotation matrices $\mathbf{R_1}[\phi]$ and $\mathbf{R_2}[\lambda]$ are defined as:

$$\begin{aligned}\mathbf{R_1}[\phi] &= \begin{pmatrix} -sin(\phi) & cos(\phi) & 0 \\ cos(\phi) & sin(\phi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ \mathbf{R_2}[\lambda] &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & -sin(\lambda) & cos(\lambda) \\ 0 & cos(\lambda) & sin(\lambda) \end{pmatrix}\end{aligned} \tag{3.5}$$

The whole transformation between the coordinates from GPS measurements expressed in the ECEF-r frame with the vector $\mathbf{r^{ECEF}}$, and the coordinates in the target ENU frame expressed by the vector $\mathbf{r^{ENU}}$ is:

$$\begin{aligned}\mathbf{r^{ENU}} = \begin{pmatrix} e \\ n \\ u \end{pmatrix} &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & -sin(\lambda) & cos(\lambda) \\ 0 & cos(\lambda) & sin(\lambda) \end{pmatrix} \cdot \begin{pmatrix} -sin(\phi) & cos(\phi) & 0 \\ cos(\phi) & sin(\phi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_d \\ y_d \\ z_d \end{pmatrix} \\ = \begin{pmatrix} -sin(\phi) & cos(\phi) & 0 \\ -cos(\phi)sin(\lambda) & -sin(\lambda)sin(\phi) & cos(\lambda) \\ cos(\lambda)cos(\phi) & cos(\lambda)sin(\phi) & sin(\lambda) \end{pmatrix} &\cdot \left(\begin{pmatrix} x \\ y \\ z \end{pmatrix} - \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} \right) = \mathbf{R^{ENU}_{ECEF}} \cdot \mathbf{r^{ECEF}}\end{aligned} \tag{3.6}$$

The transformation is possible in real-time thanks to the time synchronization of the input message (the UTC time) with the internal time reference of ROS. The code implemented by the /odometry_publisher node following the theory reported in this section, can be found in Appendix A.1.

robot_localization Package

The last part of the proposed localization pipeline exploits the robot_localization package [42] [43], a standard ROS collection of nodes in C++, freely available among ROS.org libraries [44]. The package represents the central software component of the localization method proposed, featuring different implementations of non-linear state estimators for 3-D motion assessment of mobile robots.

The package embeds both EKF and UKF algorithms, which can be properly adapted to the needs of the application to run separate instances of sensor fusion. The node accepts standard ROS topics as input data for estimation, and publishes the transforms between map, odometry and base link frames of the system, together with the filtered states. The two implementations of EKF and UKF, chosen accordingly to the wanted estimator, are used to fuse the data coming from the INS solution with the raw data of the GNSS, adding the information on the frames dependencies. Alongside the state estimation algorithms, a navsat_transform node is used to integrate in the measurements the GNSS raw data and it is responsible for the global positioning information update. In the Figure 3.13, it is presented the internal working process of the robot_localization package, which embeds the navsat_transform node together with two EKF local and global instances as an example. The implementation with the UKF, uses UKF local and UKF global instead, but works in the exact same way with the same standard ROS topics reported here. The tuning of the filters' parameters can be done by a pre-formatted configuration file (.yaml in ROS), which is reported in Appendix A.2 and A.3 for EKF and UKF respectively.

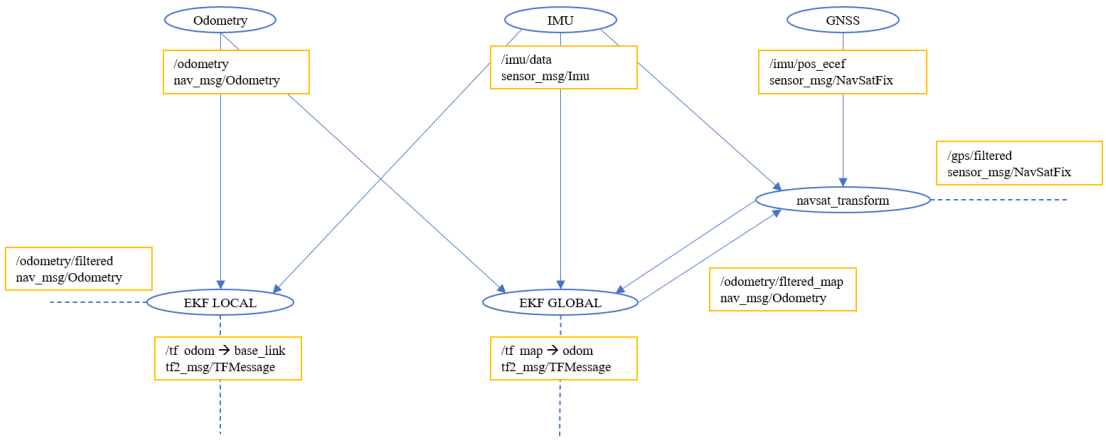


Figure 3.13: *Block Diagram of the robot_localization package working process.*

The odometry source in the figure is the odometry_publisher node, which provides the state estimation instances with a complete odometry message called **/odometry**. The state estimators are also provided with the IMU data message called **/imu/data**, coming from the INS via the sbg_driver node as described before. The local instance

of the state estimator leverages /odometry and /imu/data to establish the transform between the odom and base_link frame and to publish the filtered odometry message under the name of /odometry/filtered. In the meantime, the internal navsat_transform node acquires the GNSS raw data from the INS through the topic **/imu/pos_ecef** of the sbg_driver and the heading information from the /imu/data, providing a new odometry message with the global positioning information called /odometry/filtered_map. Now, a global instance of the state estimator is in charge of the fusion of this global odometry with the local one to finally establish the transform between the odom and the map frames. The result is an **/odometry/filtered** topic containing position and orientation information and a **/tf** message containing the chain transformations between the reference frames.

The message /tf is a standard message used in ROS environment to publish coordinate transformation between frames; more information can be found at [45]. The frames considered by the package are the **base_frame**, the **odom** frame and the **map** frame, which again are standard reference frames used in ROS for mobile robots localization and mapping. The definition and the usage of these frames in this thesis work, follows the standard conventions on frames in ROS defined by the guidelines' repositories which can be found at [46] and [47].

3.4 Vehicle Localization and State Estimation

As discussed in the previous section, among the ROS packages composing the system, the robot_localization is the most interesting one, because it implements the state estimation algorithms which are responsible for the core operations of the pipeline. The performance of the investigated method are in fact mainly linked to this node's working principle, and its parameters' tuning: both from a computational-cost point of view, and a robustness-oriented one, the performance of the estimation performed by the robot_localization package is the central topic of discussion of the results analysis carried out in Chapter 4.

In this section, the attention focuses on the link between state estimation and localization from a theoretical point of view, to better clarify the importance of state

estimation in the solution of the localization problem. The discussion carried out, continues the theoretical dissertation on estimation and filtering techniques applied to sensor fusion, started in Section 2.5. Moreover, the main focus is concentrated on the EKF and UKF algorithms adopted by the robot_localization package, which have been only introduced among the non-linear filtering techniques in the previous discussion.

3.4.1 Estimation Framework Definition

The vehicle localization and states estimation are the main tasks of the proposed method, which solves the problem of estimating the states of a stochastic system from noisy measurements. As a matter of fact, real sensors observations are prone to distortions and fluctuations due to measurement noise, linked as we have discussed to the kind of hardware used, and to the process noise, related to inaccuracies of the retained models. Furthermore, single sensors may not cover the whole range of the physical properties under study, and usually sensors work in unpredictable environment and with limited computational power, which rise the uncertainty of the observed states of the process [29]. Therefore, sensor fusion has been identified in the discussion as the most common way to overcome the addressed problems by combining data from multiple sensors to achieve more specific inferences. Filtering techniques are then used to estimate the states of the process, which as a consequence of the system modelling presented in Section 2.6, needs a non-linear filtering approach for the estimation of non-linear stochastic system's states.

For a generic discrete-time non-linear system dealing with noisy measurements, the presented framework is represented by the following state equations:

$$\begin{cases} x_t = g(u_t, x_{t-1}) + \varepsilon_t \\ z_t = h(x_t) + \delta_t \end{cases} \tag{3.7}$$

where x_t is the state to be estimated, u_t is the control input, z_t is the measured output, ε_t and δ_t are respectively the process and measurements noises. Moreover, an initial state estimate $x_{0|0}$ and the initial estimation error covariance matrix $\Sigma_{0|0}$ are given, and the initial estimate is assumed to be uncorrelated with both ε_t and

δ_t. Even under the assumption of Gaussian noises affecting the system, the filtering problem rises with complexity when dealing with non-linearities, because non-linear transformations of Gaussian variables, do not return Gaussian variables as result.

Among the optimal state estimators, the Bayes filters considered are recursive state estimators working with GRVs. The filtering method proposed by the robot_localization package is conceived to work with a complete set of 15 vehicle's states, namely linear positions (x, y, z), RPY or roll-pitch-yaw angles (ϕ, θ, ψ), linear speeds (v_x, v_y, v_z), angular rates ($\dot{\phi}$, $\dot{\theta}$, $\dot{\psi}$), and linear accelerations (a_x, a_y, a_z). Nevertheless, the filters are initialized in two-dimensional mode for the considered linear 3-DoF Bicycle Model, thus all the linear components along the z-axis are neglected. Also, roll and pitch angles are excluded from the relevant states tracked, because of the low dynamics manoeuvres considered in the first approach of the design proposed.

Therefore, the considered state vector is the following:

$$\mathbf{x_t} = \begin{bmatrix} x & y & \psi & v_x & v_y & \dot{\psi} & \dot{v_x} & \dot{v_y} \end{bmatrix}^T \tag{3.8}$$

The state vector $\mathbf{x_t}$ will be used in the state transitions equations to propagate inferences on the measurements, and to perform the estimation by the two filters considered.

3.4.2 Extended Kalman Filter Design

The first implementation of the robot_localization package considered, is the one involving the EKF. Extended Kalman filters overcome the assumption of linearity in state transition in the standard Kalman Filter algorithm, by linearising the non-linear functions. The distribution is then propagated through a first order linearisation of the non-linear system via Taylor expansion, as reported in 3.13 and 3.14 below.

Given the generic discrete-time non-linear time-variant dynamical system presented in Equation 3.7 it is possible to introduce a linearisation of the dynamic system around a nominal movement of the state $\bar{x}_k$, given a state perturbation δx_k defined in general as:

$$\delta x_k = x_k - \bar{x}_k \tag{3.9}$$

The same kind of reasoning can be done on the linearisation of the input command and the output measurement around their nominal movements, $\bar{u}_k$ and $\bar{z}_k$ respectively. The output perturbation is defined as δz_t in the following, while the input perturbation is neglected, considering the input command as exactly known and thus $\delta u_t = 0$.

At this point, the perturbation dynamics can be well approximated by a linearised dynamical system of the type:

$$\begin{cases} \delta x_t = \bar{G}_t \, \delta x_{t-1} + R_t \\ \delta z_t = \bar{H}_t \, \delta x_t + Q_t \end{cases} \tag{3.10}$$

where the matrices $\bar{G}_t$ and $\bar{H}_t$ represent respectively the Jacobian of g with respect to x and u, and the Jacobian of h with respect to x and u. The matrices R_t and Q_t are the Jacobian matrices of the non-linear functions with respect to the noises.

The Jacobians are generally defined as:

$$\begin{aligned} G_k &= \left. \frac{\partial g(\cdot)}{\partial x} \right|_{x_k = \bar{x}_k} \\ H_k &= \left. \frac{\partial h(\cdot)}{\partial x} \right|_{x_k = \bar{x}_k} \end{aligned} \tag{3.11}$$

In the Extended Kalman Filter framework, the nominal movement of the state and the output measures can be approximated thanks to the introduction of the *a-posteriori* estimate of the state (from a previous time-step), giving a new definition of the dynamical system as:

$$\begin{cases} \bar{x}_t = G_t \, \hat{x}_{t|t-1} \\ \bar{z}_t = H_t \, \bar{x}_t \end{cases} \tag{3.12}$$

where the output Jacobian matrices G_t and H_t are given by:

$$G_t := g'(u_t, \hat{x}_{t|t-1}) = \frac{\partial g(u_t, \hat{x}_{t|t-1})}{\partial \hat{x}_{t|t-1}} \tag{3.13}$$

$$H_t := h'(\hat{x}_{t|t}) = \frac{\partial h(\hat{x}_{t|t})}{\partial \hat{x}_{t|t}} \tag{3.14}$$

The linearisation is performed around the nominal movement of the predicted state

$\hat{x}_k$ defined as:

$$\hat{x}_k = \bar{x}_k + \hat{e}_{x_k} \tag{3.15}$$

where $\hat{e}_{x_k}$ is the one-step ahead evaluation of the prediction error (or innovation) $\bar{e}_{x_k}$ which is defined as:

$$\bar{e}_{x_k} \equiv x_k - \bar{x}_k \tag{3.16}$$

While the main assumption that the initial state and measurement noises are Gaussian and uncorrelated with each other still holds, it is important to notice that the distributions of the random variables considered are no longer normal after undergoing to their respective non-linear transformation.

The complete formulation of the investigated EKF implementation is presented in Algorithm 1, with reference to [48] and [29].

Algorithm 1 Extended Kalman Filter $(\hat{x}_{t-1|t-1},\, \Sigma_{t-1|t-1},\, u_t,\, z_t)$

1: $\hat{x}_{t|t-1} = g(u_t,\, \hat{x}_{t-1|t-1})$

2: $\hat{\Sigma}_{t|t-1} = G_t\, \Sigma_{t-1|t-1}\, {G_t}^T + R_t$

3: $K_t = \hat{\Sigma}_{t|t-1}\, {H_t}^T (H_t\, \hat{\Sigma}_{t|t-1}\, {H_t}^T + Q_t)^{-1}$

4: $\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(z_t - h(\hat{x}_{t|t-1}))$

5: $\Sigma_{t|t} = (I - K_t H_t)\hat{\Sigma}_{t|t-1}$

6: **return** $\hat{x}_{t|t},\, \Sigma_{t|t}$

The process noise covariance and the measurements noise covariance are denoted respectively as R_t and Q_t in Algorithm 1. While the matrix Q_t is assumed constant and will estimate its entries from measurements, the process noise covariance R_t is used as tuning variable according to the application (e.g. dynamic racing situation or urban environment scenario). The output of the EKF algorithm is the sequence of state estimates $\hat{x}_{t|t}$ and of matrices $\Sigma_{t|t}$, starting from the given $x_{0|0}$ and $\Sigma_{0|0}$.

A similar implementation of this EKF algorithm is at the basis of the sensor fusion routine performed at a sensor-level by the SBG Systems™ Ellipse-N, reported in Figure 3.14. Specifically, the INS sensor pre-processes the IMU data at high-frequency to perform the one-step ahead prediction propagation of attitude and navigation infor-

mation. Then, the GNSS raw data is used in the correction step which accounts the innovation brought by the absolute measurements at lower frequency.

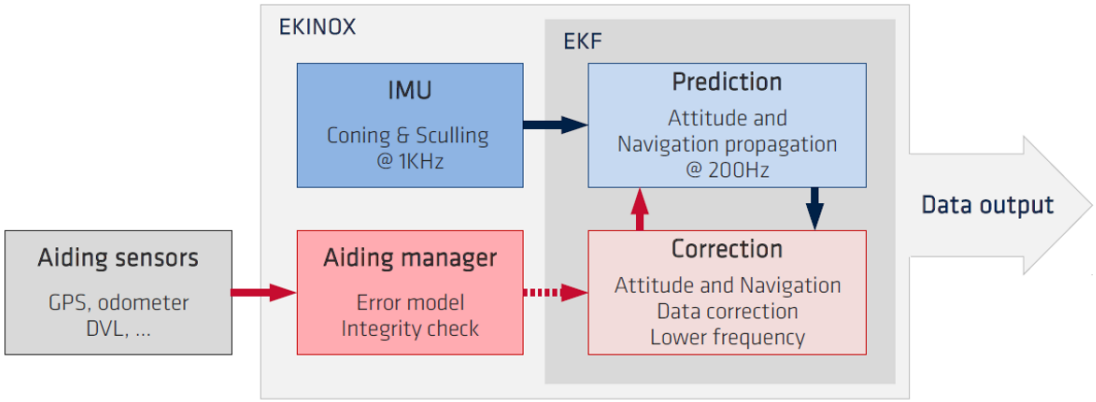


Figure 3.14: *SBG Systems™ Ellipse-N EKF simplified block diagram.*

The result is the data output stream used by the designed pipeline, which accounts the already filtered measurements. This procedure guarantees improved stability and accuracy of the measurements, because the robot_localization filter deals with pre-filtered measurements at higher frequencies. The Ellipse-N EKF runs and provides filtered inputs to ROS at 200 Hz, while the robot_localization instance processes the information and produces the /odometry/filtered topic at a rate of 30Hz. While the INS filter's update rate is tuned according to the motion profile given by the manufacturer, the developed EKF instance running on-board has an update frequency chosen to trade-off the filter's performance and the computational burden of the pipeline.

As it is discussed in Chapter 4, the best performance of the robot_localization is not achieved by increasing the filtering update frequency, but acting on the process noise covariance, which has been heuristically tuned, based on the experimentally collected data (the final configuration can be found in Appendix A.2). Nevertheless, high non-linearities of the process can still affect the EKF instance due to the underlying approximations considered. The introduction of a first-order propagation through the system dynamics, has proven to introduce large uncertainty in the estimate of parameters, thus leading to sub-optimal performance and eventually to filter's divergence [48]. This tendency will be analysed in detail in the results of Chapter 4.

3.4.3 Unscented Kalman Filter Design

The driving scenario of the considered vehicles is characterized by strong non-linearities. Moreover, the usage of a local path planner imposes the need to avoid any possible source of divergence of the measurement in real-time localization and states estimation. In this framework, the implementation of an UKF is investigated as a method to guarantee the expected performance of the pipeline, by improving the results obtained by the EKF. The robot_localization package has been chosen as main software component of the localization method, because of the possibility to interchange the EKF algorithm with the UKF one, by switching the initial configurations. This allowed an effective comparison between the performances of the two filters, applied on the same system in the same testing conditions.

The Unscented Kalman filter has proven in literature to guarantee improved performances in terms of accuracy and robustness of its estimates, with respect to a well-tuned EKF. Furthermore, UKFs have a level of complexity that is comparable with EKFs, thus the required computational effort is expected to be lower than the one required by non-Gaussian Filters, i.e. non-parametric filters [32]. In this work, the UKF instance of the robot_localization package has been adopted to properly address the limitations of the EKF experienced by performing a covariance analysis on the estimates acquired on the experimentally collected data. The reference analysis on the EKF performance and the comparison with the UKF-method presented here, can be found in Section 4.2 of the chapter dedicated to the results.

As discussed in Section 2.5.2, the UKF addresses the sub-optimal performance of EKF by using a deterministic sampling approach. The purpose is to better linearise the non-linear functions (g, h) presented in Equations 3.7, improving the linearisation performed by the EKF and the propagation of the GRV through the system dynamics. Since the UKF is still a Gaussian Filter, the state distribution can still be approximated as a GRV, but in the UKF it is represented by a set of $2n+1$ points based on its original mean and variance, called σ-points. Differently from the EKF, the UKF leverages a method of propagation which preserves the normal distributions throughout the non-linear transformation. A robust theoretical background on the implementation reported, can be found in [30].

When the σ-points are propagated through the non-linear system by means of the Unscented transformation, it has been proven that the filter is able to capture the state estimate's mean and covariance accurately to the third-order of the Taylor series expansion for any type of non-linearity, while for reference the EKF estimate is accurate only to the first-order [49]. In the proposed approach, the implemented UKF formulation is reported in Algorithm 2. The choice of the sigma points as well as the filter weights are reported in the following equations, as discussed in [30]. The UKF σ-points are defined in Equations 3.17 and 3.18, while UKF weights are defined in Equations 3.19 - 3.21.

$$\chi^{[0]} = \mu\chi^{[i]} = \mu + (\sqrt{(n+\lambda)\Sigma})_i \quad for\ i = 1,\ ...,\ n \tag{3.17}$$

$$\chi^{[i]} = \mu - (\sqrt{(n+\lambda)\Sigma})_{i-n} \quad for\ i = n+1,\ ...,\ 2n \tag{3.18}$$

$$w_m{}^{[0]} = \frac{\lambda}{n+\lambda} \tag{3.19}$$

$$w_c{}^{[0]} = w_m{}^{[0]} + (1 - \alpha^2 + \beta) \tag{3.20}$$

$$w_m{}^{[i]} = w_c{}^{[i]} = \frac{1}{2(n+\lambda)} \tag{3.21}$$

where the parameters are chosen as follows:

$\alpha = 0.001, \quad \beta = 2, \quad \kappa = 3 - n \quad$ and $\quad \lambda = \alpha^2(n + \kappa) - n$

According to [30], the choice of $\beta = 2$ minimizes the error in the fourth-order moment of the a posteriori covariance, when the state distribution is a GRV. As already retained for the EKF, the matrix Q_t representing the measurements noise covariance is assumed to be constant, while the process noise covariance R_t is used as a tuning variable.

Although the UKF brings in principle a heavier computational effort with respect of the EKF, a proper tuning of the process noise covariance can guarantee a good performance of the filter in terms of posterior estimate covariance while having the same responsiveness of the EKF instance, as stated by [32]. The robot_localization package implements a similar instance of the UKF, with the same logic discussed for the EKF in the general case of Figure 3.13. The UKF local instance is responsible for

Algorithm 2 Unscented Kalman Filter ($\hat{x}_{t-1|t-1}$, $\Sigma_{t-1|t-1}$, u_t, z_t)

1: $\chi_{t-1|t-1} = (\hat{x}_{t-1|t-1},\ \hat{x}_{t-1|t-1} + \sqrt{(n+\lambda)\Sigma_{t-1|t-1}},\ \hat{x}_{t-1|t-1} - \sqrt{(n+\lambda)\Sigma_{t-1|t-1}})$

2: ${\chi^*}_{t|t-1} = g(u_t, \chi_{t-1|t-1})$

3: $\hat{x}_{t|t-1} = \sum_{i=0}^{2n} {w_m}^{[i]}\ {\chi^*}^{[i]}_{t|t-1}$

4: $\hat{\Sigma}_{t|t-1} = \sum_{i=0}^{2n} {w_c}^{[i]}\ (\hat{\chi}^{*[i]}_{t|t-1} - \hat{x}_{t|t-1})(\hat{\chi}^{*[i]}_{t|t-1} - \hat{x}_{t|t-1})^T + R_t$

5: $\hat{Z}_{t|t-1} = h(\hat{\chi}_{t|t-1})$

6: $\hat{z}_{t|t-1} = \sum_{i=0}^{2n} {w_m}^{[i]}\ \hat{Z}^{*[i]}_{t|t-1}$

7: $S_{t|t-1} = \sum_{i=0}^{2n} {w_c}^{[i]}\ (\hat{Z}^{[i]}_{t|t-1} - \hat{z}_{t|t-1})(\hat{Z}^{[i]}_{t|t-1} - \hat{z}_{t|t-1})^T + Q_t$

8: $\hat{\Sigma}^{x,z}_{t|t-1} = \sum_{i=0}^{2n} {w_c}^{[i]}\ (\hat{\chi}^{[i]}_{t|t-1} - \hat{x}_{t|t-1})(\hat{\chi}^{[i]}_{t|t-1} - \hat{x}_{t|t-1})^T$

9: $K_t = \hat{\Sigma}^{x,z}_{t|t-1}\ {S_{t|t-1}}^{-1}$

10: $\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(z_t - \hat{z}_{t|t-1})$

11: $\Sigma_{t|t} = \hat{\Sigma}_{t|t-1} - K_t S_{t|t-1} {K_t}^{-1}$

12: **return** $\hat{x}_{t|t}$, $\Sigma_{t|t}$

the fusion of only odometry and IMU data, while the UKF global is used to account the global positioning measurements of the GNSS, supported by the same navsat_transform node analysed in Section 3.3.2. The complete robot_localization pipeline produces the same **/odometry/filtered** topic as in the case of the EKF, at a frequency of 30 Hz. The final configuration file of the UKF is presented in Appendix A.3, where also the final values of the process noise and initial state covariance matrices are reported.

In Chapter 4 dedicated to the results, it is analysed how the filters have been tuned to reach the reported performances and the comparison between the results obtained is carried out. Nevertheless, the robot_localization running both the EKF and the UKF instances, is able to deliver an accurate estimate of vehicle's localization and internal states, by means of the /odometry/filtered topic. A secondary output of the pipeline is represented by the /accel/filtered topic, which contains both linear and angular acceleration values to complete the set of the investigated states estimates.

3.5 System Integration

In this conclusive section of the chapter, a brief discussion is dedicated to the overview of the pipeline design and the integration in the complete architecture of the autonomous system.

In Figure 3.15 below, the overview of the localization method, both from the hardware and the software point of view is represented. In the figure it is possible to find also the EMLID™ Reach M+ RTK GNSS module [50], a second GNSS receiver that can be fitted to work properly with the pipeline presented, in order to add redundancy on GNSS raw data. The introduction of the new sensor in the pipeline is under development by another thesis work at the time of writing (early 2021), but some preliminary results on the benefits of this method can be found in Section 4.2.3 of the following chapter.

Another source of redundancy that was initially proposed to improve the robustness of the localization method, is represented by the vehicle odometry. This additional source of odometry, should account the measurements acquired by the motors' encoders already present on the SC19D prototype. Unfortunately, for a problem of integration of the CAN bus communication with the proposed system, the vehicle odometry source has not been introduced in the final realization of the architecture proposed. The introduction of this method can be developed as a future feature for this pipeline, as discussed in the final remarks of Chapter 5.

The possibility to introduce new sensors in the hardware architecture and effectively exploit them from a software point of view, is one of the major opportunities opened by the localization architecture presented. The usage of a multi-purpose package as the robot_localization, fitted for the specific needs of the system, allows the introduction of a common framework that can be only replicated when needed. As we have analysed in this chapter, the package is built on standard ROS messages and conventions, meaning that by following the guidelines given by the framework, it is possible at any time to expand the set of sensors used, and thus improve the performance of the estimations delivered.

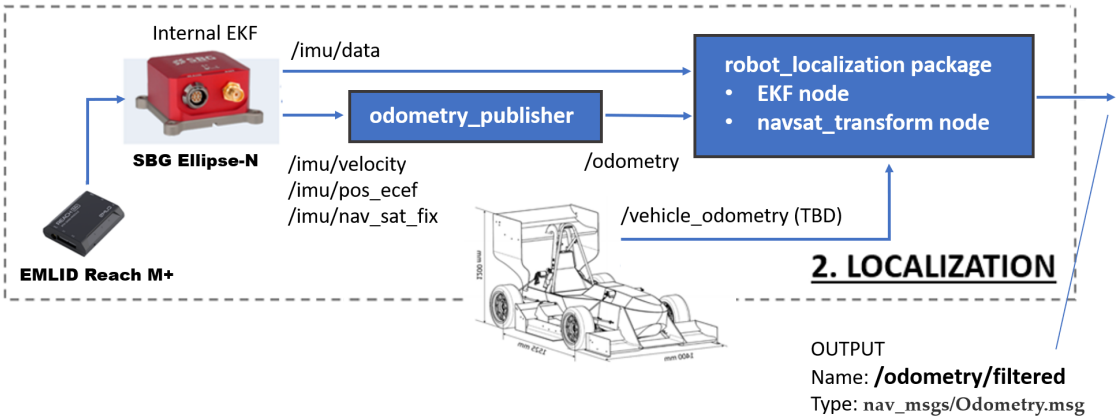


Figure 3.15: *Complete software and hardware architecture for Localization.*

Furthermore, the output of the method proposed is again composed by standard messages in ROS environment, commonly used by different pre-defined packages for running mapping or planning algorithms. One of the main design targets achieved by the proposed method, is to provide the mapping algorithm and the path planner with a precise and robust odometry message complete of means and covariances. The other relevant states of the vehicle are also provided by the /accel/filtered topic and the positioning information for post processing can be found in the /gps/filtered topic. As a result, the integration of the software component with the rest of the autonomous system has been carried out without any trouble, and it is near the time in which the Driverless Team will see all the pipelines running together.

Chapter 4

Results and Discussion

This chapter is dedicated to the results presentation and discussion. The design of the localization pipeline presented has been possible thanks to the experimental data collection and analysis carried out during the development of this thesis work. When dealing with sensors' measurements, especially if the combination of self-contained and absolute measurements is concerned, simulation is not effective to validate the design choices as the collection of real-world experimental data.

The results proposed in this chapter come from the experimental validation of the hardware and software components performed on track. An introduction to the experimental setup is reported before the results presentation and analysis, concerning the hardware layout, the sensors' initialization and the methodology adopted. Having a racing prototype always ready for multiple testing sessions is very difficult, for this reason, also the data coming from a compact SUV have been considered in this work. As analysed in the previous chapters, the modular hardware architecture, and the common software development platform adopted, allowed for rapid prototyping of the new solutions and an effective testing on both the vehicles has been possible. The compact SUV has been mainly used to rapidly test parameters' tuning without the need of setting up the racing prototype. Nevertheless, the final implementation of the solution has been effectively validated on the SC19D, the actual target platform for the method.

(a) *Cerrina racetrack (Sept. 2020).* **(b)** *Aeroclub Torino (Dec. 2020).*

Figure 4.1: *Testing sessions on the SC19D for Localization pipeline validation.*

4.1 Experimental Setup

The development of this work involved several testing sessions on the vehicles considered, so part of the discussion is dedicated to the testing setup. The setup analysis involves both the tracks used for testing purposes and the reference target platform, the SC19D racing prototype by Squadra Corse PoliTo presented in Figure 4.1.

The hardware setup introduced in the previous chapter is reported here, as well as the track setup and the dynamic testing conditions imposed to the driver of the vehicles to have a consistent data collection. In this phase of the autonomous system development, the car is still driven by a human driver for the collection of the datasets and the validation of the perception pipelines, while the planning and decision-making algorithms are tested separately in simulation.

On the other side, the results of the initialization of the sensors used for localization is presented. One of the crucial operations performed by the INS chosen for localization is the alignment phase, which is fundamental to achieve sub-meter accuracy of the estimates. In the dedicated section (4.1.2), the purpose is to validate the expected outcomes from the INS in terms of accuracy on the position measurements, which are the most affected by error accumulation and inertial sensors' drift. In nominal testing conditions, the alignment of the INS is completed before the actual data collection, so it is not reported among the data analysed in the Section 4.2.

4.1.1 Vehicles and Race Track Setup

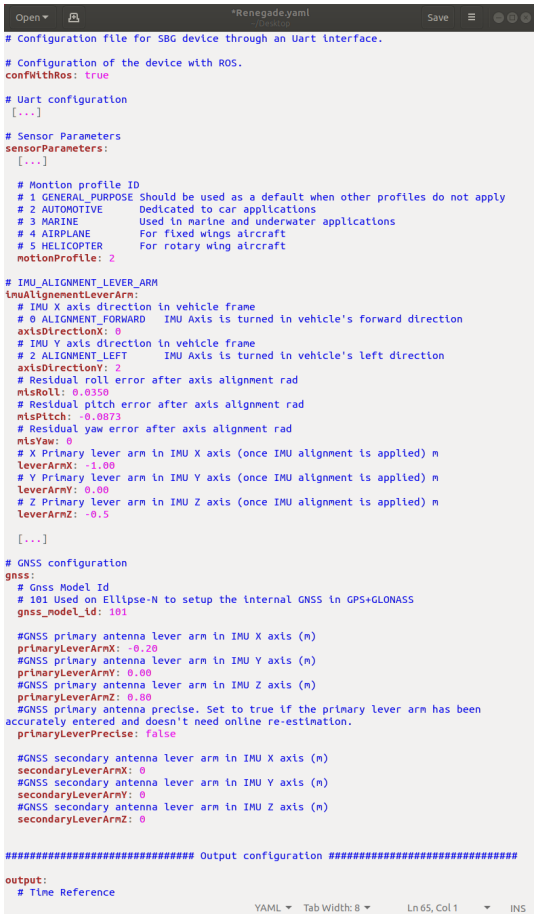
The setup of the vehicles used for testing is similar for both of them, thanks to the hardware architecture chosen. On the SC19D racing prototype by Squadra Corse PoliTo the sensors' placement has been discussed in Section 3.2.2, while for the commercial vehicle, only the setup performed via software will be presented in this section.

The software setup consists in the initialization of the vehicle's parameters in the sbg_driver node, which has an embedded routine for accounting misalignment and mechanical lever arms. Furthermore, the INS uses pre-defined motion profiles to automatically tune the internal EKF: the motion profiles provide the EKF with specific assumptions on the process to be estimated and some peculiar features of its states, under the form of additional a-priori information. In the case of our application, the motion profile is set to the *Automotive Mode*, which includes land vehicles specific assumptions such as:

- Low values of side-slip angles;
- Heading lock and ZUPT (Zero Velocity Potential Update);
- Vehicle's x-axis points in the forward direction.

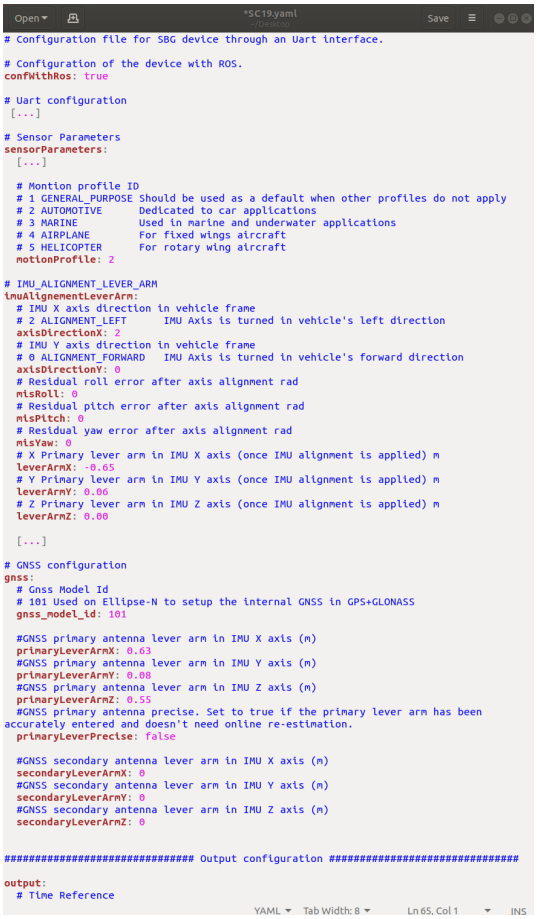
In Figure 4.2, all the input parameters needed by the INS are reported for both the vehicles considered. The alignment accounts for the device rough orientation on the vehicle and the misalignment angles, while the main lever arm is the distance of the INS body from the centre of motion of the vehicle. For what concerns the GNSS antenna, its mechanical installation (the primary lever arm) is its position on the vehicle with respect to the INS body. This lever arm is the fundamental parameter to correct the positions and velocities acquired by the GNSS receiver.

The data acquisitions have involved two different locations, whose Geodetic coordinates are reported in Figure 4.3 below. The Cerrina racetrack has been used in the early stages of the development of this work to validate the hardware setup and collect the results on the INS initialization reported in Section 4.1.2. On the other hand, the final testing of the complete pipeline (Section 4.2), has been carried out at AeroClub Torino, where wider spaces allowed for more complex and representative manoeuvres (sharp turns, u-turns, closed laps, ecc).



```yaml
# Configuration file for SBG device through an Uart interface.

# Configuration of the device with ROS.
confWithRos: true

# Uart configuration
 [...]

# Sensor Parameters
sensorParameters:
  [...]

  # Montion profile ID
  # 1 GENERAL_PURPOSE Should be used as a default when other profiles do not apply
  # 2 AUTOMOTIVE      Dedicated to car applications
  # 3 MARINE          Used in marine and underwater applications
  # 4 AIRPLANE        For fixed wings aircraft
  # 5 HELICOPTER      For rotary wing aircraft
  motionProfile: 2

# IMU_ALIGNMENT_LEVER_ARM
imuAlignementLeverArm:
  # IMU X axis direction in vehicle frame
  # 0 ALIGNMENT_FORWARD   IMU Axis is turned in vehicle's forward direction
  axisDirectionX: 0
  # IMU Y axis direction in vehicle frame
  # 2 ALIGNMENT_LEFT      IMU Axis is turned in vehicle's left direction
  axisDirectionY: 2
  # Residual roll error after axis alignment rad
  misRoll: 0.0350
  # Residual pitch error after axis alignment rad
  misPitch: -0.0873
  # Residual yaw error after axis alignment rad
  misYaw: 0
  # X Primary lever arm in IMU X axis (once IMU alignment is applied) m
  leverArmX: -1.00
  # Y Primary lever arm in IMU Y axis (once IMU alignment is applied) m
  leverArmY: 0.00
  # Z Primary lever arm in IMU Z axis (once IMU alignment is applied) m
  leverArmZ: -0.5

  [...]

# GNSS configuration
gnss:
  # Gnss Model Id
  # 101 Used on Ellipse-N to setup the internal GNSS in GPS+GLONASS
  gnss_model_id: 101

  #GNSS primary antenna lever arm in IMU X axis (m)
  primaryLeverArmX: -0.20
  #GNSS primary antenna lever arm in IMU Y axis (m)
  primaryLeverArmY: 0.00
  #GNSS primary antenna lever arm in IMU Z axis (m)
  primaryLeverArmZ: 0.80
  #GNSS primary antenna precise. Set to true if the primary lever arm has been
accurately entered and doesn't need online re-estimation.
  primaryLeverPrecise: false

  #GNSS secondary antenna lever arm in IMU X axis (m)
  secondaryLeverArmX: 0
  #GNSS secondary antenna lever arm in IMU Y axis (m)
  secondaryLeverArmY: 0
  #GNSS secondary antenna lever arm in IMU Z axis (m)
  secondaryLeverArmZ: 0


############################### Output configuration ################################

output:
  # Time Reference
```

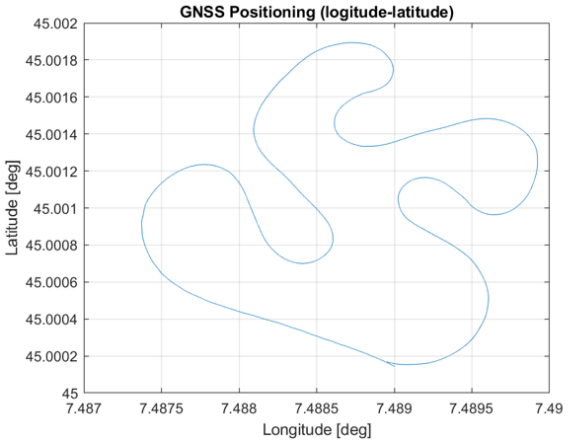
(a) *SC19D INS input parameters.*



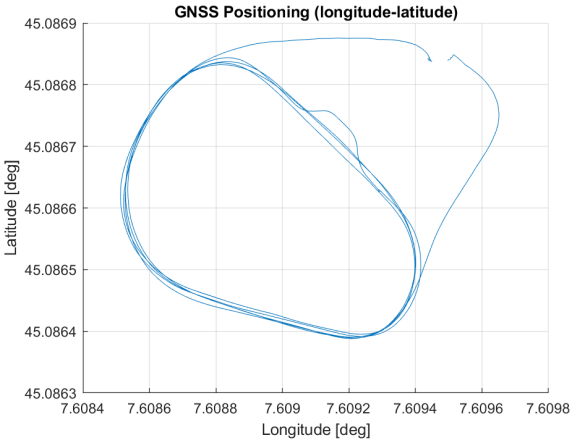
```yaml
# Configuration file for SBG device through an Uart interface.

# Configuration of the device with ROS.
confWithRos: true

# Uart configuration
 [...]

# Sensor Parameters
sensorParameters:
  [...]

  # Montion profile ID
  # 1 GENERAL_PURPOSE Should be used as a default when other profiles do not apply
  # 2 AUTOMOTIVE      Dedicated to car applications
  # 3 MARINE          Used in marine and underwater applications
  # 4 AIRPLANE        For fixed wings aircraft
  # 5 HELICOPTER      For rotary wing aircraft
  motionProfile: 2

# IMU_ALIGNMENT_LEVER_ARM
imuAlignementLeverArm:
  # IMU X axis direction in vehicle frame
  # 2 ALIGNMENT_LEFT      IMU Axis is turned in vehicle's left direction
  axisDirectionX: 2
  # IMU Y axis direction in vehicle frame
  # 0 ALIGNMENT_FORWARD   IMU Axis is turned in vehicle's forward direction
  axisDirectionY: 0
  # Residual roll error after axis alignment rad
  misRoll: 0
  # Residual pitch error after axis alignment rad
  misPitch: 0
  # Residual yaw error after axis alignment rad
  misYaw: 0
  # X Primary lever arm in IMU X axis (once IMU alignment is applied) m
  leverArmX: -0.65
  # Y Primary lever arm in IMU Y axis (once IMU alignment is applied) m
  leverArmY: 0.06
  # Z Primary lever arm in IMU Z axis (once IMU alignment is applied) m
  leverArmZ: 0.00

  [...]

# GNSS configuration
gnss:
  # Gnss Model Id
  # 101 Used on Ellipse-N to setup the internal GNSS in GPS+GLONASS
  gnss_model_id: 101

  #GNSS primary antenna lever arm in IMU X axis (m)
  primaryLeverArmX: 0.63
  #GNSS primary antenna lever arm in IMU Y axis (m)
  primaryLeverArmY: 0.08
  #GNSS primary antenna lever arm in IMU Z axis (m)
  primaryLeverArmZ: 0.55
  #GNSS primary antenna precise. Set to true if the primary lever arm has been
accurately entered and doesn't need online re-estimation.
  primaryLeverPrecise: false

  #GNSS secondary antenna lever arm in IMU X axis (m)
  secondaryLeverArmX: 0
  #GNSS secondary antenna lever arm in IMU Y axis (m)
  secondaryLeverArmY: 0
  #GNSS secondary antenna lever arm in IMU Z axis (m)
  secondaryLeverArmZ: 0


############################### Output configuration ################################

output:
  # Time Reference
```

(b) *Compact SUV INS input parameters.*

Figure 4.2: *sbg_driver configuration files (.yaml, Python language) for the two vehicles.*

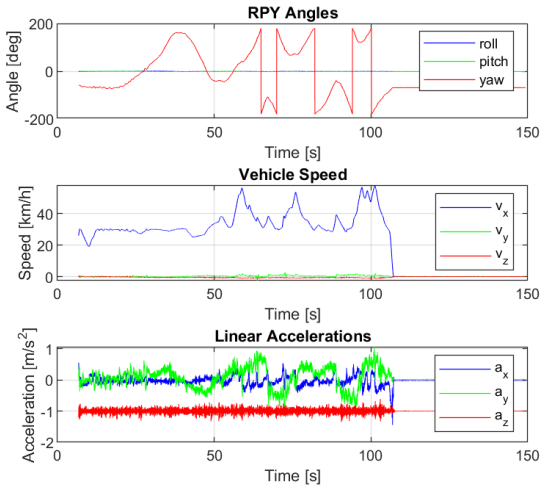


(a) *Geodetic coordinates of Cerrina racetrack.*

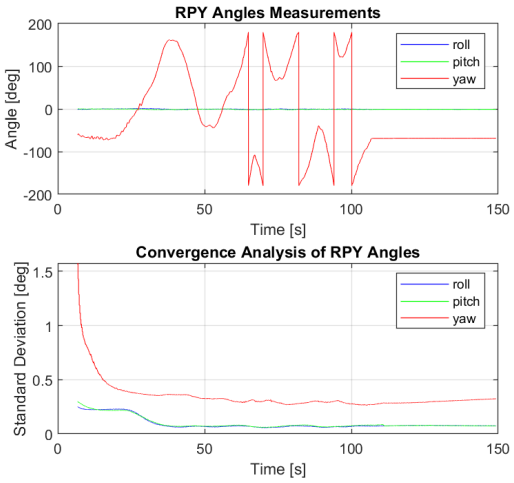


(b) *Geodetic coordinates of AeroClub closed-lap.*

Figure 4.3: *Data acquisition locations for Localization pipeline validation.*



(a) *Main vehicle's states during the test as function of time.*



(b) *RPY angles analysis: measured values and associated uncertainty during the test, both as function of time.*

Figure 4.4: *Accuracy analysis on the main vehicle's states.*

4.1.2 Sensors Setup: INS Initialisation

As presented in Section 2.4.2, INSs exploit the initial knowledge of fixed points to estimate their own positioning in the surrounding environment, through dead-reckoning techniques. In the case of a constant GNSS signal coverage, dead-reckoning is helped by the continuous update of absolute positioning information. Nevertheless the problem of INS initialisation covers a fundamental role in the performance of the sensor during the successive estimation phases.

In the following, the results of performance validation are reported in terms of standard deviation on the sensor's estimates. The analysis of measurements' covariance is the most common and robust method to evaluate accuracy on estimates for the filtering techniques used for sensor fusion [51]. Furthermore, the SBG Systems™ Ellipse-N guarantees the evaluation of the standard deviation for each of its outputs, thus those values have been used to evaluate performances. The states studied in this early stage, are the ones recognized in Section 2.4.1 as the most affected by the measurement drift of the IMU: the gyroscope measured angles and the global positions.

In Figure 4.4a, the main vehicle states of the test performed are presented to give a general idea on the dynamic conditions. In Figure 4.4b the focus is on the angles measurements and their associated standard deviations. It is possible to see how the

IMU measurements are rapidly converging in the early stages of the acquisition, without any substantial oscillation or loss of accuracy during the rest of the run.

The most interesting results of the performed test are reported in Figure 4.5. Thanks to the acquisition, it is possible to evaluate how the accuracy of the global positions' estimates is increasing with the evolution of the initialization stages of the INS. The main passages performed by the internal initialisation routine of the sensor are the following:

1. *Vertical Gyro Mode*: In this early stage, the INS relies mostly on inertial measurements (which rapidly converge to stability) to evaluate the movement of the vehicle on the map. At this stage, the quality of the position estimates is poor, lacking a stable and reliable source of attitude information, and constant GNSS signal updates.

2. *AHRS Mode*: At this stage, the GNSS measurements are acquired and compared with the Attitude and Heading Reference System (AHRS) built only on inertial measurements, but there is still no alignment between the two sources. Whilst the quality is increased, the result is not optimal.

3. *Full Navigation Mode*: The INS is "aligned", because the inertial measurements and the absolute ones output reliable and consistent estimates; at this stage, the fully functional fusion process can take place, eventually improving the accuracy of the single sensor's estimates. The result of the estimation is optimal and the quality of the estimates is the best achievable.

The test performed confirmed the high-level of accuracy achievable by the INS measurements alone. Nevertheless, the evaluated performance on the position measurements refers only to the latitude and longitude positions, without any reference to the accuracy achieved on a local map of the track course completed. A first-order estimate of the position accuracy given the standard deviations of the latitude and longitude angles, is the horizontal position standard deviation σ_{pos} defined as follows:

$$\sigma_{pos} = \sqrt{{\sigma_\lambda}^2 + {\sigma_\phi}^2} \tag{4.1}$$

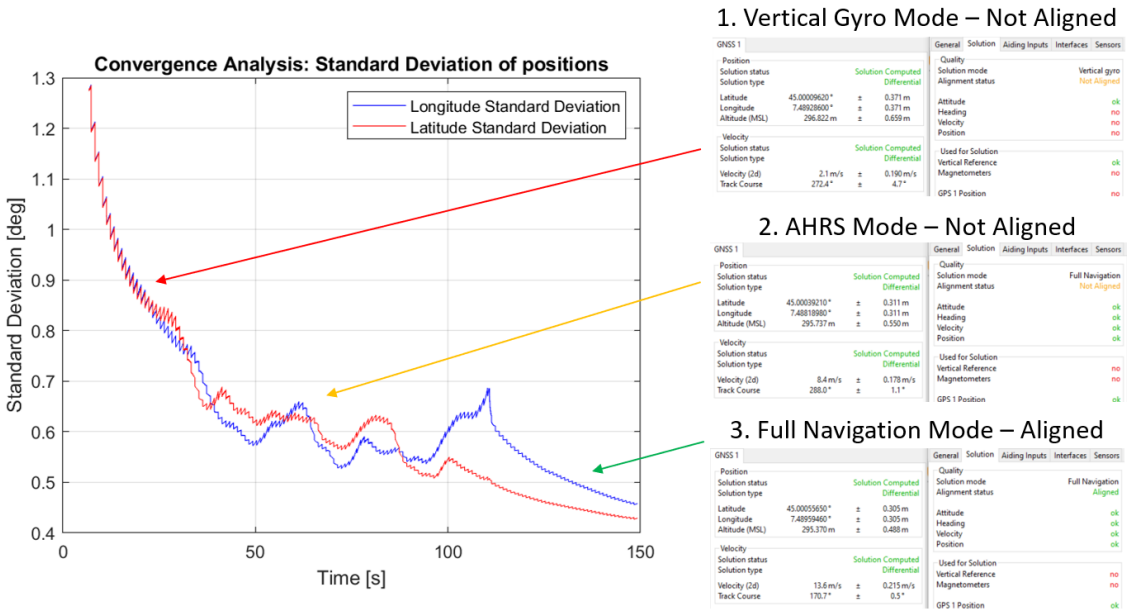


Figure 4.5: *INS initialisation stages reported by the SBG Systems™ software development kit graphical interface (sbgCenter) on the right side. The accuracy on the position estimates associated to each stage of the initialization process, is reported in the chart on the left, as standard deviation of longitude and latitude measures, as functions of time.*

In Equation 4.1, σ_λ and σ_ϕ are the standard deviations of the latitude and the longitude measures, respectively. The resulting horizontal position standard deviation, reported in Figure 4.6 as function of time, is however not a reliable source of information for what concerns the local map position accuracy. In the following, the output position estimate of the robot_localization package accounts the position error on the transformation between the reference frame of the GPS measurements and the ENU frame, resulting in a more accurate and robust evaluation of the positioning accuracy on the local map.

Nevertheless, this preliminary result confirmed the potential of application of the INS to the localization problem under study, and is at the base of the results presented in the following. Once in *Full Alignment Mode*, the INS is a reliable source of odometry and positioning information, achieving sub-meter accuracy in stand-alone mode. The motivation for the adoption of a centralized sensor fusion routine is to improve the delivered performance and expand the possibility of the system to receiver inputs from additional sensors, which will be discussed in the final remarks of Chapter 5.

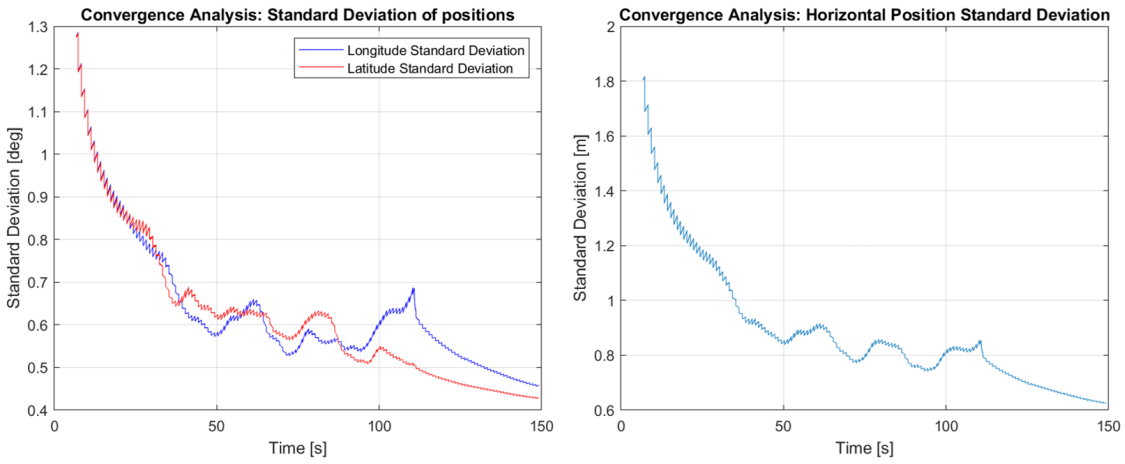


Figure 4.6: *Comparison between longitude and latitude standard deviation σ_λ and σ_ϕ (on the left) and horizontal position standard deviation σ_{pos} (on the right).*

4.2 Validation and Results

In this section, the results of the state estimation obtained with the two investigated filtering methods are presented and discussed. The testing phase has been carried out on both the vehicles introduced, in the same testing environment at AeroClub Torino, but on different paths.

The performance of the investigated algorithms is evaluated on the racing prototype during five consecutive laps, in which the vehicle starts at standstill and finishes its route again with a null longitudinal speed. The same approach has been used in the dataset recorded on the commercial vehicle, however the performance of the pipeline has been tested separately on an open lap and a closed lap.

Both the EKF and the UKF algorithms are properly initialized at the beginning of the acquisition, when the INS sensor output is already computed in Full Navigation Mode, to have an unbiased evaluation of both the filtering techniques, avoiding the influence of the sensor convergence procedure. The motion along the z-axis is neglected as assumed in the previous analysis: it is consistent with the adopted vehicle's dynamics model, i.e. planar motion, and is also compatible with the ground flatness of the chosen testing environment. The measurements are recorded in the ROS topics /odometry/filtered and /accel/filtered (presented in Table 3.1), which report the complete odometry of the vehicle (orientation with covariance and pose with covariance)

and the set of its linear accelerations with covariance, respectively.

Figure 4.7 reports the maximum and mean values of the vehicle states estimates recorded during the two performed tests. Both the proposed filters achieve similar numerical results in the main vehicle's state estimation, with no significant oscillations of the values.

Estimated Vehicle's States		**Dataset1: Closed Laps – SC19D**				**Dataset2: Open Lap – SUV**			
		Mean		**Maximum**		**Mean**		**Maximum**	
State	**Unit**	**EKF**	**UKF**	**EKF**	**UKF**	**EKF**	**UKF**	**EKF**	**UKF**
Yaw Angle	$[deg]$	13.61	13.19	179.98	179.87	-17.69	-17.65	179.58	179.96
Longitudinal Speed	$[m/s]$	4.13	4.12	11.68	11.68	3.57	3.57	6.33	6.32
Lateral Speed	$[m/s]$	-0.37	-0.37	0.08	0.08	-0.04	-0.04	0.15	0.18
Yaw Rate	$[deg/s]$	-0.14	-0.14	0.45	0.46	-0.11	-0.11	0.92	0.92
Longitudinal Acceleration	$[m/s^2]$	0.00	0.00	2.13	2.13	0.00	0.00	1.00	1.00
Lateral Acceleration	$[m/s^2]$	-0.33	-0.33	0.26	0.26	0.48	0.48	0.15	0.15

Figure 4.7: *Maximum and mean values of the estimated vehicle's states in the considered datasets.*

The following discussion is divided in three subsections: in subsection 4.2.1 are discussed the results on the Dataset 1 in Figure 4.7, recorded on board of the SC19D; subsection 4.2.2 presents the outcomes of Dataset 2, the open lap on the commercial vehicle; in the last one 4.2.3 an interesting solution to the EKF divergence tendency is proposed, by adopting an additional GNSS receiver.

4.2.1 Scenario 1 - SC19D Closed Laps

The computed localization paths for Dataset 1 recorded with the racing prototype is presented in Figure 4.8. The paths represented in Figure 4.8 show the vehicle's position on the xy-plane computed by the EKF and the UKF, respectively in blue and red. Specifically, the localization path computed with UKF is less prone to divergence with respect to the EKF one, thanks to the improved estimation of the uncertainty in the unscented transform. Nevertheless, the proposed filters have comparable performance in estimating the vehicle's position on the track having a few meters difference in the xy-coordinate estimation.

The resulting path in ROS is visualized through its graphical interface, RViz. In Figure 4.9 is possible to see how the odometry topic is visualized as a vector pointing in the heading direction of the vehicle. The *map* frame is the fixed reference and

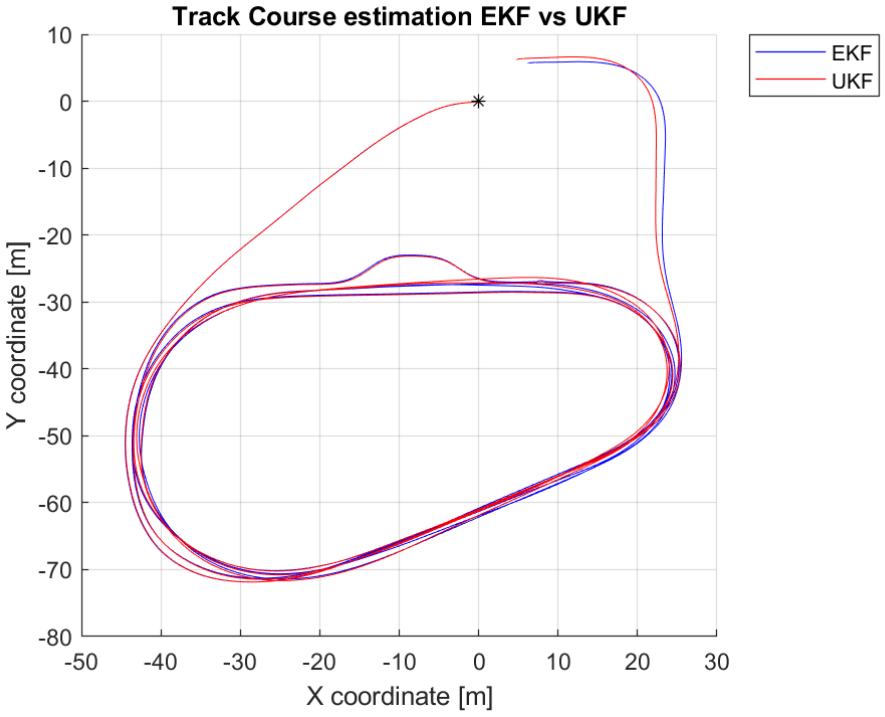


Figure 4.8: *Track Course Dataset 1 estimation, EKF result in blue and UKF result in red.*

is the origin of the reference grid in which the *base_link*, the frame associated to the vehicle body, moves following the desired path. The purple area of the figure represents the "covariance cloud" associated to the instantaneous measurement reported. In our case, the covariance has the shape of an ellipse, whose minor axis is the variance of x coordinate estimate and major axis the one on the y coordinate.

The estimated vehicle states are illustrated in Figure 4.10 for Dataset 1 recorded on the racing prototype. Figure 4.10 reports all the relevant states estimated by the two realizations of the localization pipelines, namely the linear speeds, the linear accelerations, RPY angles and the yaw rate. The motion along the z-axis is neglected in the proposed localization pipeline, thus speed and acceleration along this axis have null values. Only the yaw rate dynamics analysed in sub-figure 4.10 d) reports the comparison of the state estimation for both the filters. The investigated UKF and EKF methods in fact can compute equivalent values for all the other states considered, thus in the rest of the sub-figures only the data computed by the UKF are reported.

The performance of the two methods can be better evaluated by analysing the covariance matrices associated to the state estimates of both the filters [51].

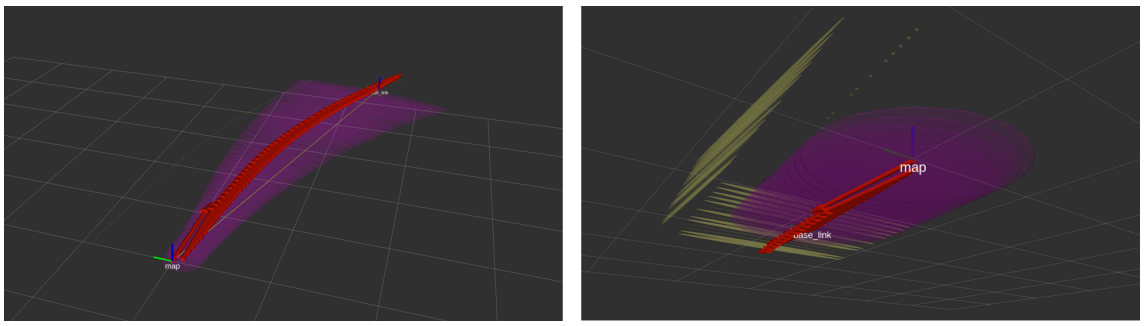


Figure 4.9: *Odometry representation in ROS. The red arrows represent the direction of motion; the origin of those vectors is the actual position of the* base_link *frame at each time-step (the update is fixed to* 30 Hz*), while the yellow line is the instantaneous transformation between the parent frame (*map*) and the child frame (*base_link*). The purple area is a representation of the covariance associated to the estimated positions reported by the odometry topic. The shape is an ellipse, whose axis magnitude increase with the uncertainty of the estimate.*

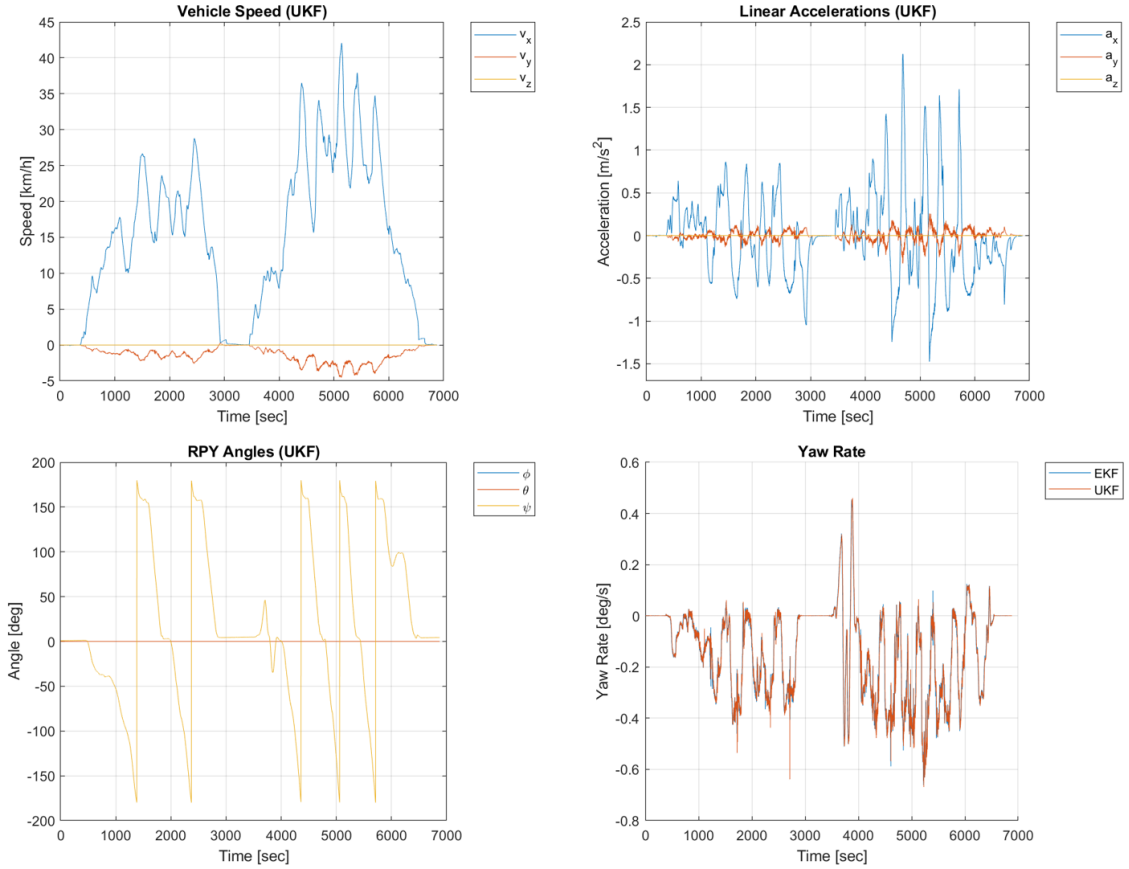


Figure 4.10: *Estimated vehicle's states from Dataset 1. a) Linear Speed with UKF, b) Linear Accelerations with UKF, c) RPY Angles with UKF, d) Yaw rate comparison for the two filters.*

Specifically, this analysis has been carried out on the standard deviation of the filtered states: the considered ROS topics (/odometry/filtered and /accel/filtered) deliver the complete covariance matrix of pose, twist, and acceleration of the vehicle. However, only the position measurements (namely, the x and y coordinates) are characterized by a non-null behaviour of the associated variance. It is possible to see this trend in Figure 4.11 for the EKF-method and in Figure 4.12 for the UKF one. Also the yaw rate estimate suffers from uncertainty increase during the update of both the filters and has a slight tendency to divergence over time. This is due to the needed correction of gyroscope's bias and error. Furthermore, fusing the yaw measure with the GNSS heading information at a lower rate, leads to suboptimal estimation performance, which can be only solved by the usage of a dual GNSS antenna solution.

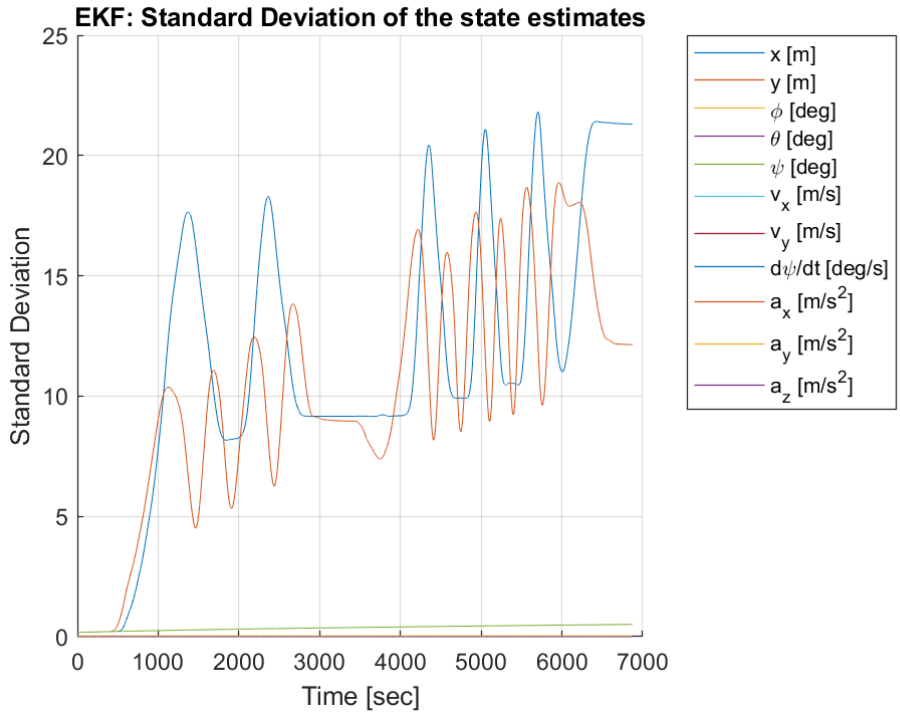


Figure 4.11: *Dataset 1: Standard Deviation of the complete set of estimated parameters by using the Extended Kalman Filter (EKF) embedded in robot_localization package as main sensor fusion algorithm.*

For all the other retained vehicle's states (speeds, accelerations and RPY angles), both the EKF and UKF have consistent performance and are characterized by a low and constant variance, while assuring that the estimated parameters are uncorrelated

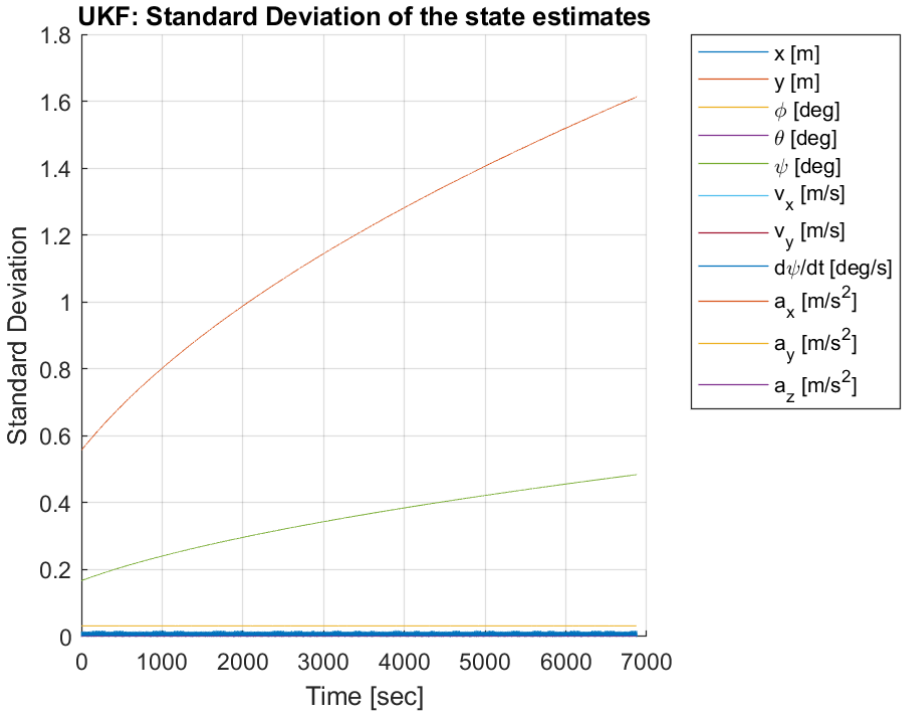


Figure 4.12: *Dataset 1: Standard Deviation of the complete set of estimated parameters by using the Unscented Kalman FIlter (UKF) embedded in robot_localization package as main sensor fusion algorithm.*

with each other (null values of all the off-diagonal elements of the covariance matrix). This result is obtained thanks to both the accuracy of the measurements delivered by the INS sensor and to the pre-processing operation performed by the de-centralized EKF embedded in the sensor.

On the contrary, the filtering process of x and y coordinates suffer a consistent divergence of their associated variance for both the investigated filters, since they depend mostly on the GNSS acquisitions. Even in presence of clear view of the sky during the entire acquisitions, the GPS signal has the lowest acquisition rate among the used sensors, thus it is more prone to divergence. Figure 4.13 presents the horizontal position error (HPE) for the positioning measurements of the EKF-based method in comparison with the UKF-based method. The horizontal position error (HPE) has been chosen as main method of comparison because accounts in a single value all the information associated to the position error on the local map. The HPE can be interpreted as the uncertainty on the current position on the map expressed in terms of mean estimation error on the two coordinates.

The horizontal position error (HPE) can be evaluated starting from the knowledge of the variance associated to the x and y coordinates of the map, as follows:

$$HPE = \sqrt{{\sigma_{xx}}^2 + {\sigma_{yy}}^2} \tag{4.2}$$

where σ_{xx} is the standard deviation of the longitudinal coordinate (x), and σ_{yy} is the standard deviation of the lateral coordinate (y). The horizontal position error calculated for both the filters is reported in Figure 4.13 below.

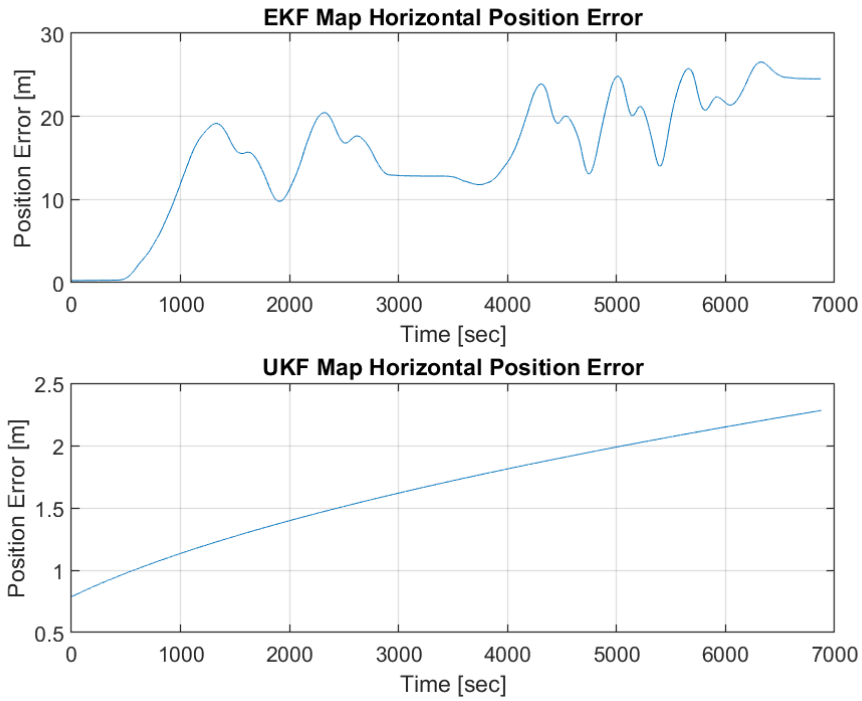


Figure 4.13: *Horizontal position error on the map for both the methods considered on Dataset 1. EKF results on the top and UKF results on the bottom of the figure.*

The UKF-based method is characterized by a better performance in containing the growth of the uncertainty, delivering a consistent and non-oscillating behaviour during the entire acquisition. The overall localization performance on the position of the vehicle on the map built with the GNSS measurements acquired, is better implementing the Unscented Kalman Filter (UKF) as main estimation method to be combined with the EKF routine running at sensor-level.

4.2.2 Scenario 2 - Compact SUV Open Lap

An equivalent discussion about the results obtained with the compact SUV on Dataset 2 is carried out in this section. The results of the data acquisition in the same driving environment are illustrated in Figures 4.14-4.18 during a different set of handling manoeuvres on an open lap.

The computed localization paths for Dataset 2 recorded with the compact SUV is presented in Figure 4.14. The acquisition is performed to validate the robustness of the method on a different type of scenario. The figure reports the paths computed by the EKF and the UKF: also on this dataset, UKF estimation is less prone to divergence, but the overall performance of the two methods is more comparable, due to the shorter duration of the acquisition.

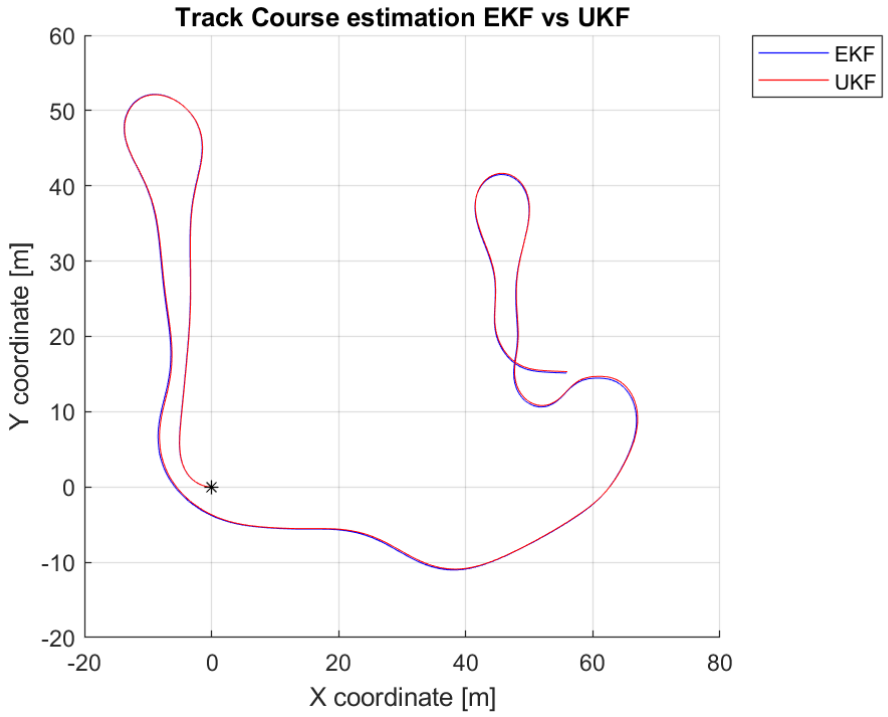


Figure 4.14: *Track Course Dataset 2 estimation, EKF result in blue and UKF result in red.*

The estimated states of the commercial vehicle are illustrated in Figure 4.15, adopting the same logic presented in the previous analysis for an easier comparison.

The performances of the two methods evaluated on the standard deviation of the states are reported in Figure 4.16 for the EKF and in Figure 4.17 for the UKF.

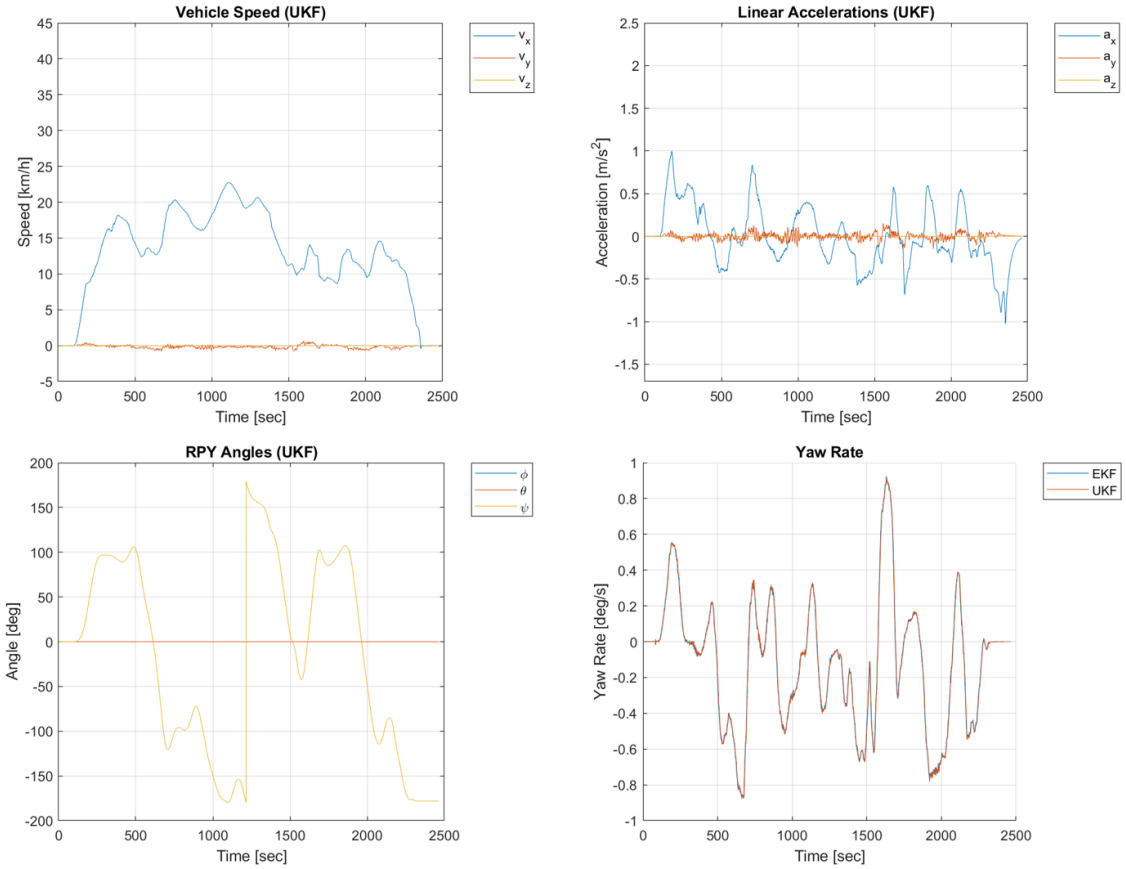


Figure 4.15: *Estimated vehicle's states from Dataset 2. a) Linear Speed with UKF, b) Linear Accelerations with UKF, c) RPY Angles with UKF, d) Yaw rate comparison for the two filters.*

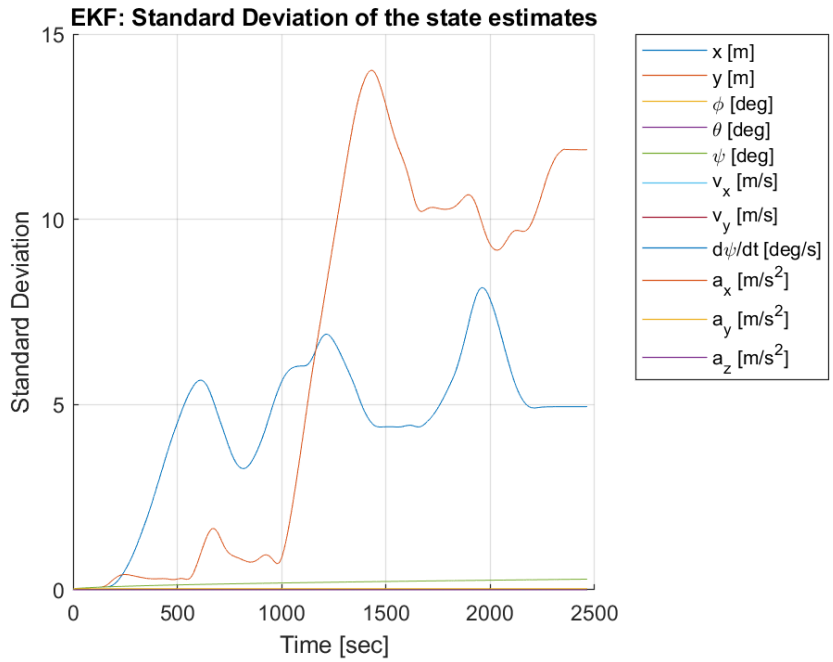


Figure 4.16: *Dataset 2: Standard Deviation of the complete set of estimated parameters by using the Extended Kalman Filter (EKF) embedded in robot_localization package.*

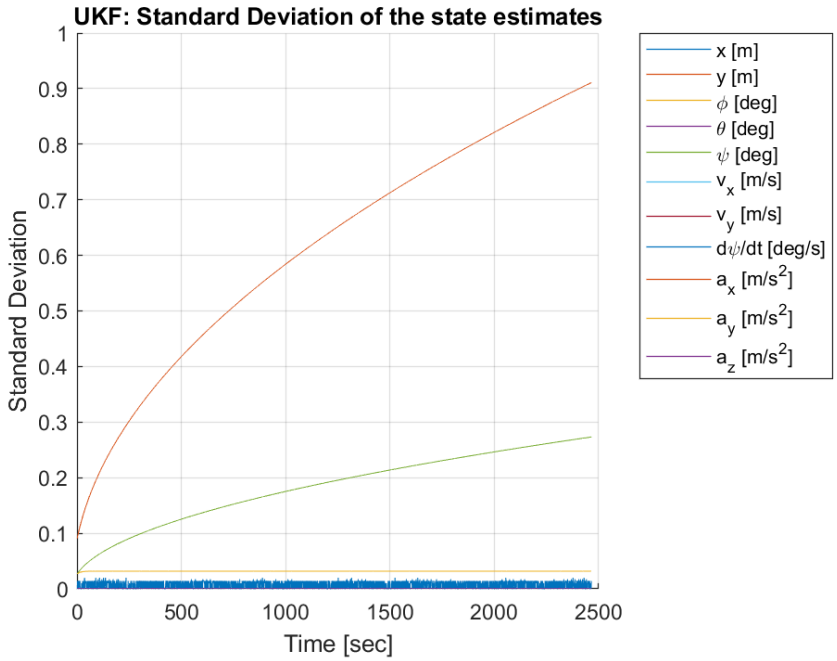


Figure 4.17: *Dataset 2: Standard Deviation of the complete set of estimated parameters by using the Unscented Kalman FIlter (UKF) embedded in robot_localization package.*

The performance of the pipeline is consistent with the results obtained with Dataset 1 in Section 4.2.1 on the racing vehicle. The EKF-method suffers for divergence on the x-y positions as in the previous scenario with the same oscillatory behaviour reported before. The resulting horizontal position error in Figure 4.18 has still one order of magnitude of difference between the methods, with the UKF assuring a more stable and reliable estimation of the position throughout the entire test.

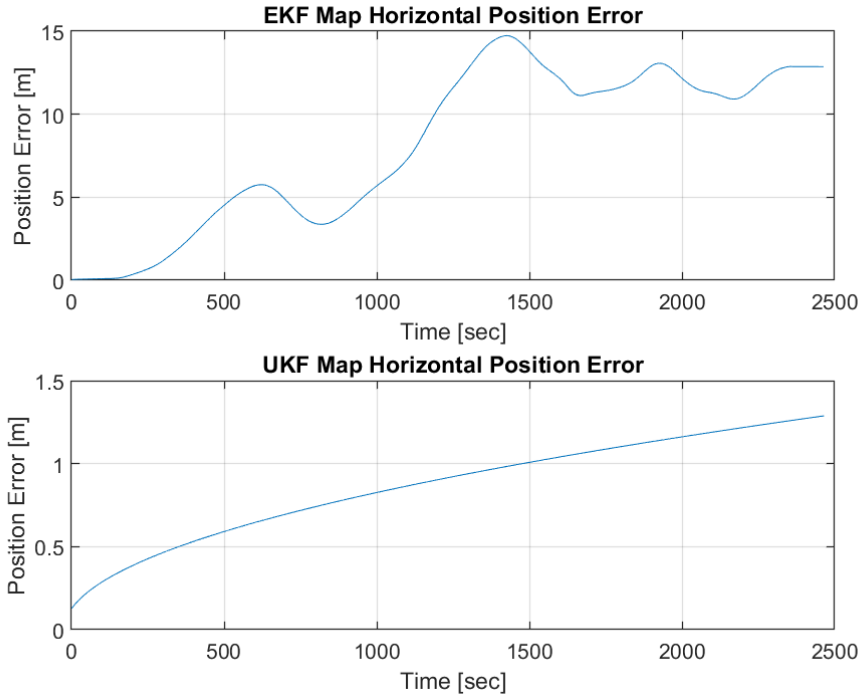


Figure 4.18: *Horizontal position error on the map for both the methods considered on Dataset 2. EKF results on the top and UKF results on the bottom of the figure.*

4.2.3 Scenario 3 - Compact SUV Closed Laps

The final testing scenario considers a different setup of the hardware layout, which introduces an additional source of GNSS raw data among the considered inputs of the robot_localization filters. The additional sensor is the EMLID™ Reach M+ introduced in Section 3.5.

The motivation for this additional experiment is to perform a pipeline robustness test and a preliminary analysis on the benefits of the introduction of additional sources of information. The test is performed on a couple of closed laps performed with the

compact SUV in AeroClub Torino, with the same INS setup considered in Scenario 2. The additional GNSS antenna of the EMLID™ Reach M+ is placed on the anterior part of the roof of the vehicle, in order to guarantee a short-baseline dual-antenna setup.

For what concerns the software setup, the choice has been to use the robot_localization EKF and try to improve its estimates on positions, by means of the additional global position information. The renewed setup of the robot_localization can be found in Appendix A.4. The best performance has been achieved by implementing a double EKF instance of the package, which deals separately with local measures (INS estimations) and global ones (additional GNSS raw data).

The mean and maximum values of the considered test are comparable with the ones reported in Figure 4.7 for the Dataset 2 considered in Scenario 2 (4.2.2). The estimated path followed during the test is reported in Figure 4.19, while the relevant vehicle states considered (vehicle speed and RPY angles) are reported for reference in Figure 4.20.

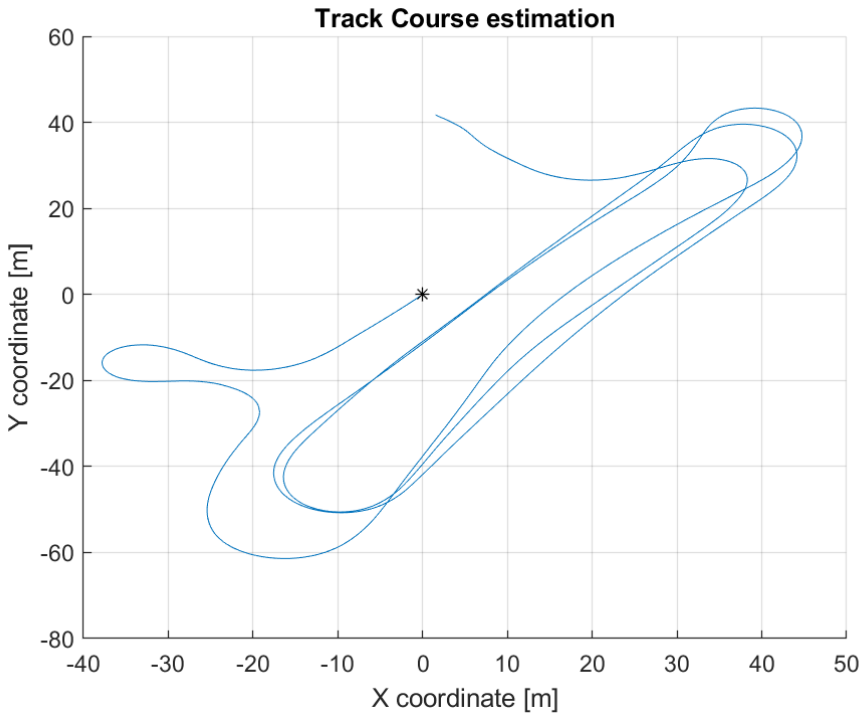


Figure 4.19: *Track Course estimation in Scenario 3, EKF result with additional GNSS source.*

In the following, the preliminary results are reported to show the benefits of the introduction of the additional GNSS raw data. The double instance of the EKF man-

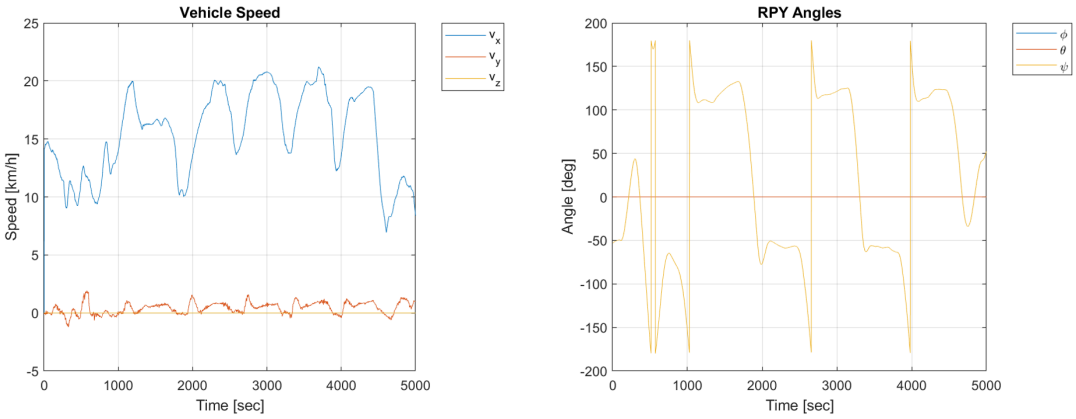


Figure 4.20: *Estimated vehicle's states from Dataset 3. a) Linear Speed and b) RPY Angles.*

ages to stabilize the uncertainty of the estimation, delivering a more accurate result of both the single EKF method and the UKF one. In Figure 4.21 the standard deviation on the complete set of estimates is reported: position estimates achieve sub-meter accuracy during the entire duration of the test, whose length is comparable with the time of Scenario 1. A slight tendency of error accumulation is still present on the position estimates, but it represents a major improvement of the previously analysed methods.

To complete the analysis of the Scenario 3, also the horizontal position error (HPE) has been evaluated to have a consistent comparison with the nominal methods analysed. The result is reported in Figure 4.22 and confirms the enhancements in the overall accuracy expected. The quality of the estimates delivered is satisfactory across the entire test without major divergence tendencies nor oscillatory behaviours as the ones reported in the top of Figures 4.13 and 4.18.

Once again, it is important to clarify that the results reported in this section are just the preliminary outcomes of the implementation of a dual-antenna setup. As it can be seen in Figure 4.21, a strange tendency in the uncertainty of the acceleration measurements has been experienced during the initialization of the double filter method, leading to poor results in the linear accelerations estimation. This issue as well as the fine tuning of the parameters and the validation of the hardware mounting on the vehicle, are the subject of another thesis work parallel to the one proposed, thus these results are for now excluded from the final considerations of this thesis work.

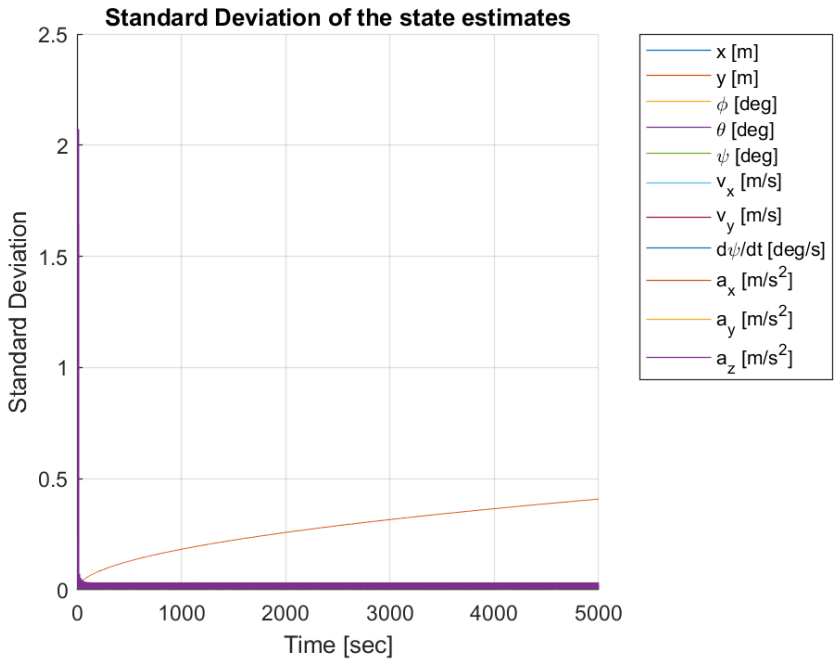


Figure 4.21: *Dataset 3: Standard Deviation of the complete set of estimated parameters by using the double Extended Kalman FIlter (EKF) instances on the dual GNSS receiver setup proposed.*

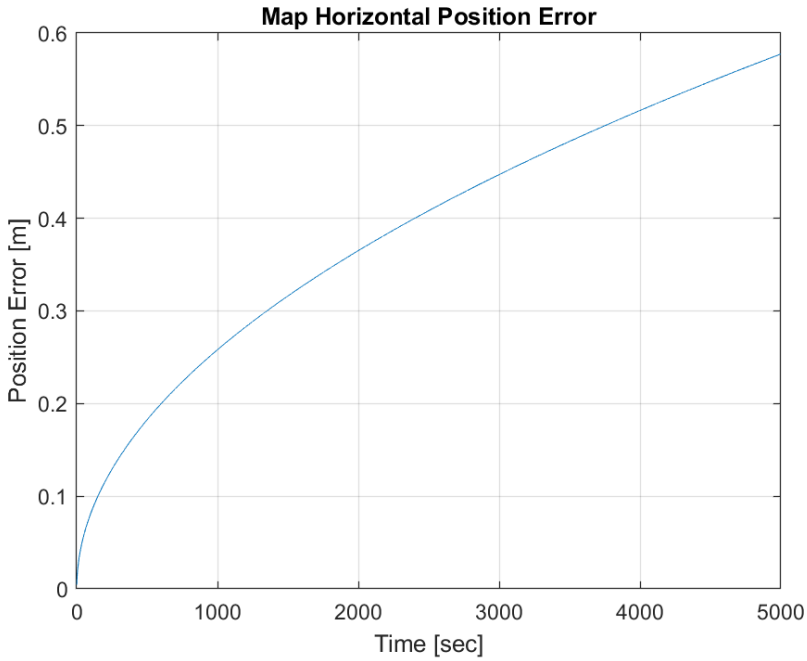
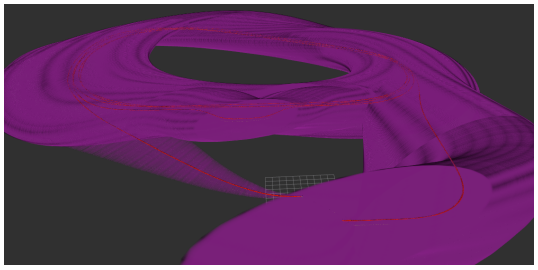
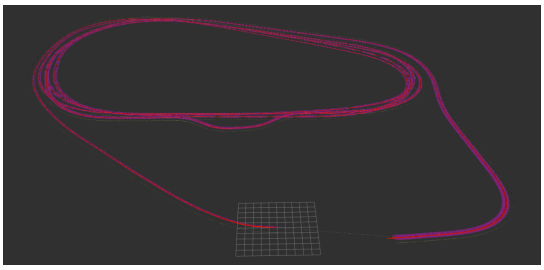


Figure 4.22: *Dataset 3: Preliminary results of the dual GNSS receiver setup. The quality of the overall map position estimate is reported as HPE as function of time.*



(a) *Path visualization in RViz for Dataset 1: EKF-method.*



(b) *Path visualization in RViz for Dataset 1: UKF-method.*

Figure 4.23: *Comparison between path visualization in RViz. The dataset chosen for comparison is the one considered by Scenario 1, which accumulates more uncertainty during the run. a) EKF-method with associated covariance. b) UKF-method with associated covariance.*

4.3 EKF vs UKF Final Comparison

This section is dedicated to the final conclusions on the method used to design the two filters' instances presented. The data presented here, refer only to the validated Dataset considered so far, thus the ones from Scenario 1 and Scenario 2 of subsections 4.2.1 and 4.2.2, respectively. Overall, the comparison between the EKF and the UKF methods used to solve localization can be summarized by the figures presented in this section, which represent both the quantitative and the qualitative difference between the considered methods.

From a qualitative point of view, in Figure 4.23 it is presented the representation of the paths estimated by the filters directly in the development environment, ROS. Through the visualization tool (RViz) it is possible to visualize the positioning result on the map with the associated covariance.

From an implementation point of view, both the methods achieve satisfactory results in the overall state estimation as well as in the computational effort required by the embedded computer. As analysed for the two datasets considered, the main dynamical states of the vehicle useful for controlling the motion and the path planning tasks are estimated with high accuracy and good robustness to sensors' errors and drift. Moreover, as pointed out during the discussion of the pipeline requirements 3.1, the fundamental output of localization to guarantee the effectiveness of a local path planner as main source of motion control, is to achieve the highest level of confidence on the position measurements.

As reported in Figure 4.24 and 4.25, by analysing the direct comparison of the

two localization methods presented, the usage of an Unscented Kalman Filter (UKF) improves the quality of the position estimates for the entire pipeline proposed. This result is more evident analysing the performance on the closed track of Scenario 1, where different system's dynamics are introduced: the presence of fast corners and on the opposite narrow or sharp ones, introduces non-linearities which are badly represented by the first-order linearisation proposed by the Extended Kalman Filter (EKF) framework, thus leading to the accumulation of the errors linked to orientation and position.

The configuration files used to properly initialize the robot_localization package can be found in the Appendix A. The configuration file of the EKF (A.2) and the one of the UKF (A.3) use the same set of inputs and are tuned to achieve the best results in terms of estimates covariance. The results reported in this chapter are all based on the same fixed configuration of the filters, which is the optimal one based on the experimental tuning performed during the acquisitions.

The final implementation of the pipeline on the target racing vehicle, the SC19D, will adopt the UKF-method as main localization source for Path Planning and Global Mapping tasks, with the possibility of extending the set of input sensors available for sensor fusion, thanks to the modular design of the ROS packages used.

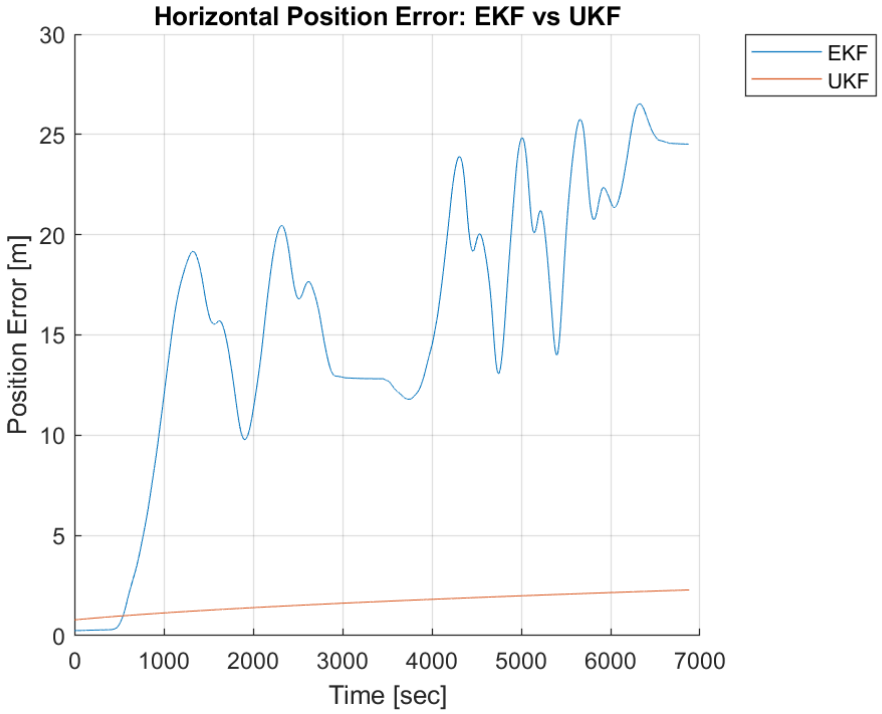


Figure 4.24: *Dataset 1: Comparison of the filters' performance with the horizontal position error.*

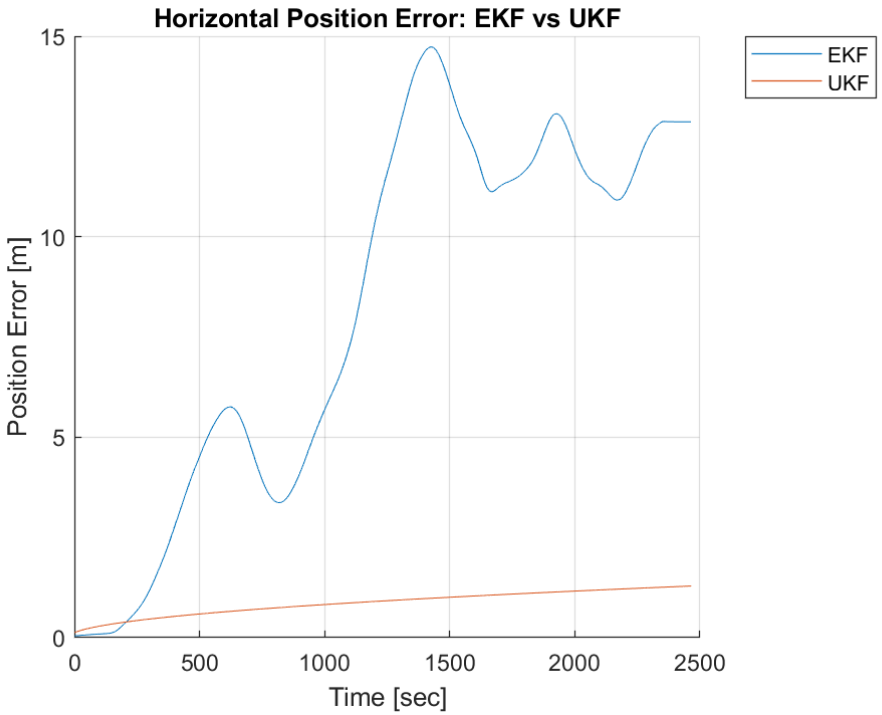


Figure 4.25: *Dataset 2: Comparison of the filters' performance with the horizontal position error.*

Chapter 5

Conclusions and Future Works

Research interest towards autonomous vehicles has recently motivated design challenges related to safety and performances of the next generation of automated cars. Vehicle's robust localization represents one of the most relevant and challenging issues for autonomous vehicles. In the experimental work presented, the Localization problem for an autonomous race car has been addressed and extensively analysed.

A first stage of hardware architecture design has been considered, regarding the choice of the smallest set of sensors useful to achieve satisfactory results on the requirements defined. The second stage of the design, involved the software architecture design, for which state-of-the-art solutions for sensor fusion have been adopted to guarantee the performance of the pipeline. The investigated approach proposed and compared the application of an Extended Kalman Filter (EKF) and an Unscented Kalman Filter (UKF) to the sensor fusion of on-board data streaming from a Global Navigation Satellite System (GNSS) receiver and an industrial-grade Inertal Navigation System (INS).

The solution presented has proven to be a reliable alternative to existing methods, since it can accurately estimate vehicle's states and provide a sub-meter localization accuracy. The performance of the method was evaluated experimentally during handling manoeuvres in real driving environments both on a racing prototype and on a commercial compact SUV. The performance of the complete pipeline has been validated in real-time, showing a substantial equivalence of the application of the two

filters presented. Nevertheless, the UKF-based method is characterized by a lower estimation variance in the localization task, thus providing more robust results. After a further extensive on-field validation stage, the localization method described in this thesis work could provide a reliable pipeline for autonomous vehicles.

In the future developments of this work, the number of sensors considered to implement the localization pipeline can be increased to achieve improved performances. The set of sensors analysed in this work has been constrained by the hardware resources available on the processing unit of the autonomous system, which is in charge of the whole Perception and Mapping stages. Nevertheless, other sensors already considered for those tasks can be used as sources of odometry, such as the adopted ZED™ Stereo Camera from STEREOLABS™ [52] and the adopted Puck™ LiDAR sensor from Velodyne™ [53].

Both the sensors have proven to be reliable sources of odometry information, thus can be adopted in the proposed pipeline as additional entries of the EKF or the UKF implemented by the robot_localization package. As previously remarked, the modular architecture of the solution developed in this work, allows for additional sensors to be considered in order to add redundancy on the states, ultimately achieving better estimation performances. The main drawback in the adoption of stereo cameras and LiDARs as sources of odometry, is the computational cost of the information, which requires additional CPU and GPU power to the NVIDIA™ on-board computer.

As introduced in Section 3.5 and analysed on the data of subsection 4.2.3, the introduction of an additional GNSS receiver in the pipeline provides improved performances at the cost of a more complex overall architecture. As a matter of fact, an extensive study on the integration of the EMLID™ Reach M+ is already under development by another thesis work. The advantage of an additional GNSS receiver for localization is dual: on one side the navigation system can rely on a dual-antenna configuration, which enhances the stability of the heading estimation as proved by the preliminary results presented. On the other side, the EMLID™ Reach M+ can give to the system the possibility of leveraging Real-Time Kinematic (RTK) through the connection over the Internet to an NTRIP Caster [54]. The preliminary results of the adoption of this method are promising, but major results useful to the final implementation are still

lacking. As an example, the exact computational cost of running a double centralized EKF instance on the NVIDIA™ Jetson is still unknown as well as the quantitative benefits of adopting the NTRIP Caster for additional online corrections.

In conclusion, the final implementation of the localization pipeline on its target platform, the SC19D, may also leverage the communication of the SBG Systems™ INS through the CAN network of the vehicle. In this case, the INS sensor is already fitted to receive from the CAN bus the odometry information coming from the incremental encoders of the car's outboard electric motors. The additional source of odometry might enhance the result of the embedded EKF of the sensor in the estimation of the vehicle's states. However, a deeper analysis hasn't been considered in this work because no improvement of robustness on the position information (the least accurate among the states considered) is expected to be introduced by with this method.

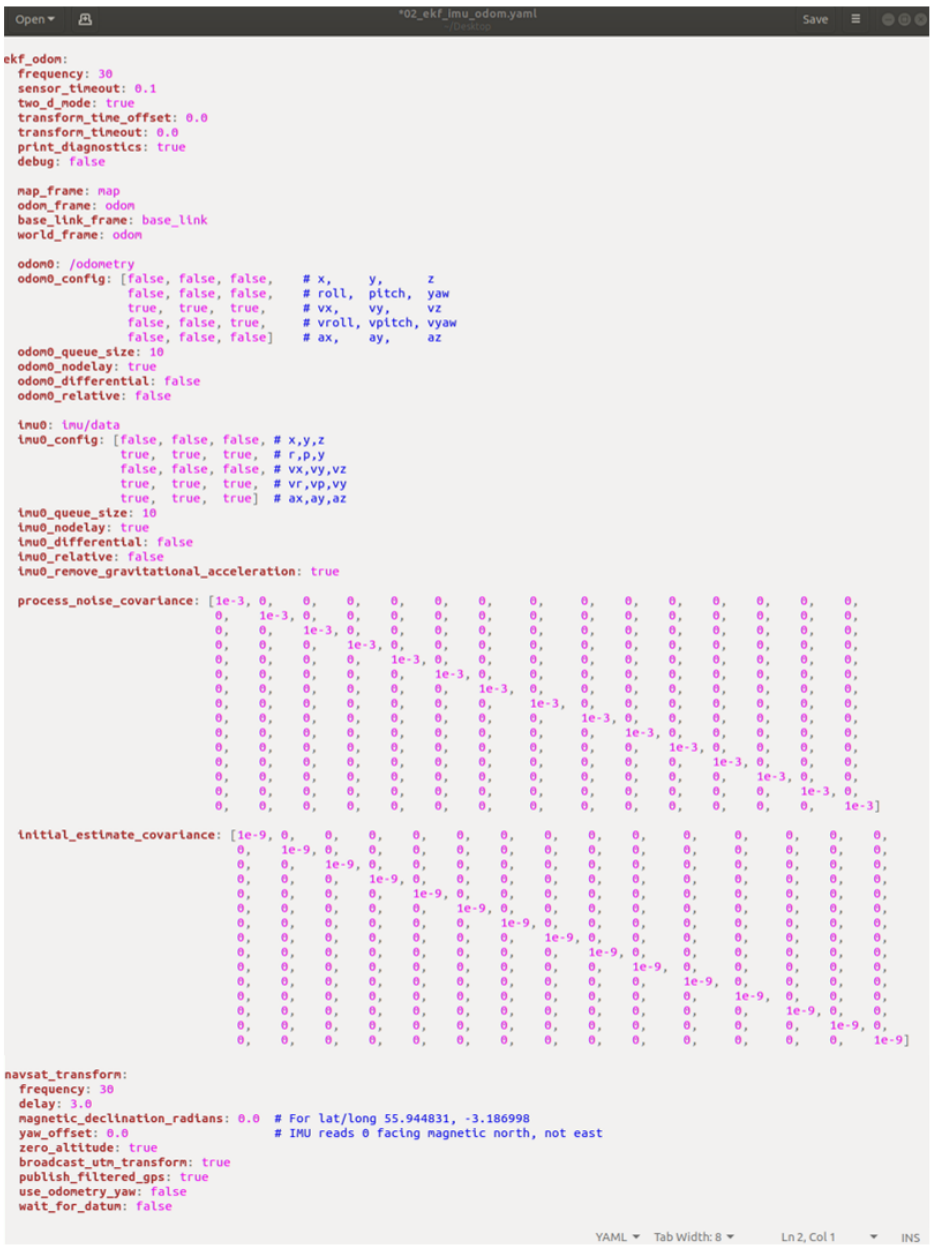
Appendix A

Codes and Configurations

A.1 odometry_publisher: ECEF to ENU Transform

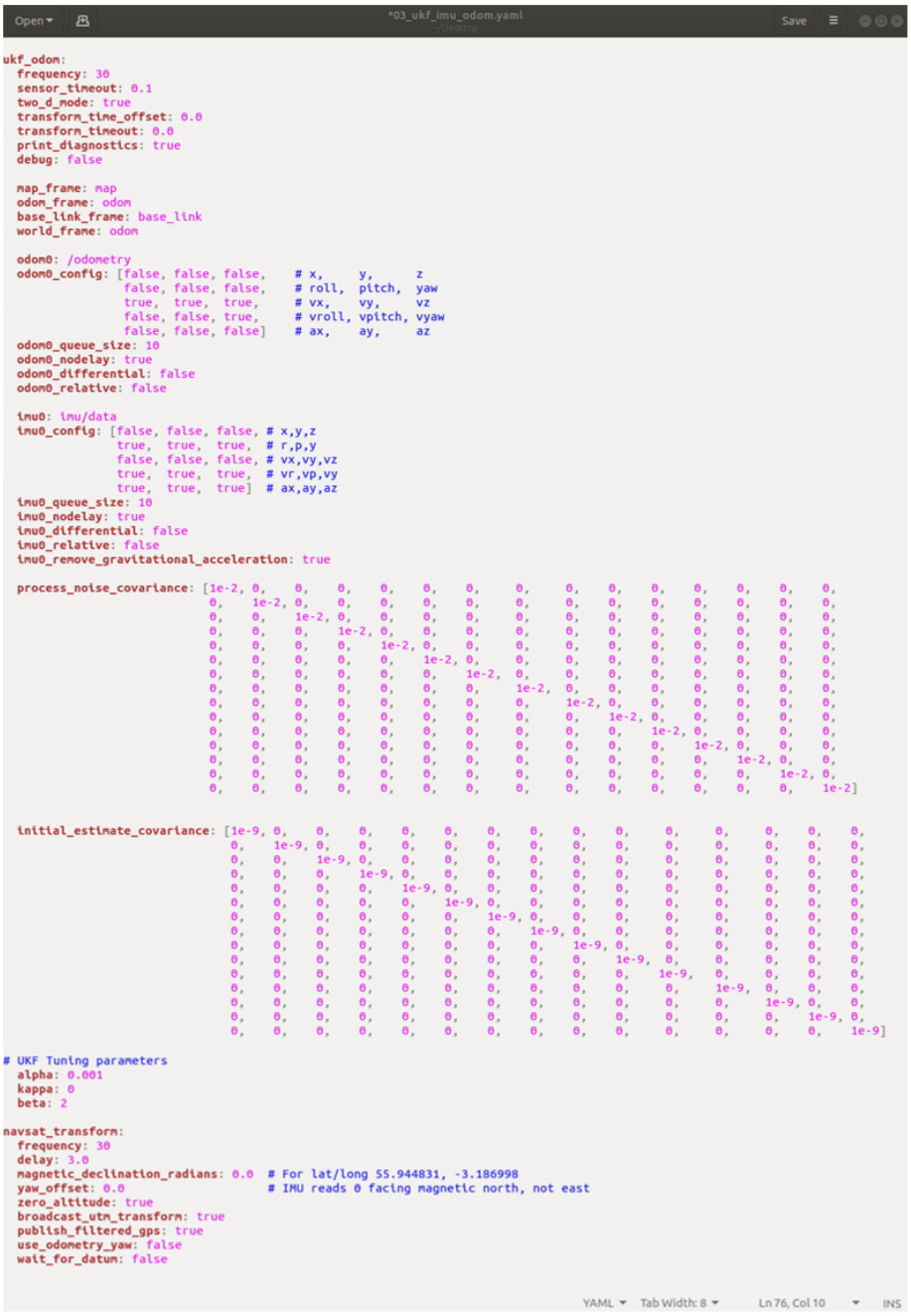
```
// WGS-84 geodetic constants
const double a = 6378137.0;         // WGS-84 Earth semimajor axis (m)

const double b = 6356752.314245;     // Derived Earth semiminor axis (m)
const double f = (a - b) / a;           // Ellipsoid Flatness
const double f_inv = 1.0 / f;       // Inverse flattening

//const double f_inv = 298.257223563; // WGS-84 Flattening Factor of the Earth
//const double b = a - a / f_inv;
//const double f = 1.0 / f_inv;

const double a_sq = a * a;
const double b_sq = b * b;
const double e_sq = f * (2 - f);    // Square of Eccentricity

// Converts the Earth-Centered Earth-Fixed (ECEF) coordinates (x, y, z) to
// East-North-Up coordinates in a Local Tangent Plane that is centered at the
// (WGS-84) Geodetic point (lat0, lon0, h0).
public static void EcefToEnu(double x, double y, double z,
                             double lat0, double lon0, double h0,
                             out double xEast, out double yNorth, out double zUp)
{
    // Convert to radians in notation consistent with the paper:
    var lambda = DegreesToRadians(lat0);
    var phi = DegreesToRadians(lon0);
    var s = Sin(lambda);
    var N = a / Sqrt(1 - e_sq * s * s);

    var sin_lambda = Sin(lambda);
    var cos_lambda = Cos(lambda);
    var cos_phi = Cos(phi);
    var sin_phi = Sin(phi);

    double x0 = (h0 + N) * cos_lambda * cos_phi;
    double y0 = (h0 + N) * cos_lambda * sin_phi;
    double z0 = (h0 + (1 - e_sq) * N) * sin_lambda;

    double xd, yd, zd;
    xd = x - x0;
    yd = y - y0;
    zd = z - z0;

    // This is the matrix multiplication
    xEast = -sin_phi * xd + cos_phi * yd;
    yNorth = -cos_phi * sin_lambda * xd - sin_lambda * sin_phi * yd + cos_lambda * zd;
    zUp = cos_lambda * cos_phi * xd + cos_lambda * sin_phi * yd + sin_lambda * zd;
}
```

A.2 robot_localization: EKF Configuration File



```
Open   *02_ekf_imu_odom.yaml   Save

ekf_odom:
  frequency: 30
  sensor_timeout: 0.1
  two_d_mode: true
  transform_time_offset: 0.0
  transform_timeout: 0.0
  print_diagnostics: true
  debug: false

  map_frame: map
  odom_frame: odom
  base_link_frame: base_link
  world_frame: odom

  odom0: /odometry
  odom0_config: [false, false, false,    # x,     y,     z
                 false, false, false,    # roll,  pitch, yaw
                 true,  true,  true,     # vx,    vy,    vz
                 false, false, true,     # vroll, vpitch, vyaw
                 false, false, false]    # ax,    ay,    az
  odom0_queue_size: 10
  odom0_nodelay: true
  odom0_differential: false
  odom0_relative: false

  imu0: imu/data
  imu0_config: [false, false, false, # x,y,z
                true,  true,  true,  # r,p,y
                false, false, false, # vx,vy,vz
                true,  true,  true,  # vr,vp,vy
                true,  true,  true]  # ax,ay,az
  imu0_queue_size: 10
  imu0_nodelay: true
  imu0_differential: false
  imu0_relative: false
  imu0_remove_gravitational_acceleration: true

  process_noise_covariance: [1e-3, 0,    0,    0,    0,    0,    0,     0,     0,    0,    0,    0,    0,    0,    0,
                             0,    1e-3, 0,    0,    0,    0,    0,     0,     0,    0,    0,    0,    0,    0,    0,
                             0,    0,    1e-3, 0,    0,    0,    0,     0,     0,    0,    0,    0,    0,    0,    0,
                             0,    0,    0,    1e-3, 0,    0,    0,     0,     0,    0,    0,    0,    0,    0,    0,
                             0,    0,    0,    0,    1e-3, 0,    0,     0,     0,    0,    0,    0,    0,    0,    0,
                             0,    0,    0,    0,    0,    1e-3, 0,     0,     0,    0,    0,    0,    0,    0,    0,
                             0,    0,    0,    0,    0,    0,    1e-3,  0,     0,    0,    0,    0,    0,    0,    0,
                             0,    0,    0,    0,    0,    0,    0,     1e-3,  0,    0,    0,    0,    0,    0,    0,
                             0,    0,    0,    0,    0,    0,    0,     0,     1e-3, 0,    0,    0,    0,    0,    0,
                             0,    0,    0,    0,    0,    0,    0,     0,     0,    1e-3, 0,    0,    0,    0,    0,
                             0,    0,    0,    0,    0,    0,    0,     0,     0,    0,    1e-3, 0,    0,    0,    0,
                             0,    0,    0,    0,    0,    0,    0,     0,     0,    0,    0,    1e-3, 0,    0,    0,
                             0,    0,    0,    0,    0,    0,    0,     0,     0,    0,    0,    0,    1e-3, 0,    0,
                             0,    0,    0,    0,    0,    0,    0,     0,     0,    0,    0,    0,    0,    1e-3, 0,
                             0,    0,    0,    0,    0,    0,    0,     0,     0,    0,    0,    0,    0,    0,    1e-3]

  initial_estimate_covariance: [1e-9, 0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
                                0,    1e-9, 0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
                                0,    0,    1e-9, 0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
                                0,    0,    0,    1e-9, 0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
                                0,    0,    0,    0,    1e-9, 0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
                                0,    0,    0,    0,    0,    1e-9, 0,    0,    0,    0,    0,    0,    0,    0,    0,
                                0,    0,    0,    0,    0,    0,    1e-9, 0,    0,    0,    0,    0,    0,    0,    0,
                                0,    0,    0,    0,    0,    0,    0,    1e-9, 0,    0,    0,    0,    0,    0,    0,
                                0,    0,    0,    0,    0,    0,    0,    0,    1e-9, 0,    0,    0,    0,    0,    0,
                                0,    0,    0,    0,    0,    0,    0,    0,    0,    1e-9, 0,    0,    0,    0,    0,
                                0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    1e-9, 0,    0,    0,    0,
                                0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    1e-9, 0,    0,    0,
                                0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    1e-9, 0,    0,
                                0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    1e-9, 0,
                                0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    1e-9]

navsat_transform:
  frequency: 30
  delay: 3.0
  magnetic_declination_radians: 0.0  # For lat/long 55.944831, -3.186998
  yaw_offset: 0.0                    # IMU reads 0 facing magnetic north, not east
  zero_altitude: true
  broadcast_utm_transform: true
  publish_filtered_gps: true
  use_odometry_yaw: false
  wait_for_datum: false

YAML  Tab Width: 8  Ln 2, Col 1  INS
```

A.3 robot_localization: UKF Configuration File

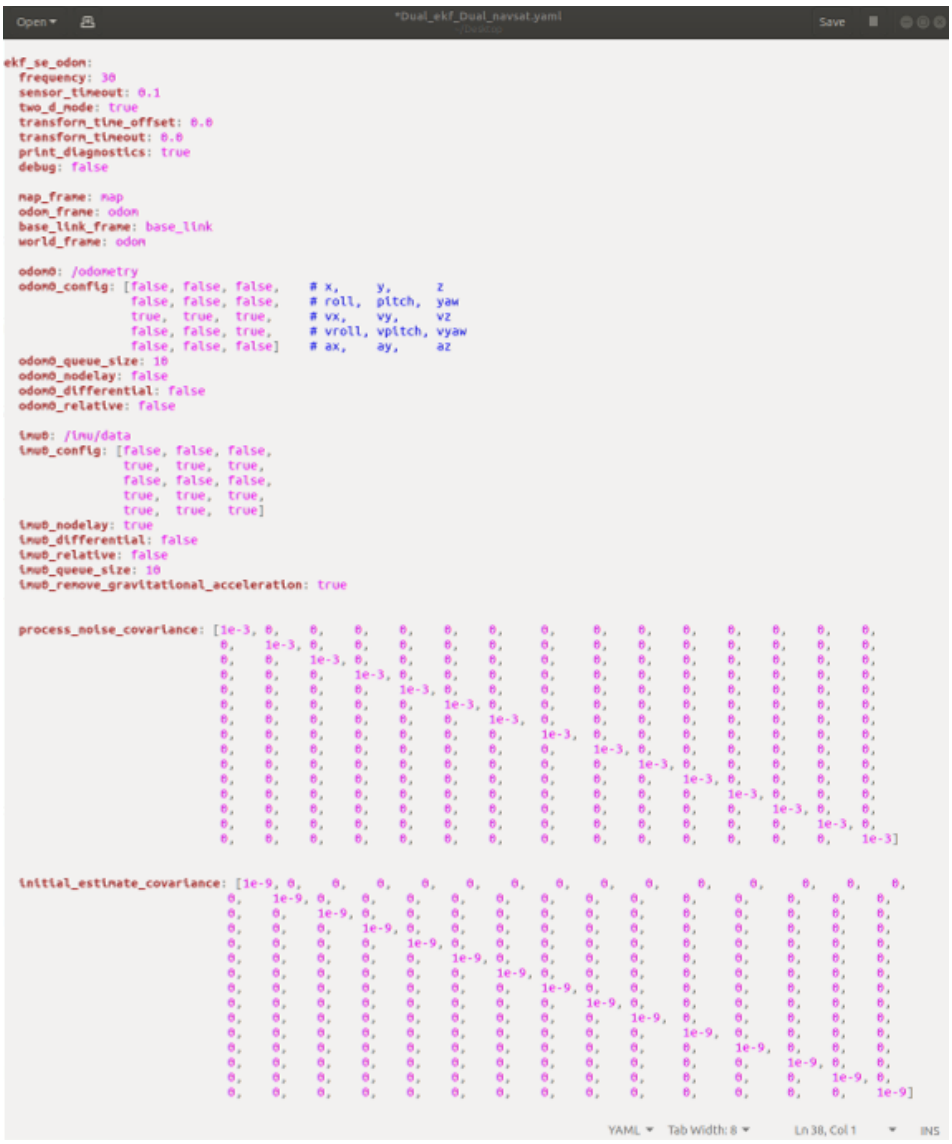


Open ▾ *03_ukf_imu_odom.yaml ~/Desktop Save

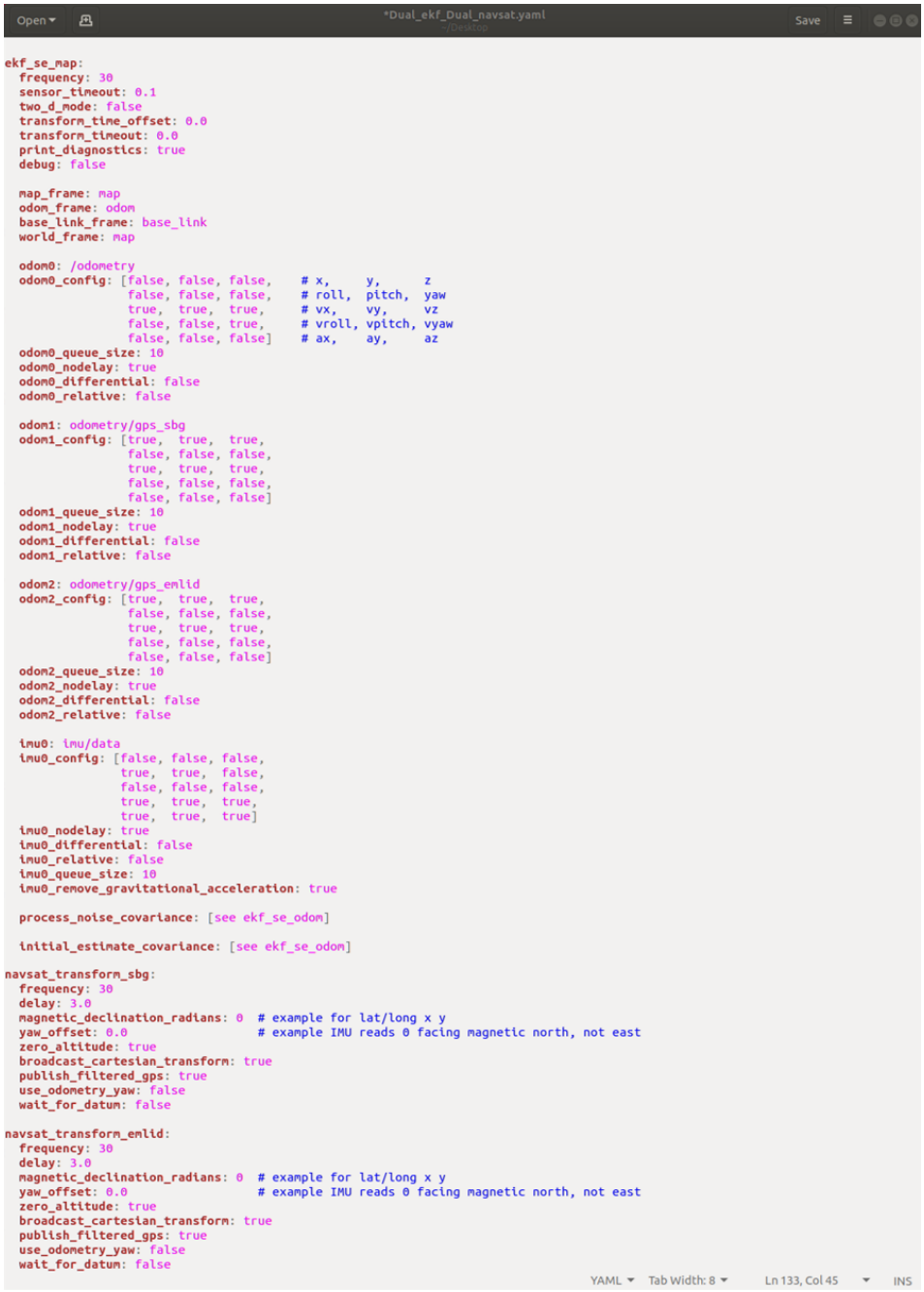
```
ukf_odom:
  frequency: 30
  sensor_timeout: 0.1
  two_d_mode: true
  transform_time_offset: 0.0
  transform_timeout: 0.0
  print_diagnostics: true
  debug: false

  map_frame: map
  odom_frame: odom
  base_link_frame: base_link
  world_frame: odom

  odom0: /odometry
  odom0_config: [false, false, false,    # x,     y,      z
                 false, false, false,    # roll,  pitch,  yaw
                 true,  true,  true,     # vx,    vy,     vz
                 false, false, true,     # vroll, vpitch, vyaw
                 false, false, false]    # ax,    ay,     az
  odom0_queue_size: 10
  odom0_nodelay: true
  odom0_differential: false
  odom0_relative: false

  imu0: imu/data
  imu0_config: [false, false, false, # x,y,z
                true,  true,  true,  # r,p,y
                false, false, false, # vx,vy,vz
                true,  true,  true,  # vr,vp,vy
                true,  true,  true]  # ax,ay,az
  imu0_queue_size: 10
  imu0_nodelay: true
  imu0_differential: false
  imu0_relative: false
  imu0_remove_gravitational_acceleration: true

  process_noise_covariance: [1e-2, 0,    0,    0,    0,    0,    0,     0,     0,    0,    0,    0,    0,    0,    0,
                             0,    1e-2, 0,    0,    0,    0,    0,     0,     0,    0,    0,    0,    0,    0,    0,
                             0,    0,    1e-2, 0,    0,    0,    0,     0,     0,    0,    0,    0,    0,    0,    0,
                             0,    0,    0,    1e-2, 0,    0,    0,     0,     0,    0,    0,    0,    0,    0,    0,
                             0,    0,    0,    0,    1e-2, 0,    0,     0,     0,    0,    0,    0,    0,    0,    0,
                             0,    0,    0,    0,    0,    1e-2, 0,     0,     0,    0,    0,    0,    0,    0,    0,
                             0,    0,    0,    0,    0,    0,    1e-2,  0,     0,    0,    0,    0,    0,    0,    0,
                             0,    0,    0,    0,    0,    0,    0,     1e-2,  0,    0,    0,    0,    0,    0,    0,
                             0,    0,    0,    0,    0,    0,    0,     0,     1e-2, 0,    0,    0,    0,    0,    0,
                             0,    0,    0,    0,    0,    0,    0,     0,     0,    1e-2, 0,    0,    0,    0,    0,
                             0,    0,    0,    0,    0,    0,    0,     0,     0,    0,    1e-2, 0,    0,    0,    0,
                             0,    0,    0,    0,    0,    0,    0,     0,     0,    0,    0,    1e-2, 0,    0,    0,
                             0,    0,    0,    0,    0,    0,    0,     0,     0,    0,    0,    0,    1e-2, 0,    0,
                             0,    0,    0,    0,    0,    0,    0,     0,     0,    0,    0,    0,    0,    1e-2, 0,
                             0,    0,    0,    0,    0,    0,    0,     0,     0,    0,    0,    0,    0,    0,    1e-2]


  initial_estimate_covariance: [1e-9, 0,    0,    0,    0,    0,    0,    0,    0,    0,     0,     0,    0,    0,    0,
                                0,    1e-9, 0,    0,    0,    0,    0,    0,    0,    0,     0,     0,    0,    0,    0,
                                0,    0,    1e-9, 0,    0,    0,    0,    0,    0,    0,     0,     0,    0,    0,    0,
                                0,    0,    0,    1e-9, 0,    0,    0,    0,    0,    0,     0,     0,    0,    0,    0,
                                0,    0,    0,    0,    1e-9, 0,    0,    0,    0,    0,     0,     0,    0,    0,    0,
                                0,    0,    0,    0,    0,    1e-9, 0,    0,    0,    0,     0,     0,    0,    0,    0,
                                0,    0,    0,    0,    0,    0,    1e-9, 0,    0,    0,     0,     0,    0,    0,    0,
                                0,    0,    0,    0,    0,    0,    0,    1e-9, 0,    0,     0,     0,    0,    0,    0,
                                0,    0,    0,    0,    0,    0,    0,    0,    1e-9, 0,     0,     0,    0,    0,    0,
                                0,    0,    0,    0,    0,    0,    0,    0,    0,    1e-9,  0,     0,    0,    0,    0,
                                0,    0,    0,    0,    0,    0,    0,    0,    0,    0,     1e-9,  0,    0,    0,    0,
                                0,    0,    0,    0,    0,    0,    0,    0,    0,    0,     0,     1e-9, 0,    0,    0,
                                0,    0,    0,    0,    0,    0,    0,    0,    0,    0,     0,     0,    1e-9, 0,    0,
                                0,    0,    0,    0,    0,    0,    0,    0,    0,    0,     0,     0,    0,    1e-9, 0,
                                0,    0,    0,    0,    0,    0,    0,    0,    0,    0,     0,     0,    0,    0,    1e-9]

# UKF Tuning parameters
  alpha: 0.001
  kappa: 0
  beta: 2

navsat_transform:
  frequency: 30
  delay: 3.0
  magnetic_declination_radians: 0.0  # For lat/long 55.944831, -3.186998
  yaw_offset: 0.0                    # IMU reads 0 facing magnetic north, not east
  zero_altitude: true
  broadcast_utm_transform: true
  publish_filtered_gps: true
  use_odometry_yaw: false
  wait_for_datum: false
```

YAML ▾ Tab Width: 8 ▾ Ln 76, Col 10 ▾ INS

A.4 robot_localization: Dual EKF Config. File



```
Open ▾   *Dual_ekf_Dual_navsat.yaml   Save

ekf_se_odom:
  frequency: 30
  sensor_timeout: 0.1
  two_d_mode: true
  transform_time_offset: 0.0
  transform_timeout: 0.0
  print_diagnostics: true
  debug: false

  map_frame: map
  odom_frame: odom
  base_link_frame: base_link
  world_frame: odom

  odom0: /odometry
  odom0_config: [false, false, false,    # x,     y,      z
                 false, false, false,    # roll,  pitch,  yaw
                 true,  true,  true,     # vx,    vy,     vz
                 false, false, true,     # vroll, vpitch, vyaw
                 false, false, false]    # ax,    ay,     az
  odom0_queue_size: 10
  odom0_nodelay: false
  odom0_differential: false
  odom0_relative: false

  imu0: /imu/data
  imu0_config: [false, false, false,
                true,  true,  true,
                false, false, false,
                true,  true,  true,
                true,  true,  true]
  imu0_nodelay: true
  imu0_differential: false
  imu0_relative: false
  imu0_queue_size: 10
  imu0_remove_gravitational_acceleration: true

  process_noise_covariance: [1e-3, 0,    0,    0,    0,    0,    0,     0,     0,    0,    0,    0,    0,    0,    0,
                             0,    1e-3, 0,    0,    0,    0,    0,     0,     0,    0,    0,    0,    0,    0,    0,
                             0,    0,    1e-3, 0,    0,    0,    0,     0,     0,    0,    0,    0,    0,    0,    0,
                             0,    0,    0,    1e-3, 0,    0,    0,     0,     0,    0,    0,    0,    0,    0,    0,
                             0,    0,    0,    0,    1e-3, 0,    0,     0,     0,    0,    0,    0,    0,    0,    0,
                             0,    0,    0,    0,    0,    1e-3, 0,     0,     0,    0,    0,    0,    0,    0,    0,
                             0,    0,    0,    0,    0,    0,    1e-3,  0,     0,    0,    0,    0,    0,    0,    0,
                             0,    0,    0,    0,    0,    0,    0,     1e-3,  0,    0,    0,    0,    0,    0,    0,
                             0,    0,    0,    0,    0,    0,    0,     0,     1e-3, 0,    0,    0,    0,    0,    0,
                             0,    0,    0,    0,    0,    0,    0,     0,     0,    1e-3, 0,    0,    0,    0,    0,
                             0,    0,    0,    0,    0,    0,    0,     0,     0,    0,    1e-3, 0,    0,    0,    0,
                             0,    0,    0,    0,    0,    0,    0,     0,     0,    0,    0,    1e-3, 0,    0,    0,
                             0,    0,    0,    0,    0,    0,    0,     0,     0,    0,    0,    0,    1e-3, 0,    0,
                             0,    0,    0,    0,    0,    0,    0,     0,     0,    0,    0,    0,    0,    1e-3, 0,
                             0,    0,    0,    0,    0,    0,    0,     0,     0,    0,    0,    0,    0,    0,    1e-3]

  initial_estimate_covariance: [1e-9, 0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
                                0,    1e-9, 0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
                                0,    0,    1e-9, 0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
                                0,    0,    0,    1e-9, 0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
                                0,    0,    0,    0,    1e-9, 0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
                                0,    0,    0,    0,    0,    1e-9, 0,    0,    0,    0,    0,    0,    0,    0,    0,
                                0,    0,    0,    0,    0,    0,    1e-9, 0,    0,    0,    0,    0,    0,    0,    0,
                                0,    0,    0,    0,    0,    0,    0,    1e-9, 0,    0,    0,    0,    0,    0,    0,
                                0,    0,    0,    0,    0,    0,    0,    0,    1e-9, 0,    0,    0,    0,    0,    0,
                                0,    0,    0,    0,    0,    0,    0,    0,    0,    1e-9, 0,    0,    0,    0,    0,
                                0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    1e-9, 0,    0,    0,    0,
                                0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    1e-9, 0,    0,    0,
                                0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    1e-9, 0,    0,
                                0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    1e-9, 0,
                                0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    1e-9]

YAML ▾   Tab Width: 8 ▾   Ln 38, Col 1 ▾   INS
```


Open
*Dual_ekf_Dual_navsat.yaml
~/Desktop
Save
ekf_se_map:
frequency: 30
sensor_timeout: 0.1
two_d_mode: false
transform_time_offset: 0.0
transform_timeout: 0.0
print_diagnostics: true
debug: false
map_frame: map
odom_frame: odom
base_link_frame: base_link
world_frame: map
odom0: /odometry
odom0_config: [false, false, false, # x, y, z
false, false, false, # roll, pitch, yaw
true, true, true, # vx, vy, vz
false, false, true, # vroll, vpitch, vyaw
false, false, false] # ax, ay, az
odom0_queue_size: 10
odom0_nodelay: true
odom0_differential: false
odom0_relative: false
odom1: odometry/gps_sbg
odom1_config: [true, true, true,
false, false, false,
true, true, true,
false, false, false,
false, false, false]
odom1_queue_size: 10
odom1_nodelay: true
odom1_differential: false
odom1_relative: false
odom2: odometry/gps_emlid
odom2_config: [true, true, true,
false, false, false,
true, true, true,
false, false, false,
false, false, false]
odom2_queue_size: 10
odom2_nodelay: true
odom2_differential: false
odom2_relative: false
imu0: imu/data
imu0_config: [false, false, false,
true, true, false,
false, false, false,
true, true, true,
true, true, true]
imu0_nodelay: true
imu0_differential: false
imu0_relative: false
imu0_queue_size: 10
imu0_remove_gravitational_acceleration: true
process_noise_covariance: [see ekf_se_odom]
initial_estimate_covariance: [see ekf_se_odom]
navsat_transform_sbg:
frequency: 30
delay: 3.0
magnetic_declination_radians: 0 # example for lat/long x y
yaw_offset: 0.0 # example IMU reads 0 facing magnetic north, not east
zero_altitude: true
broadcast_cartesian_transform: true
publish_filtered_gps: true
use_odometry_yaw: false
wait_for_datum: false
navsat_transform_emlid:
frequency: 30
delay: 3.0
magnetic_declination_radians: 0 # example for lat/long x y
yaw_offset: 0.0 # example IMU reads 0 facing magnetic north, not east
zero_altitude: true
broadcast_cartesian_transform: true
publish_filtered_gps: true
use_odometry_yaw: false
wait_for_datum: false
YAML
Tab Width: 8
Ln 133, Col 45
INS

List of Figures

List of Tables

Bibliography

[1] T. Luettel, M. Himmelsbach, and H. Wuensche. Autonomous Ground Vehicles—Concepts and a Path to the Future. *Proceedings of the IEEE*, 100(Special Centennial Issue):1831–1839, May 2012.

[2] E. Fraedrich and B. Lenz. Automated Driving: Individual and Societal Aspects. *Transportation Research Record*, 2416(1):64–72, 2014.

[3] A. Ziębiński, R. Cupek, D. Grzechca, and L. Chruszczyk. Review of advanced driver assistance systems (ADAS). In *AIP Conference Proceedings*, volume 1906, page 120002, 2017.

[4] A. Bonfitto, S. Feraco, A. Tonoli, and N. Amati. Combined regression and classification artificial neural networks for sideslip angle estimation and road condition identification. *Vehicle System Dynamics*, 58(11):1766–1787, 2020.

[5] S. Luciani, A. Bonfitto, N. Amati, and A. Tonoli. Comfort-Oriented Design of Model Predictive Control in Assisted and Autonomous Driving, Aug 2020.

[6] S. Hörl, F. Ciari, and K. W. Axhausen. Recent perspectives on the impact of autonomous vehicles. *Arbeitsberichte Verkehrs- und Raumplanung*, 1216, 2016.

[7] K. Kaur and G. Rampersad. Trust in driverless cars: Investigating key factors influencing the adoption of driverless cars. *Journal of Engineering and Technology Management*, 48:87–96, 2018.

[8] On-Road Automated Driving (ORAD) committee. Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. *SAE International*, page 35, 2018.

[9] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis. A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications. *IEEE Internet of Things Journal*, 5(2):829–846, 2018.

[10] A. Woo, B. Fidan, and W. W. Melek. *Localization for Autonomous Driving*, chapter 29, pages 1051–1087. John Wiley & Sons, Ltd, 2018.

[11] S. Feraco, A. Bonfitto, I. Khan, N. Amati, and A. Tonoli. Optimal Trajectory Generation Using an Improved Probabilistic Road Map Algorithm for Autonomous Driving, Aug 2020.

[12] S. Feraco, S. Luciani, A. Bonfitto, N. Amati, and A. Tonoli. A local trajectory planning and control method for autonomous vehicles based on the RRT algorithm. In *2020 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE)*, pages 1–6, 2020.

[13] Formula Student Germany. Formula Student Rules 2020, 2020.

[14] P. Marín-Plaza, A. Hussein, D. Martín, and A. D. L. Escalera. Global and Local Path Planning Study in a ROS-Based Research Platform for Autonomous Vehicles. *Journal of Advanced Transportation*, 2018:1–10, 2018.

[15] J. Kabzan, M. I. Valls, V. J. F. Reijgwart, H. F. C. Hendrikx, C. Ehmke, M. Prajapat, A. Bühler, N. Gosala, M. Gupta, R. Sivanesan, A. Dhall, E. Chisari, N. Karnchanachari, S. Brits, M. Dangel, I. Sa, R. Dubé, A. Gawel, M. Pfeiffer, A. Liniger, J. Lygeros, and R. Siegwart. AMZ Driverless: The full autonomous racing system. *Journal of Field Robotics*, 37(7):1267–1294, 2020.

[16] S. Nekkah, J. Janus, M. Boxheimer, L. Ohnemus, S. Hirsch, B. Schmidt, Y. Liu, D. Borbély, F. Keck, K. Bachmann, and L. Bleszynski. The Autonomous Racing Software Stack of the KIT19d. *arXiv e-prints*, page arXiv:2010.02828, Oct 2020.

[17] M. Zeilinger, R. Hauk, M. Bader, and A. Hofmann. Design of an Autonomous Race Car for the Formula Student Driverless (FSD). 2017.

[18] T. Chen, Z. Li, Y. He, Z. Xu, Z. Yan, and H. Li. From perception to control: an autonomous driving system for a formula student driverless car. *arXiv e-prints*, page arXiv:1909.00119, Aug 2019.

[19] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[20] O. Garcia-Bedoya and J. Ferreira. Sensor fusion tests for an autonomous vehicle, using extended kalman filter. *Journal of Engineering Science and Technology Review*, 11:1–8, 04 2018.

[21] X. Meng, H. Wang, and B. Liu. A Robust Vehicle Localization Approach Based on GNSS/IMU/DMI/LiDAR Sensor Fusion for Autonomous Vehicles. *Sensors*, 17(9), 2017.

[22] M. Lin, J. Yoon, and B. Kim. Self-Driving Car Location Estimation Based on a Particle-Aided Unscented Kalman Filter. *Sensors*, 20(9), 2020.

[23] M. Kok, J. D. Hol, and T. B. Schon. Using Inertial Sensors for Position and Orientation Estimation. *Foundations and Trends in Signal Processing*, 11(1-2):1–153, 2017.

[24] M. S. Grewal, L. R. Weill, and A. P. Andrews. *Global Positioning Systems, Inertial Navigation, and Integration.* John Wiley & Sons, Ltd, 2001.

[25] J. Fraden. *Handbook of Modern Sensors.* Springer-Verlag New York, 2004.

[26] M. J. McGrath and C. N. Scanaill. *Sensing and Sensor Fundamentals.* Apress, 2013.

[27] P. S. Maybeck. *Stochastic models, estimation, and control*, volume 141 of *Mathematics in Science and Engineering.* 1979.

[28] H.B. Mitchell. *Multi-Sensor Data Fusion.* Springer-Verlag Berlin Heidelberg, 2007.

[29] Burgard W. Fox D. Thrun, S. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents Series).* The MIT Press, 2006.

[30] S. J. Julier and J. K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[31] V. Imani, K. Haataja, and P. Toivanen. Three main paradigms of simultaneous localization and mapping (slam) problem. page 74, 04 2018.

[32] L. D'Alfonso, W. Lucia, P. Muraca, and P. Pugliese. Mobile Robot Localization via EKF and UKF. *Robot. Auton. Syst.*, 74(PA):122–127, Dec 2015.

[33] U. Kiencke and L. Nielsen. *Automotive Control Systems*. Springer, 2005.

[34] I. Khan, S. Feraco, A. Bonfitto, and N. Amati. A Model Predictive Control Strategy for Lateral and Longitudinal Dynamics in Autonomous Driving, Aug 2020.

[35] N. H. Amer, H. Zamzuri, K. Hudha, and Z. A. Kadir. Modelling and Control Strategies in Path Tracking Control for Autonomous Ground Vehicles: A Review of State of the Art and Challenges. *Journal of Intelligent & Robotic Systems*, 86(2):225–254, 2017.

[36] S. Feraco, A. Bonfitto, N. Amati, and A. Tonoli. Combined Lane Keeping and Longitudinal Speed Control for Autonomous Driving, aug 2019.

[37] R. Rajamani. *Vehicle Dynamics and Control*. Springer Science & Business Media, 2012.

[38] B. Barshan and H. F. Durrant-Whyte. Inertial navigation systems for mobile robots. *IEEE Transactions on Robotics and Automation*, 11(3):328–342, 1995.

[39] T. Lazarou and C. Danezis. Assessment of modern smartphone sensors performance on vehicle localization in urban environments. In *Fifth International Conference on Remote Sensing and Geoinformation of the Environment (RSCy2017)*, volume 10444, pages 560 – 570. International Society for Optics and Photonics, SPIE, 2017.

[40] Mo. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng. Ros: an open-source robot operating system. volume 3, 01 2009.

[41] ROS.org. sbg_driver for ros. `http://wiki.ros.org/sbg_driver`, 2020.

[42] T. Moore and D. Stouch. A Generalized Extended Kalman Filter Implementation for the Robot Operating System. In *IAS*, 2016.

[43] ROS.org. robot_localization wiki. `http://docs.ros.org/en/melodic/api/robot_localization/html/index.html`, 2020.

[44] ROS.org. The robot_localization package for ros. `http://wiki.ros.org/robot_localization`, 2020.

[45] T. Foote. tf: The transform library. In *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*, pages 1–6, 2013.

[46] ROS.org Documentation. https://www.ros.org/reps/rep-0103.html.

[47] ROS.org Documentation. https://www.ros.org/reps/rep-0105.html.

[48] G. Welch and G. Bishop. An Introduction to the Kalman Filter. Technical report, 1995.

[49] E. A. Wan and R. Van Der Merwe. The unscented Kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, pages 153–158, 2000.

[50] EMLID. Reach m+. https://docs.emlid.com/reachm-plus/, 2020.

[51] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, 1996.

[52] STEREOLABS. Zed stereo camera. https://www.stereolabs.com/zed/, 2020.

[53] J. R. Kidd. Performance evaluation of the velodyne vlp-16 system for surface feature surveying. 2017.

[54] EMLID. Emlid ntrip caster. https://caster.emlid.com/, 2020.